BOLT BERANEK

This document describes services which a Message Transfer System (MTS) might make available to its users. CCITT Study Group VII, Question 5's "Living Document" served as a basis for this work.

For each service listed, there is a definition, a comparison with the CCITT definition from which it was drawn, and a compliance classification. The compliance classification indicates if the service is REQUIRED (must always be provided by all MSAs), BASIC (must be provided by all MSAs), OPTIONAL (defined, but need not be provided), or for future study.

## Address_Information_Enquiry (SF62)

Compliance Class: Basic — Cooperating MSA Service

### Comments_or_Changes

The original seemed to treat address as a synonym of delivery slot, which doesn't fit the model.

### Revised_Service_Statement

An address information enquiry is a request for the address associated with a given name belonging to a potential message recipient. The address is determined by consulting a data base which associates names with addresses. An address information enquiry is initiated by a UA or an MSA and processed by the MTS. The exact location within the MTS at which processing takes place depends upon the architectures of the MTS and the data base containing the names and addresses.

A number of responses to an address information enquiry are possible, depending on the information in the data base.

- o No potential recipient is associated with the name. The service returns a failure response.

- The name is associated with just one potential recipient. The service returns that recipient's address.
- The name is associated with more than one potential recipient. The service returns one of three responses, depending on UA options.
  - An "ambiguous name" error.
  - The addresses of all potential recipients.
  - The full names and addresses of all potential recipients.

This service has also been referred to as directory assistance.

## Advice_on_Recipient_Content_Capabilities (SF76)

Compliance Class: Future Cooperating MSA Service

### Comments_or_Changes

The original treated this as an end-user capability rather than as an MSA service.

### Revised_Service_Statement

This service is invoked by the originator's UA or an MSA to obtain the list of content types which can be presented to a specific recipient.

## Audit_Trail (SF72)

Compliance Class: Basic Cooperating MSA Service

### Comments_or_Changes

Making the audit trail information part of the envelope (with intended implications about the scope of the information).

### Revised_Service_Statement

The audit trail service provides detailed information for testing, controlling, and verification purposes. The information necessary to provide this service is part of the envelope of a message.

## Charging_Information (SF72)

Compliance Class: Future Cooperating MSA Service

### Comments_or_Changes

Reworded to emphasize that the information goes to UA, not to end-user directly.

### Revised_Service_Statement

The charging information service provides information to the UA about the charge for providing a message service, such as sending a message.

## Closed_User_Group (SF63)

Compliance Class: Optional Cooperating MSA Service

### Comments_or_Changes

No changes.

### Revised_Service_Statement

A closed user group is a group of message originators and recipients whose abilities to originate and receive messages are restricted. There are two restrictions which may be placed upon a member of a closed user group:

1. A member may send messages only to other members of the same closed user group.

2. A member may receive messages only from other members of the same closed user group.

If both restrictions are applied, then no messages may pass between a closed user group member and a non-member.

If only the first restriction is applied, then a member may receive messages from a non-member. This allows an administrative announcement, for example, to be distributed to all members of a set of closed user groups. A member of such a closed user group cannot communicate with a member of another closed user group who is not also in the first member's group.

If only the second restriction is applied, then a member may send messages to a non-member. This allows an administrator, for example, to distribute a notice to others outside his group, while protecting him from unwanted incoming messages. A member of such a closed user group cannot communicate with a member of another closed user group who is not also in the first member's group.

## Content_Converted_Indication (SF80)

Compliance Class: Future Cooperating MSA Service

### Comments_or_Changes

Revised to emphasize that conversion takes place within MTS rather than at the recipient's UA.

### Revised_Service_Statement

This service is used to inform a recipient's UA that a conversion function was executed while the message was in the MTS.

The initial and final content types are indicated to the recipient's UA.

## Content_Type_Incompatibility_Indication (SF81)

Compliance Class: Future Cooperating MSA Service

### Comments_or_Changes

The original mentioned "non-receipt notification", which is not consistent with the model (because MSA cannot detect receipt). The original assumed that messages with incompatible content types are not delivered, which is not a necessary restriction.

### Revised_Service_Statement

This service must be associated with a non-delivery notification service.

This service indicates to an originator that the content type of a message is incompatible with the recipient content capabilities. The message might be delivered anyhow, or it might be returned, depending on UA options (to be determined).

See also the service "Advice on Recipient Content Capabilities."

## Delivery_Cancellation (SF7)

**Compliance Class: Basic** Cooperating MSA Service

### Comments_or_Changes

Removed specification that originator identify message to be cancelled by using its message identifier. Question is open, depending on what information goes on the envelope.

### Revised_Service_Statement

This service allows the originator's UA to request that the MTS stop the delivery of a previously posted message. If the message has already been delivered, then the cancellation attempt fails. The message originator is notified of the success or failure of the cancellation attempt. (It might be difficult to cancel the delivery of messages that were not sent using timed delivery.)

The service could return a copy of the cancelled message to the originator. This would allow the originator to modify the message and resubmit it for delivery.

## Delivery_Notification (SF5)

Compliance Class: Optional Cooperating MSA Service

### Comments_or_Changes

Terminology change: mail item => message; message identifier => transaction identifier. Clarified meaning of notification for multi-address messages.

### Revised_Service_Statement

The originator of a message may request that delivery of that message be confirmed. The delivery notification indicates that the delivery MSA has successfully delivered the message to the recipient's UA (that is, that there has been a positive confirmation of delivery by the recipient's UA; acknowledgment by a lower-level protocol is not adequate). Delivery notification does not carry any implication that any user action, such as reading the message contents, has taken place. Delivery notification is sent to the originator's UA. It is related to the original message by means of the transaction identifier and contains the date and time of delivery. In the case of a multi-address message, a delivery notification refers to a specific copy of the original message but the service may be invoked for any or all recipients.

## Distribution_Lists (SF29)

Compliance Class: Basic                Cooperating MSA Service

### Comments_or_Changes

Rewording. Removed the capability to specify a group of recipients by specifying attributes because this should be a separate service.

### Revised_Service_Statement

This service enables the originator to specify that a message is to be delivered to a set of recipients by assigning a reference name to that set. The originator specifies the reference name rather than specifying each recipient explicitly. The service does not imply simultaneous delivery to all recipients.

This service is supported by a distribution list data base. A distribution list might be accessible only to an individual originator, or to a wider circle of users. Additional services must be provided that allow creation, deletion, and modification of distribution lists.

Various other services can be invoked on a per-recipient basis.

## Encryption (SF64)

Compliance Class: Optional Cooperating MSA Service

### Comments or Changes

Removed the ability to encrypt the envelope because that causes more problems than it solves. In addition, the MSA is supposed to be trusted.

### Revised Service Statement

Encryption is the enciphering of all or part of a message. Encryption is used to keep a message from being "understood" by those who are not authorized to inspect it. Only the message contents can be encrypted, not the message envelope.

## Guaranteed Delivery (SF86)

Compliance Class: Optional Cooperating MSA Service

### Comments or Changes

Terminology change: replace "service provider" with "MTS" or "MTS administration" in order to clarify responsibility. Some minor rewording.

### Revised Service Statement

With guaranteed delivery, the MTS undertakes to deliver the message before a certain time (this could be indefinite). Failure to achieve this renders the MTS administration liable to compensation for any financially quantifiable loss by the "customer" (originator, recipient, or some other party). This service covers what sometimes has been referred to as registered mail. However, "registered mail" has also been used to describe delivery notification or receipt notification, and so its use has been avoided here.

Guaranteed delivery is roughly analogous to the postal insured mail service, which is normally used to indemnify the customer against loss of some valuable object being transported by the postal service. However, in the case of electronic message transfer, it is less likely that the transferred message will have any intrinsic value, as it will generally be possible for the originator to maintain a copy of any valuable data. It is much more likely for a financial loss to be incurred by late delivery (for example, a deadline may have passed or a late payment may incur interest charges).

Providing this service could cause legal and other problems, particularly where a message is transferred between several MTS administrations. The service is probably necessary, however, for certain applications, such as electronic funds transfer, that involve financial transactions. This service requires further study; it is included now for completeness.

## Priority_Delivery (SF84)

Compliance Class: Optional — Cooperating MSA Service

### Comments_or_Changes

Formerly called Low-Priority Delivery. The description has been generalized to make low priority simply one case in a facility for specifying priority for a message.

### Revised_Service_Statement

Priority delivery allows the originator to specify the urgency with which the message is to be delivered. This requires that messages with a higher priority be sent first. It is intended that the higher the priority the greater the cost to the sender. Although no particular time until delivery is guaranteed, each class of service would have a different probable time until delivery, with the lower the priority the greater the time until delivery. Low priority may be of value to the MTS in that it can wait for other messages before setting up an expensive link, or it could wait for a more favorable time to send the message.

## Multi-Address/Name_Delivery (SF28)

Compliance Class: Optional Cooperating MSA Service

### Comments_or_Changes

The reason for needing this service is poorly defined. It depends on some unstated assumptions about posting and the form of envelopes and messages after posting.

### Revised_Service_Statement

This service enables the originator to specify that a message is to be delivered to more than one recipient, where each recipient is specified explicitly by the originator. Simultaneous delivery to all recipients is not implied by the service.

Various other services may be invoked on a per-recipient basis.

## Name/Address/Route_Selection (SF65)

Compliance Class: Optional Cooperating MSA Service

### Comments_or_Changes

Reworded to clarify the different levels of responsibility of end-user, UA, and MSA.

### Revised_Service_Statement

This service specifies an intended message recipient, according to the slot at which delivery of a message should take place and the route a message should take as it is transferred there. An originator, an originator's UA, and an MSA are all permitted to specify a route. Only an MSA is required to be able to do so. A message originator is not expected to supply routing information except in exceptional cases.

This service might be used in conjunction with an address information enquiry service.

## Normal_Delivery (SF15)

Compliance Class: Required Cooperating MSA Service

### Comments_or_Changes

Added discussion of total permissible delay. Added distinction that non-delivery notification is optional (at UA's request).

### Revised_Service_Statement

When an originator's UA requests the normal delivery service, it requests the MTS to take responsibility for the transfer of a message from the posting slot to the delivery slot and for the delivery of that message to the recipient's UA.

Normal delivery involves two components to transfer delay. The first component is that contributed by the MTS, namely that experienced in transferring the message from the Posting MSA to the delivery MSA. The normal action is to pass the message through the MTS without undue delay. The maximum permissible delay for this part of normal delivery will be defined. The second delay component is that which may be experienced while the recipient's UA is temporarily not operational. The normal action is to pass the message to the recipient's UA as soon as the delivery MSA is in contact with it. The maximum permissible delay for this second part of normal delivery will also be specified. It is apt to be considerably longer than that for the first delay component. If the maximum permissible delay is exceeded, the messsage is considered undeliverable. Total permissible delay will also be specified, and can be less than the sum of the component maximum delays.

An originator's UA (optionally) receives non-delivery notification when a message cannot be delivered to the recipient's UA. The exact circumstances under which a message is to be considered undeliverable are as yet unspecified, but a few examples are the following:

1. The recipient is unknown to the MTS. This case will probably be due to an error on the part of the originator or because the intended recipient is no longer a user.

2. Message transfer between the originator and the recipient is not permitted, for example, due to closed user group restrictions.

3. The recipient's UA refuses to accept the message (for example, because of content type incompatibility). It may refuse all messages or select only certain messages.

4. The originator specified a latest delivery time in the timed delivery service, which has now passed.

5. The recipient's UA has been out of service, or otherwise unable to accept delivery, for a prolonged time.

Other delivery-related services (for example, priority) either modify or enhance the behavior of the normal delivery service.

## On-line_Subscription (SF67)

Compliance Class: Future Cooperating MSA Service

### Comments_or_Changes

Removed original implication that subscription was an end-user capability. Still needs something to indicate relation between this service and any address data base service.

### Revised_Service_Statement

The on-line subscription service enables a user agent to enter a request for a subscription to the message handling service. The user agent specifies the duration of the subscription. Handling requests for subscriptions is a matter of administrative policy.

## Originator-Requested_Alternative_Recipient (SF12)

Compliance Class: Optional Cooperating MSA Service

### Comments_or_Changes

Unchanged so far. Objection: This service description assumes a single recipient. The situation is more complex with more recipients or with groups of recipients. The question needs more study.

### Revised_Service_Statement

The originator may specify an alternative recipient to which the message will be sent if it cannot be delivered to the UA of the intended recipient. More than one alternative recipient may be specified. The criteria for determining message non-delivery need to be developed.

## Posting_Time_Stamp (SF88)

Compliance Class: Required Cooperating MSA Service

### Comments_or_Changes

Not supplied in original.

### Revised_Service_Statement

This service adds the date and time to the envelope of a message at the end of the posting protocol.

## Prevention_of_Non-Delivery_Notification (SF82)

**Compliance Class: Optional** **Cooperating MSA Service**

### Comments_or_Changes

Not supplied in original.

### Revised_Service_Statement

This service allows the originator of a message to request that the MTS not provide notification when it fails to deliver a message.

## Proof_of_Posting_or_Delivery (SF85)

Compliance Class: Future Cooperating MSA Service

### Comments_or_Changes

Removed lengthy paragraph speculating on how MSA would implement the service.

### Revised_Service_Statement

This service allows the originator of a message to prove to the intended recipient or a third party that the message was indeed posted and/or delivered to the intended recipient's UA. The proof depends on all parties trusting the MTS and is intended to help resolve disputes which arise when one user claims to have sent a message and the alleged recipient claims not to have received it.

This could range from a simple confirmation that a message with the supplied transaction identifier was posted, to a complete copy of the message with a delivery record identifying the UA to which it was successfully delivered.

The standardization of the presentation of this proof to the requestor (that is, originator, recipient, or third party) is for further study.

## Recipient-Requested_Redirection (SF13)

Compliance Class: Optional Cooperating MSA Service

### Comments_or_Changes

Reworded the examples. The question of defining a "different UA" within the logical model requires serious study.

### Revised_Service_Statement

Recipient-controlled redirection services require further definition. Examples of such services are:

1. Designating another recipient to receive one's messages and act on one's behalf.

2. Redirecting messages, permanently or temporarily, to the recipient at a different UA.

Note: This service should only be classified as delivery MSA.

## Repeated_Timed_Delivery (SF2)

Compliance Class: Optional Cooperating MSA Service

### Comments_or_Changes

Added requirement for permanent storage service.

### Revised_Service_Statement

An originator may cause a message to be delivered at several specified times or at a specified frequency. Although the process is automatic, its effect is the same as that of manually sending the message for delivery at each of the specified times. This service could be used for automatic announcements of regular meetings.

This service assumes a permanent storage service.

## Resource_Status_and_Resource_Warning (SF38)

Compliance Class: Future Cooperating MSA Service

### Comments_or_Changes

Added sentence to point out distinction between local system resources and MTS resources.

### Revised_Service_Statement

The status and availability of certain system resources can significantly impact the services and facilities which are offered by MHF. For example, the failure of a communications line may result in decreased message throughput and/or the complete suspension of message exchanges between certain end users. When a shortage of memory resources occurs, the number and/or size of posted messages may be restricted throughout the MHF domain, or limitations may be imposed upon the message preparation and editing services provided to one or some subset of users. The consequences of resource status changes are different in different MHF environments and depend upon such factors as the physical distribution of resources, allocation algorithms, and the existence of alternative or "back-up" facilities.

Since it is actually the conspicuous effect on services which is of primary concern to the MHF user, it is in this context that resource status and resource warning facilities are best provided. An MSA service can provide information only concerning MTS resources. Information about local system resources must be handled by either the local UA or by local system facilities outside the MTS.

As an illustration, a user might be able to obtain upon request, information on the amount of space which remains available to him for storing previously received messages. Similarly, it might be desirable to warn users automatically when only a limited number of additional messages can be posted or stored, or that message throughput for the present period has become seriously degraded.

Note: This service is also classified as Cooperating UA, Local MSA, and Local UA.

## Security_Classification (SF60)

Compliance Class: Future Cooperating MSA Service

### Comments_or_Changes

Note: This cannot be an MSA service. It should be classified only as Cooperating UA.

### Revised_Service_Statement

Security classification is a service whereby an indicator is associated with a message in order to determine the security functions which must be applied to it. Examples of such security functions are encryption, message decay, and access control mechanisms.

Note: This service is also classified as Cooperating UA.

## Timed_Delivery (SF1)

Compliance Class: Optional Cooperating MSA Service

### Comments_or_Changes

Added part about MTS being able to reject service requests that it could not guarantee.

### Revised_Service_Statement

An originator may constrain the date and time of a message's delivery, for example, to request that the message be delivered no sooner than a specified date and time, no later than a specified date and time, or both. In the first case, delivery will take place as close to the date and time specified as possible, but not before. In the second case, delivery is cancelled if unaccomplished by the specified date and time. Using an indication of both the earliest and latest delivery date and time allows for delivery at a specified instant with some minimum tolerance. The issue of tolerances requires further study. Users must not be able to request services that the MTS cannot guarantee.

Timed delivery is realized within the MTS, rather than the recipient's UA, in order to maintain the certainty that the message will not be available to the recipient at an earlier instant.

## Urgent_Delivery (SF6)

Compliance Class: Optional — Cooperating MSA Service

### Comments_or_Changes

This service has been combined with the original item on low priority and expanded into into Priority delivery.

### Revised_Service_Statement

Superseded.

## Alarms (SF47)

Compliance Class: Future — Posting MSA Service

### Comments_or_Changes

This service is redundant with Resource Warning.

### Revised_Service_Statement

A "resource warning" is sent from the MSA to the UA when there is a resource shortage in the MSA.

Note: This service is also classified as Recipient MSA.

## Origintor_Authentication (SF87)

Compliance Class: Basic Posting MSA Service

### Comments_or_Changes

Formerly called Authentication. No specification in original.

### Revised_Service_Statement

This service is used to authenticate the UA which sent a message. The posting UA includes the information pertaining to the posting UA on the envelope.

## Charge_Advice (SF71)

Compliance Class: Future Posting MSA Service

### Comments_or_Changes

This service is redundant with Charging Information.

### Revised_Service_Statement

The charge advice service gives a user the possibility to request information about the tariffs of the various usage possibilities of the message handling service.

Note: This service is also classified as Recipient MSA.

## Address_Verification (None)

Compliance Class: Basic Posting MSA Service

### Comments_or_Changes

Not in the original document.

### Revised_Service_Statement

Check that requested address is real.

## Distribution_Lists (SF29)

Compliance Class: Basic Posting MSA Service

### Comments_or_Changes

Not in the original document.

### Revised_Service_Statement

This service is the same as Distribution Lists for Cooperating MSA, but is provided by the local MSA rather than the MTS.

## Resource_Status_and_Resource_Warning (SF38)

Compliance Class: Optional Posting MSA Service

### Comments_or_Changes

Not in original document.

### Revised_Service_Statement

The service is described in Resource Status and Resource Warning (Cooperating MSA service) and is the same except it is provided by the posting MSA.

## Historical_Recovery (SF61)

Compliance Class: Optional Posting MSA Service

### Comments_or_Changes

Not in original document.

### Revised_Service_Statement

This service is described in Historical Recovery (Delivery MSA service) and is identical except that it is provided by the posting MSA rather than the delivery MSA.

## Posting (SF14)

Compliance Class: Basic Posting MSA Service

### Comments_or_Changes

Listed as a function rather than as a service in the CCITT document. We should fight this out or just use modified text that is below.

### Revised_Service_Statement

This service realizes the transfer of responsibility for a message from the originator's UA to the posting MSA through the posting slot.

## Delivery_Time_Stamp (SF89)

**Compliance Class: Basic** **Delivery MSA Service**

### Comments_or_Changes

Not in original document.

### Revised_Service_Statement

This sevice adds the date and time to the envelope at the start of the delivery protocol.

## Explicit_Content_Type_Conversion (SF76)

Compliance Class: Future Delivery MSA Service

### Comments_or_Changes

Revised to indicate that service applies during transfer, not specifically to delivery. Adds statement that recipient can request the service.

### Revised_Service_Statement

This service allows the originator to request that a conversion function be performed upon the message during its transfer through the MTS.

The function that provides the transformation may be invoked by a recipient.

## Historical_Recovery (SF61)

Compliance Class: Optional — Delivery MSA Service

### Comments_or_Changes

No changes.

### Revised_Service_Statement

Historical recovery is a service whereby each message is stored in the MTS for an agreed period of time after delivery to the recipient's UA. During this period of time, the recipient's UA may request that a copy of the message be retrieved from MTS storage and transferred to the recipient's UA. When this service is enabled by the recipient's UA, it applies to all messages delivered to that UA.

The intent is to permit the UA to recover previously delivered messages if these messages have been lost due to catastrophic storage failure in the UA. Some UAs will possess their own backup facilities and, therefore, will not require this service.

## Delivery_on_Request (SF18)

Compliance Class: Required — Delivery MSA Service

### Comments_or_Changes

Formerly Hold For Delivery. Clarifies that either UA or MSA can initiate normal delivery.

### Revised_Service_Statement

This service is the default mode for message delivery. The delivery MSA holds messages within the MTS until the recipient's UA asks for them to be delivered. Although the UA is the entity that asks for delivery to occur, either the MSA or the UA may initiate the transaction. This service allows, for example, reduction of the recipient's communication costs by "batching" messages. Although a default exists, the recipient may specify the length of time for which such messages are to be held before they are considered undeliverable. (This service underscores the need for careful specification of the criteria for non-delivery notification.)

This service assumes a storage service.

## Implicit_Content_Type_Conversion (SF77)

Compliance Class: Future Delivery MSA Service

### Comments_or_Changes

Rewording only.

### Revised_Service_Statement

This service can be associated with a delivery service.

The most appropriate conversion function is executed implicitly when incompatibility between the content type and the recipient type is detected during delivery.

## Specific_Conversions_Prohibited (SF79)

Compliance Class: Future Delivery MSA Service

### Comments_or_Changes

This cannot be a service. An originator can only suggest, not control what a recipient UA should do to a message.

### Revised_Service_Statement

## Delivery_on_Condition — None

Compliance Class: Optional — Delivery MSA Service

### Comments_or_Changes

Not in original document.

### Revised_Service_Statement

This service allows the recipient's UA to indicate conditions under which the MSA may deliver mail to it.

## Posting_Validation — None

Compliance Class: Optional — Delivery MSA Service

### Comments_or_Changes

Not in original document.

### Revised_Service_Statement

Insure that the service(s) requested during posting can be supplied by the MTS.

## Notification_of_Messages_waiting_for_Delivery — None

Compliance Class: Basic — Delivery MSA Service

### Comments_or_Changes

Not in original document.

### Revised_Service_Statement

This service provides the Delivery MSA with the means of notifying the Recipient UA that one or more messages are in the

MSA awaiting delivery. This may include passing the number of messages waiting to the UA as well as other information.

## Recipient-Requested_Redirection — None

Compliance Class: Optional — Delivery MSA Service

### Comments_or_Changes

Not in original document.

### Revised_Service_Statement

This service is identical to Recipient-Requested Redirection (Cooperating MSA service), but is provided in the delivery MSA rather than in the MTS.

## Resource_Status_and_Resource_Warning — None

Compliance Class: Optional — Delivery MSA Service

### Comments_or_Changes

Not in original document.

### Revised_Service_Statement

The service is described in Resource Status and Resource Warning (Cooperating MSA service) and is the same except it is provided by the delivery MSA.

## Message_Intercept

(SF48)

Compliance Class: Optional

Delivery MSA Service

### Comments_or_Changes

### Revised_Service_Statement

A delivery MSA may be instructed to intercept any message that it is responsible for delivering and generate a predetermined response. This is an MSA service, and might be used when a UA is going to be off-line for an extended period. The incoming message could be either saved until intercept is disabled or discarded.

BOLT

Report No. ICST/CBOS-82-1

NOV. 1981

DEPARTMENT OF COMMERCE

NATIONAL BUREAU OF STANDARDS

Institute for Computer Sciences and Technology
Center for Computer Systems Engineering
Systems and Network Architecture Division

DRAFT REPORT

# Features of a Message Transfer Protocol

November, 1981

Prepared by:
Bolt Beranek and Newman,
Incorporated
10 Moulton Street
Cambridge, MA 02238

Prepared for:
Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, Maryland

## Foreword By the National Bureau of Standards

Protocol, interface and formatting standards are being produced by the National Bureau of Standards (NBS) for global, national and international networks, for local area data networks, and to support computer based office systems applications. Guidelines on the use and selection of supporting technologies are also being produced by this NBS networking program. The objective of this program is to facilitate the interconnection and integration of computer systems and devices through networking technology to support distributed processing.

This draft report is one of a series of draft reports being prepared under the computer based office systems segment of the networking program for distribution to Government agencies, manufacturers, and other interested parties. We urge readers to provide comments and to further interact with us on this report. Written comments should address both the advantages and disadvantages (from the reader's viewpoint) of individual features described in this report. Responses should be directed to the address below, and NOT to the NBS contractor that prepared the report.

Reply to:

National Bureau of Standards (Code CBOS-82-1)
Systems and Network Architecture Division
Technology Building, Room B218
Washington, DC 20234

## TABLE OF CONTENTS

## 1. INTRODUCTION

This report is the first step in the development of protocols for exchanging messages between Computer Based Message Systems (CBMSs). This work is part of a program sponsored by the U.S. National Bureau of Standards to develop a family of computer network protocols [4].

This report is based primarily upon the work of two groups, CCITT Study Group VII, Question 5 (Message Handling Facilities) and IFIP TC6 Working Group 6.5 (International Computer Message Services). Additional input was drawn from the CBMS literature and CBMS vendors. Like many other computer and communications technologies, CBMS technology is continuing to evolve. The descriptions in this report, however, represent the state of the art.

### 1.1 Feature Analysis

A protocol consists of a kernel set of features and of clusters of value-added features. The kernel set provides the essential features of a protocol; value-added features are non-essential enhancements to the kernel protocol.

A feature analysis classifies the features of a protocol into a kernel set and value-added clusters. The criteria for selecting the kernel and ranking the value-added features include relative cost-effectiveness, level of demand, implementation expense, elegance, and sufficiency. As a result of applying feature analysis to CBMS protocols, we will have identified both the kernel set of features that must be integrated into a protocol and several levels of value-added functionality. The value-added levels are integrated with the kernel to form a protocol family answering the needs of disparate applications. In addition, the rationale used in the analysis will be applicable in the later step of specifying the protocol itself.

## 1.2 A Model of a CBMS Environment

In order to provide a framework for discussing a message transfer protocol, this section describes a simple functional model for a CBMS. The model provides a high-level description of both user facilities and system architecture.

A CBMS permits the transfer of a message from an originator to a recipient. "Originator" and "recipient" are used in their normal English senses. A message (in its most abstract definition) is simply a unit of communication from an originator to a recipient. A CBMS offers several classes of functions to its users:

- o Message Creation: The facilities used by a message originator to create messages and specify to whom they are to be sent.
- o Message Transfer: The facilities used to convey a message to its recipient(s).
- o Recipient Processing: The facilities used by a message recipient to process messages that have arrived.

A message transfer protocol governs how a message is moved from its originator's CBMS to its recipients' CBMSs. It does not say anything about how messages are created or what happens to them after transfer has been completed.

CBMS facilities for message creation, transfer, and recipient processing are reflected in a logical model of a CBMS developed by IFIP Working Group 6.5 [6]. (An essentially identical model is being used by CCITT Study Group VII, Question 5, regarding Message Handling Facilities.) The model consists of a Message Transfer System and a number of User Agents. (See Figure 1.)

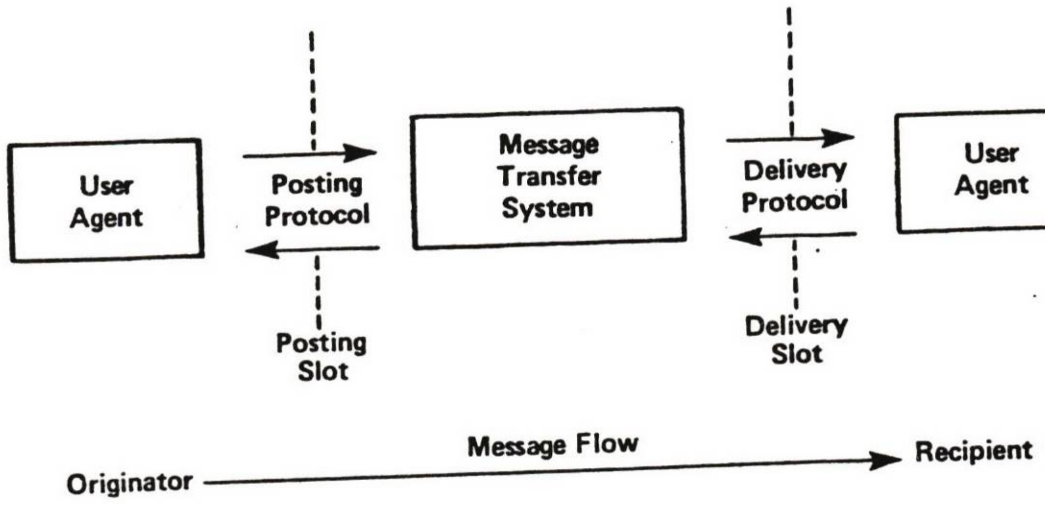

FIG. 1. LOGICAL MODEL OF A COMPUTER BASED MESSAGE SYSTEM

A <u>User Agent</u> is a functional entity that acts on behalf of a user, assisting with creating and processing messages and communicating with the Message Transfer System.

The <u>Message Transfer System</u> is an entity that accepts a message from its originator's User Agent and ultimately passes it to each of its recipients' User Agents. The Message Transfer System may perform routing and storage functions (among others) in order to accomplish its task.

Transferring a message from an originator's User Agent to the Message Transfer System is called Posting; the originator's User Agent and Message Transfer System engage in a Posting Protocol in order to accomplish Posting. Transferring a message from the Message Transfer System to a recipient's User Agent is called Delivery; the recipient's User Agent and Message Transfer System engage in a Delivery Protocol in order to accomplish Delivery.

The point at which responsibility for a message is transferred is called a Slot. The Posting Slot is the point at which responsibility for a message passes from an originator's User Agent to the Message Transfer System; the Delivery Slot is the point at which responsibility for a message passes from the Message Transfer System to a recipient's User Agent.

The Message Transfer System is composed of one or more Message Transfer Agents. Message Transfer Agents cooperate to provide the services of the Message Transfer System. A Message Transfer Agent accepts a message from a User Agent or another Message Transfer Agent. Then the Message Transfer Agent passes the message along to its destination User Agent or to some other Message Transfer Agent which is closer to the destination User Agent. This means that a Message Transfer Agent must be able to buffer entire messages. A Message Transfer Agent also must be able to determine where each message should be passed next. The use of Message Transfer Agents as intermediaries between User Agents makes it possible for a message to be transferred between two User Agents regardless of their ability to communicate directly. The originating User Agent does not need to know anything about the availability of the receiving User Agent. In fact, the originating and receiving User Agents do not have to be available simultaneously at all.

The model divides messages into two parts, the message content and the message envelope. The message content is the information that the originator wishes to send to the recipient. The format of the message content has been specified in [5]. The message envelope consists of all the information necessary for the Message Transfer System to do its job; the format of a message envelope will be specified as part of a message transfer protocol. Some of the data appearing on the message envelope could be redundant with some data found in the message content. The Message Transfer System is not expected to examine the message content unless it is told to do so by the originator's or recipient's User Agent.

## 1.3 Placement of Message Transfer Protocols in the ISO Referemce Model

Message transfer protocols exist above the session layer of the ISO reference model for Open Systems Architecture (see figure 2.) Because a CBMS is an application program, the message transfer protocol is primarily an application layer protocol. It must also extend, however, to deal with some presentation issues.

CBMS protocols can be divided into two sublayers, the message processing layer and the message transfer layer (illustrated in figure 2). The message processing layer is the upper sublayer. It is concerned with the processing that UAs perform on message contents. The originator's and recipient's

UAs are the peer entities communicating at this layer. The CBMS user, who is not part of the protocol, is above the message processing layer.

The message transfer layer is the lower of the two sublayers. It is concerned with the work that must be performed in order to move a message from an originating UA to a destination UA. There are two kinds of entities in the message transfer layer, Message Transfer Service Access entities (MTSAEs) and MTAs. An MTSAE gives a UA access to the services of the MTS. An MTSAE and its UA are always located in the same system. An MTSAE and an MTA can be in the same system or in different systems.

The message transfer layer is above a session layer protocol, which might be the same as one used by Teletex, for example. An originator's MTSAE and a posting MTA are peer entities that communicate at this sublayer; so are two MTAs, or, a delivering MTA and a recipient's MTSAE. This report describes features of a protocol at this level.

### 1.3.1 Role of the presentation layer

The entities at the message handling sublayer communicate asynchronously. Their communication channel is formed by the sessions held between the entities at the message transfer layer. This makes it difficult to separate presentation from the message transfer protocols. At this time there is no consensus about the role of the presentation layer in message transfer protocols. The following paragraphs propose an approach to that issue.

Normally, a presentation layer protocol is used to negotiate

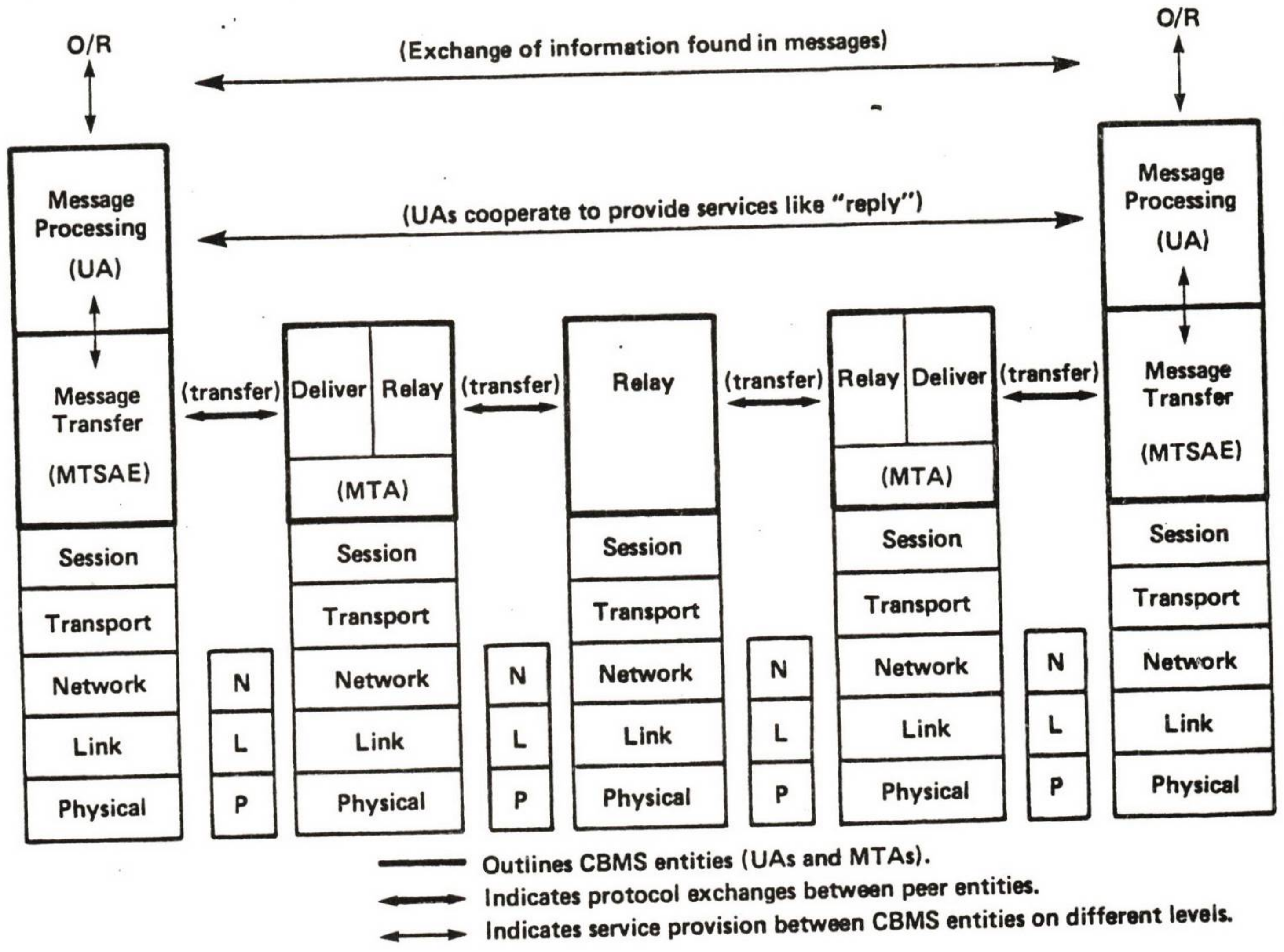
O/R
(Exchange of information found in messages)
O/R
Message Processing (UA)
(UAs cooperate to provide services like "reply")
Message Processing (UA)
Message Transfer (MTSAE)
(transfer)
Deliver
Relay
(MTA)
(transfer)
Relay
(transfer)
Relay
Deliver
(MTA)
(transfer)
Message Transfer (MTSAE)
Session
Transport
Network
Link
Physical
N
L
P
Outlines CBMS entities (UAs and MTAs).
Indicates protocol exchanges between peer entities.
Indicates service provision between CBMS entities on different levels.

FIG. 2. MESSAGE TRANSFER PROTOCOLS IN THE ISO REFERENCE MODEL

the form and representation of data being exchanged between application entities. This is done by presentation entities that are part of the sending and receiving systems during a session between the two systems. In the case of CBMS, however, the sending and receiving systems do not communicate directly. A presentation entity in the originator's UA cannot negotiate directly with a similar entity in the recipient's UA.

Presentation entities can exist in CBMSs during sessions between entities at the message transfer sublayer. They would negotiate the form and representation of the message envelope only. They would do nothing with the message content. This is because the message content data belongs to a layer above the message transfer sublayer and must be passed unchanged, according to the definition of a layered protocol. The message envelope on the other hand is the responsibility of the message transfer sublayer, and the presentation entities can properly negotiate about it.

## 1.3.2 Role of the Transport Layer

CBMS entities are applications, and therefore use application layer protocols to communicate with each other. However, some similarities between transport (layer 4 of the ISO model) and message transfer have been pointed out. The following paragraphs contrast the two layers.

The transport layer [1] supplies reliable end-to-end movement of data between two hosts. The transport layer performs error detection and flow control (among other things) to do this. The transport entities at the sending and receiving hosts communicate with each other by using the services of the network layer. The data exchanged by the transport entities may pass through many intermediaries in the network layer.

The transport layer offers both connection-oriented and connectionless data transmission. It may offer a data broadcast service, which allows a sending transport-user to request that a single piece of data be transmitted to more than one receiving transport-users.

At first glance it may seem that the transport layer supplies exactly the services required for message transfer. The two end-hosts may have one or more intermediary systems between them. The data transmission is reliable, and it is possible to broadcast one piece of data to more than one users. However, the transport layer is not sufficient for the purposes of message transfer.

There are a number of important differences between data transport and message transfer services, including addressing, time considerations, and data unit size. In addition, message transfer services such as user directories or format conversion also are far more application-specific than might be found in a general-purpose transport layer.

Transport-layer addressing is not adequate to identify users of a message transfer service. A transport address offers identification only down to the host level. Session addresses identify the processes which use the transport layer. Message transfer addresses must be at least as granular as session addresses. A host designation is adequate to identify a message transfer-user only when there is only one user on a host and it is a message transfer-user. This is the case only in dedicated, single-user systems.

Whether it is connection-oriented or connectionless, the transport layer is intended to provide speedy, real-time data transmission. Message transfer is not always done in real-time. CBMS messages can be buffered in an MTA for hours or even days. The transport and network layers do not work on such a large time-scale. Their time-out intervals are much smaller than what is needed to provide message transfer.

While an average CBMS message is about a thousand octets long, CBMS messages may range from one hundred to many thousands of octets in length. The transport and network layers deal with data units which are smaller and more homogeneous in size. They cannot deal with CBMS messages as integral objects.

Given the above arguments and the application-specific services offered by a message transfer layer, it is apparent that the transport layer cannot by itself offer message transfer services.

## 1.4 Distinction between Service and Protocol Features

This report discusses the features that a message transfer protocol provides to its users; these are called service features. This report also discusses the additional features that the message transfer protocol must provide for its own use; these are called protocol features. The difference between service and protocol features is discussed in greater depth in [1].

## 2. FEATURES OF A MESSAGE TRANSFER SYSTEM

### 2.1 Service Features

Service features are the features that a protocol provides for its users. This section describes the service features of a message transfer protocol in various classes.

One fundamental class of service is message transfer. Message transfer is the operation that "moves" a message from an originator's UA to a recipient's UA. As a result, the recipient's UA gains access to the message and becomes responsible for its storage and integrity. Message transfer takes place in three phases, posting, relay, and delivery. Posting is the entry of a message into the Message Transfer System by an originator's UA. Relay is the movement of a message from one part of the Message Transfer System to another. Delivery is the exit of a message from the Message Transfer System to a recipient's User Agent.

Posting and delivery are service features and are treated below. Relay cannot be considered a service feature because it is not directly requested or seen by the UAs.

#### 2.1.1 Message Posting

Posting is the transfer of responsibility for a message from an originator's UA to an MTA. The UA gives the MTA a copy of the message content and instructions about what to do with the message content. These instructions include the destination (or destinations) of the message, handling for the message en route, and options for its delivery. As part of posting, the MTA verifies, to the extent possible, that these instructions can be carried out. When the MTA acknowledges to the UA that it has accepted the message, it can also give the UA some identifying information that the UA can use in future references to the MTA about this particular posting transaction.

As part of the instructions given during posting, the originating UA can indicate what kind of notification it should receive about the delivery of the message. Two options are nondelivery notification, if delivery fails for some reason, and return-receipt following successful delivery.

### 2.1.2 Message Delivery

Delivery is the transfer of responsibility for a message from an MTA to a recipient's UA. The MTA gives the UA a copy of the message content; it might also give the UA all or part of the message envelope. Several major sets of options affect message delivery.

- o What initiates delivery. Either the delivering MTA or the recipient's UA can initiate delivery of a message. This option is under control of the recipient UA and the delivery MTA; it is not specified by the originator.
- o When delivery may occur. The originator can specify a time before which delivery should not take place; the originator can also specify a time after which delivery should not take place. (Default values for these options are to deliver as soon as the message is in the possession of the delivering MTA and not to deliver after a standard timeout interval.) Apart from that, the originator can specify a priority at which the message is handled by the MTS. Finally, the originator can request that a message be delivered more than once, at stated times or at some regular interval.

- Conditions for delivery. A third set of delivery options concerns other conditions, not related to time, that must be met for delivery to take place. An originator might be able to specify that delivery not take place if the recipient's UA cannot handle the format of the message content, for example.

- Actions on failure. A final set of options concerns the action that the originating UA wants the MTS to take should message delivery fail. The UA can request that a copy of the message be returned, that some alternate recipients get the message instead, or that it not be notified at all.

### 2.1.3 Delivery Cancellation

Delivery cancellation is a service that can be requested by the UA that has posted a particular message. The service aborts delivery of the message to its recipient's UA. If the message has already been delivered, the request cannot be filled. In addition, it might not always be possible to cancel the delivery of a message, even if it had not been delivered at the time the cancellation request was made. This is because the cancellation request might not catch up to the message in time.

### 2.1.4 Message Trace

Message trace is a service which locates a message in the MTS. It may be used to find the location of a message which has been posted but has not been delivered. Message trace may also be used by the providers of the MTS for diagnostic purposes. Whether or not subscribers to the MTS can use message trace is a decision of the MTS provider.

### 2.1.5 Subscriber Directory

Subscriber directory services help identify and locate CBMS users. CBMSs differ from other forms of electronic communications in that message originators and recipients can be identified by name instead of (or in addition to) address. The name of a message recipient does not necessarily indicate where a message for that individual should be sent. That information, the address, is determined by looking up the name in a directory.

There are two kinds of subscriber directory services. Directory assistance is used to look up the names and addresses of CBMS subscribers. Subscriber registration is used to register new CBMS users and to update directory information about old CBMS users.

Directory assistance is used by UAs and the MTS itself. Two forms of directory assistance might be available.

- Look-up. The entity requesting the service supplies a name. If the name is an unambiguous match to a name in the directory, then the address associated with the name is returned. Additional information associated with the name can also be returned. Look-up can fail for several reasons: the name matched more than one entry in the directory, no such name, name in wrong format. In the first case, the list of possible matches might be returned; in the other cases there is simply an error indication returned.

- Confirmation. The entity requesting the service supplies a name or an address. If the information supplied was valid, there is a positive response, otherwise there is a negative response.

The directory service may be used for a number of purposes. Users may invoke it as part of message composition, in order to verify that they can send the message to its intended recipient.

The directory service may be called during posting, in order to insure that the destination information supplied by the UA is sufficient and valid. In that case, if there is some problem, the UA knows about it immediately. Finally, the directory service can be used after posting. This is done when an MTA has in its possession a message which has a name for its destination but no address.

The subscriber directory may be used by MTS providers as a partial basis for billing.

Subscriber registration is used to update the CBMS user directory. Updates can be entered by the directory's provider. Another method is to allow CBMS subscribers to change the contents of the directory. In that case, there may be limitations placed upon the subscribers' ability to alter the directory. For example, update access to the directory may be limited to the subscriber's own directory.

Directory implementations are not constrained by the functional model. A directory may be resident in the same host as a UA, the same host as an MTA, or in a host having neither a UA nor an MTA. Directories may be distributed across several hosts; they may also be redundant.

### 2.1.6 Interworking

Incompatibilities between UAs can arise on two levels, message content format and the media found in the message content. Interworking services are used to overcome these incompatibilities.

An international message transfer system will be used to move many different kinds of messages. For example, it might be used for CBMS messages and Teletex messages. Even on a national level, private formats might be used between two consenting systems. One type of interworking service transforms a message from one format to another.

A variety of types of data can be found in CBMS messages, including text and facsimile. The representation of data for these media might be specified in a single format standard, but it cannot be expected that all UAs would be able to deal with all of them. Another kind of interworking service transforms data for one medium into a representation suitable for another (for example, text to facsimile). When a transformed message is delivered to its recipient's UA, the UA is told what kind of transformation was done.

Interworking services are never performed unless they have been requested by either the message originator or recipient. A message originator might also request that a message not undergo a format or medium transformation. If delivery fails because of an incompatibility in message format or medium, this information is included in the non-delivery notification.

### 2.1.7 Permanent Storage

At the request of a message recipient's UA, a message transfer system can permanently store a copy of an incoming message. This service is used to provide backup or archival storage beyond the usual storage facilities that are part of the recipient's UA.

### 2.1.8 Message History

Message history is a record of what has happened to a message while it is in the message transfer system. Message history is used by UAs to determine message authenticity and integrity. It also serves as a basis for customer billing and settling accounts between the administrations that provide the MTS. The MTS could use message history to analyze traffic flow and other usage patterns.

In the context of a CBMS, authentication is the positive identification of the entity that posted a message. Integrity refers to whether or not unwanted changes have occurred in the message content as it is moved through the MTS.

In some situations, a message recipient might require some assurance that the message came from a particular originator. The MTS can authenticate that a given message was posted by a particular UA. There are two reasons why it may not be able to authenticate that the UA was representing some particular user when it sent the message. First, while the MTS is "trusted software", a UA is usually not trusted. A UA cannot always be relied upon to provide factual information to the MTS. Secondly, a computer can be fooled by a person who has acquired someone else's password.

The MTS may be able to tell the recipient's UA what path a message took as it moved through the MTS, and what processing was done upon the message along the way. This information determines the degree of trust that the recipient can have in the integrity of the message.

### 2.1.9 Message Transfer System Status Information

A status information service is used to tell UAs about the operation of the MTS. UAs query the MTS for information. In emergency situations, the MTS can initiate the service. MTS-initiated status reports, called alarms, are used to warn UAs about resource shortages or MTS failures that could seriously affect (or even curtail) the operation of the MTS.

A UA can query the MTS about its operating status. It can also ask the MTS for other information, such as tariffs. This allows the UA and perhaps ultimately the CBMS user to make informed decisions about the best way to use the services provided by the MTS.

## 2.2 Protocol Features

Protocol features are features of the protocol that the message transfer protocol must provide for its own use. This section describes various classes of protocol features.

### 2.2.1 Negotiation Mechanism

A negotiation mechanism is used by message transfer protocol participants when there is no prior agreement on which features they will use and how they will use them. Two items on which message transfer protocol participants must agree are use of parameters to the protocol and use of features not in the protocol kernel.

Although all of the kernel features must be provided in every implementation, it may be necessary to negotiate their parameters. Error recovery is an example of this. While the use of recovery units is in the kernel of a message transfer protocol, the size of the window (the number of recovery units that may be transmitted without a positive acknowledgment being received) is negotiable.

The use of value-added features is also subject for negotiation between peer message transfer protocol entities. Both peers must implement a value-added feature in order for it to be used. In addition, there may also be a need to negotiate parameters for the feature.

### 2.2.2 Error recovery

High-level error recovery features provide for orderly recovery from errors not normally detectable by lower-level protocol entities. File system and operating system errors are examples of this kind of error.

One common way to do error recovery is to insert checkpoints in the data stream. The data between two checkpoints is called a _recovery unit_. Receiving processes acknowledge the receipt of recovery units. If a recovery unit is negatively acknowledged, transmission is "rolled back" to the beginning of that unit. .

In order to provide for continuous data flow, more than one recovery unit may be transmitted before an acknowledgment is received by the transmitter. A _window_ is the number of recovery units which may be transmitted before the sender receives an acknowledgment. The size of the window is subject to negotiation by the sending and receiving entities.

The session layer does not provide checkpointing or a similar error recovery mechanism. Therefore, it must be provided within the message transfer protocol.

### 2.2.3 Timestamping

A timestamp is used to indicate when a given operation or transaction took place. The MTS is expected to provide timestamps at posting, relay, and delivery, and whenever an interworking service is performed on a message in the MTS.

Message delivery uses the posting timestamp in order to determine if a message has timed out. Timestamps are also used in estimating a message's integrity.

### 2.2.4 Unique Identifiers

Unique identifiers provide a means of referring to messages. The MTS must be able to generate unique identifiers for its own use. The unique identifiers passed between a UA and an MTSAE may be the same as those passed between the MTSAE and an MTA or they may be different. If they are different (presumably to simplify the operation of the UA), they have only local significance. In that case, the MTSAE must be able to maintain the correspondence between the local identifiers and the identifiers used by the MTS.

### 2.2.5 MTS Addressing

An MTS address specifies a location relative to the MTS. It is distinct from a name, which identifies a particular entity but does not say where that entity is located. A name can be mapped into an address. The mapping of a name into an address may change with time.

The MTS uses addresses to identify where messages should be routed for delivery to a UA. It also uses addresses to locate other CBMS entities relative to the MTS.

### 2.2.6 Message Routing

Given an initial location and the address of a CBMS entity, the MTS must be able to determine a route between those two points. A route is a series of MTAs through which a message passes along its way to the destination UA. Routing can be performed incrementally, or the entire route can be determined in a single operation.

Routing in the message transfer layer is analogous to but not the same as the routing that is done in the transport and network layers. MTAs are application entities which perform application-specific functions beyond the simple transfer of messages; they also can buffer entire messages for hours or days. The transport and network layers are concerned with general data transmission and are designed to work in real-time.

### 2.2.7 Message Buffering and Storage

Each MTA must be able to buffer the messages in its possession. Typically, a message would reside in an MTA for a period of time measured in seconds or minutes. The MTA would store such a message internally, for fast and easy access. In cases such as timed delivery, however, an MTA might have to store a message for longer periods. In this case, other storage can be used.

### 2.2.8 Accounting

The MTS may perform accounting functions in order to enable customer billing and settlement between different CBMS administrations.

### 2.2.9 Acknowledgments

The MTS provides acknowledgments in order to signal the success or failure of various operations. There are two kinds of acknowledgments. Both are used by the MTS. The first kind is used within the MTS between two MTAs. The second kind is used between a UA and an entity in the MTS.

Most operations, such as message relay, require a positive acknowledgment before they can be assumed completed. Sometimes an operation is assumed to have succeeded unless a negative acknowledgment is received. This is the case for a UA that has posted a message without requesting that a return-receipt be generated upon its delivery.

### 2.2.10 Security

The MTS must provide protection against unauthorized use of its services. In order to do this, it must be able to positively identify UAs. This might be done by the use of passwords, permanent connections believed to be secure from tampering, or some other method. In addition, the MTS may implement a security policy which segregates information, routes messages through secure MTAs, etc.

## 3. FEATURES ANALYSIS

Protocols are often defined in two parts. The kernel is that part of a protocol that is required in every implementation. The value-added portions of a protocol are optional to implement. The kernel must be sufficient and complete. That is, it must provide all of the essential services of the protocol and it cannot rely on any value-added features for correct operation. The value-added features can be regarded as enhancements. While they need not be implemented, if they are implemented, they must operate as defined in the protocol.

This section divides the features of a message transfer protocol into a kernel and a value-added portion. The kernel consists of a basic form of most of the features. The value-added portion consists of enhancements to the features in the kernel plus any additional features. The decision to put a particular feature in the kernel or treat it as an enhancement was based on two main criteria.

1. Is the feature required for proper operation of the protocol? If the feature were not universally implemented, would the protocol be unable to provide reliable message transfer? Required features were placed in the kernel.
2. Is the feature now widely implemented? Features without wide acceptance were placed in the value-added set. Features that are of limited interest, because of their cost or purpose, should not be required in all systems.

Other criteria entering the decision included cost of implementation and efficiency. Features that are not required and not widely implemented were classified as value-added enhancements.

The following sections discuss each of the features described in the previous chapter. Within each feature class, the features are classified as kernel or as value-added parts of the protocols. The rationale for the classification is presented.

## 3.1 Service Features

### 3.1.1 Message Posting

Kernel:

An originating UA can request that a message be delivered to a single receiving UA. Should delivery of the message fail, the originating UA receives notice of nondelivery.

Note: This does not refer to the ability of a user to specify to the UA that a message content be sent to more than one recipient; instead it refers to the ability of the UA to specify to the MTS that a single message content be delivered to a list of recipients. The ability of a UA to specify that a single message be delivered to a list of recipients is called "multiple addressing".

Value-added features:

1. The originating UA can specify more than one destination (multiple addressing).
2. Message transfer can occur at any of several levels of priority.
3. Originating UAs can specify a time before which delivery may not take place and a time after which delivery may not take place.

4. Originating UAs can specify a time interval for repeated delivery of the same message.

5. Originating UAs can specify a list of times at which a message is to be repeatedly delivered.

6. Originating UAs can specify that the message content be transformed into another format.

7. Originating UAs can specify that a message not be delivered unless its content can be interpreted by its recipient in its original form.

8. Originating UAs can request that they be notified on successful delivery of a message.

9. Originating UAs can request that they not be notified when delivery fails.

10. Originating UAs can request that a copy of the failed message be returned when delivery fails.

11. Originating UAs can specify one or more alternate recipients to whom delivery should be attempted if delivery to a given recipient fails.

Rationale:

The simplest useful posting service would allow a single message to be posted to a single recipient. Delivery would be said to fail if it could not be successfully completed in some default time interval, on the order of a week. Should delivery fail, the originating UA would be notified. This is the most commonly used type of posting service. The only way to simplify it or make it less expensive to provide would be to eliminate the nondelivery notification. However, that seems unwise, as a nondelivery notification is extremely desirable in most cases.

The value-added services are not widely implemented, though they are felt to be desirable by CBMS vendors and users. Of the entire list, only return-receipt upon delivery and multiple addressing are found in more than a few systems.

### 3.1.2 Message Delivery

Kernel:

An MTA can transfer a message content to a receiving UA. The service is initiated by the MTA.

Value-added features:

1. An MTA can transfer both the message content and envelope to a receiving UA.
2. A receiving UA can request the transfer of a message to it from an MTA.
3. A receiving UA can ask an MTA if the MTA has a message awaiting transfer to the UA.

Rationale:

Logically, message delivery can be initiated either by a delivering MTA or by a receiving UA. In the first case, the MTA signals the UA that a message is awaiting delivery. In the other method, the UA polls the MTA to check for messages awaiting delivery. In most current systems the MTA initiates message delivery.

### 3.1.3 Delivery Cancellation

Kernel:

No delivery cancellation features in the kernel.

Value-added features:

The originating UA can request that the delivery of a specific message be cancelled. If the message has already been delivered, the request will fail. If the request does not "catch up" with the message before the message is delivered, the request will fail.

Rationale:

Implementin~ a delivery cancellation service would require the ability to transmit the cancellation request at a high priority in order to catch up with the message to be cancelled. Because it is not always possible to cancel delivery of a message, and because providing that service is likely to be expensive, delivery cancellation is likely to be a little-used feature.

### 3.1.4 Message Trace

Kernel:

No message trace feature in the kernel.

Value-added features:

An authorized entity may request the location of a message which has already been posted. The MTS searches for the message. If the message is found in the MTS, the location of the message and when it was found is returned. If the message is no longer in the MTS, the time and location where the delivery took place (or where the message was found to be undeliverable) are returned. The request fails if the MTS has no record of the message.

Rationale:

Message trace is not necessary for the operation of a message transfer system. Most current systems do not support this feature.

### 3.1.5 Subscriber Directory

Kernel:

There is a directory listing all registered CBMS subscribers, either on-line or on some physical medium (such as paper).

Value-added features:

1. Either a UA or an MTA can request the address associated with a given user name. If the name can be unambiguously matched to a name in the user directory, the address associated with that name is returned.

2. Either a UA or an MTA can request the address associated with a given user name. If the name can be unambiguously matched to a name in the user directory, the address associated with that name is returned. If there is no unambiguous match, a list of all candidate names along with their addresses is returned.

3. Either a UA or an MTA can request verification that a given address is associated with a given user name. If the address is associated with the name, a positive acknowledgment is returned, otherwise a negative acknowledgment is returned.

4. UAs can request that the subscriber directory be updated.

Rationale:

Some directory listing CBMS subscribers must be available. The directory might be partitioned according to the administrations or organizations that provide the MTS service.

While new and planned CBMSs make use of a directory assistance service, most current CBMSs do not. Should it become necessary to provide some directory assistance feature in the kernel, the first value-added feature above would be appropriate.

On-line update and registration are implemented by very few current CBMSs. The provision of such services in an open, public environment requires further study.

### 3.1.6 Interworking

Kernel:

A receiving UA can reject the delivery of messages expressed in a format or containing data that it cannot handle.

Value-added features:

1. All or part of a message content can be transformed from one specified format to another. This service can be invoked by any UA.

2. All or part of a message content can be transformed from one specified format to another. The message is marked as having undergone this transformation. This service is invoked by an MTA when it discovers that the message content, in its original form, cannot be handled by the receiving UA.

3. An originating UA can determine a receiving UA's capabilities for handling different message content formats and media.

4. A receiving UA can reject the delivery of a message whose message content has undergone a format transformation.

Rationale:

A receiving UA must be able to reject messages that it cannot interpret. This makes its operation simpler and divests it of responsibility for properly processing messages in formats it is not expected to handle.

Services that allow UAs to explicitly request a transformation on a message content are useful in environments where private formats may be used or where more than one format standard exists. Whenever the decision to request the service is made by an MTA, this information must be available to the receiving UA. Because format transformations involve a potential loss of data, a UA might need to reject messages that have undergone format transformations.

### 3.1.7 Permanent Storage

Kernel:

No permanent storage feature in the kernel.

Value-added features:

A receiving UA can request that a given message be stored permanently. The request is made during delivery of the message. The UA can request retrieval of the message at a later date. The retrieval operation need not occur immediately; a substantial delay is permitted.

Rationale:

It is not absolutely necessary to have this feature in a message transfer protocol. However, some UAs might lack significant amounts of long-term storage and therefore such a service might be useful. Since a stored message may be off-line (on an unmounted tape, for example), the retrieval of such a message might take some time to complete.

### 3.1.8 Message History

Kernel:

The MTS maintains a record of each operation on a message in the MTS. The record identifies the operation, the entity that performed it, and when it took place. At delivery time, a receiving UA can determine where a message originated. The timestamps for each of these events are included.

Value-added features:

1. At delivery time, a receiving UA can determine the path a message took across the MTS, and what operations were performed upon it en route.

2. Authorized entities can verify that an originating UA posted a specific message to a specific recipient at a specific time. This service will be available only to the originating UA, the receiving UA, or other authorized entity.

3. Authorized entities can verify that a specific message was delivered to a specific receiving UA at a specific time. This service shall be available only to the originating UA, the receiving UA, or other authorized entity.

Rationale:

The MTS must maintain records of what happens to the messages in its possession. This is necessary to provide authentication and integrity services. It is also needed for investigating any problems that occur in the operation of the MTS. Receiving UAs sometimes require reliable information about the origin of a message and what happened to it while it was in the MTS. These services are in the kernel because of their great usefulness.

In addition, it is sometimes necessary that an originating UA be able to prove that it posted a given message. Originating UAs might need to prove that a given message was delivered to a receiving UA. These services are not widely implemented.

### 3.1.9 Message Transfer System Status Information

Kernel:

No message transfer system status information feature in the kernel.

Value-added features:

1. A UA can request the price that would be charged for delivering a given message to a given receiving UA.
2. A UA can request the price that would be charged for any given service provided by the MTS.
3. A UA can request general tariff information for the services provided by the MTS.
4. A UA can request a report about the status of a specific component of the MTS.

5. A UA can request a general report about the status of the MTS.

6. An MTA can warn a UA about a condition or impending problem that is expected to affect the service provided by the MTS.

Rationale:

Few if any CBMSs support a message transfer system status information feature. Such features have been proposed, however, and they would undoubtedly be useful.

## 3.2 Protocol Features

### 3.2.1 Negotiation Mechanism

Kernel:

Simple true/false negotiation options are used.

Value-added Features:

Negotiable parameters are stated using qualifiers.

Rationale:

Most negotiations which occur in a message transfer protocol are to determine each peer entity's repertoire of value-added receiving capabilities. The simplest method for doing that is for the two entities to exchange lists of receiving capabilities. A sending entity never requests a capability which the receiving entity is known to lack.

Another kind of negotiation concerns the values of parameters used in either kernel or value-added features. These parameters could be negotiated by having the first entity propose a value and allowing the second entity accept or reject it. However, this method is rather inflexible and potentially costly to carry out. The Teletex [7] protocol allows the second entity to respond by either accepting the entire proposal, describing a subset of the proposal which it is willing to accept, describing its own value-added capabilities, or rejecting the entire proposal. The NBS File Transfer Protocol [2] allows for the use of qualifiers such as "greater than" or "equal" to be used in a negotiation.

In order to facilitate interworking between CBMSs and Teletex, the kernel protocol includes a Teletex-like negotiation mechanism. The use of qualifiers is included as a value-added option which is of less immediate use in the CBMS application.

### 3.2.2 **Error recovery**

Kernel:

Checkpoint and restart features are provided.

Value-added Features:

No error recovery value-added features.

Rationale:

High-level error recovery features provide for orderly recovery from errors not normally detectable by lower-level protocol entities. File system and operating system errors are examples of this kind of error. Interruption of the session over which data is being transmitted is another example. Error recovery is particularly important for message transfer because CBMS messages can be quite large.

Both Teletex [7] and FTP [2, 3], which are applications similar to CBMS, use checkpointing to implement error recovery. Checkpointing is a simple mechanism which is entirely adequate for the needs of message transfer.

Using an error recovery mechanism does have a cost in terms of implementation complexity. On the other hand, the size of CBMS messages practically dictates the availability of some error recovery mechanism. Since error recovery is so valuable in message transfer, it is included in the kernel protocol.

### 3.2.3 **Timestamping**

Kernel:

Timestamps will be used for marking events in the MTS. The universal date and time will be used. The resolution and accuracy of a timestamp must lie within some defined ranges.

Value-added features:

The estimated accuracy of a timestamp may be included with the timestamp.

Rationale:

Timestamps are needed to determine the ordering of events, when they took place, and how long they lasted. Timestamps are expected to be reasonably accurate, within some maximum error interval. In addition, timestamps must maintain some minimum resolution, probably one minute. In some cases it might be reasonable for the system that generates a timestamp to include with the timestamp an estimate of its accuracy.

### 3.2.4 Unique identifiers

Kernel:

MTAs must be able to generate identifiers which are unique within the domain of their usage.

Value-added features:

MTAs may be able to generate identifiers which are and will always be unique across all interconnected CBMSs.

Rationale:

MTAs use identifiers to unambiguously refer to messages and other objects. An MTA (or other CBMS entity) should never encounter a situation where two different objects both have identifiers with the same value. This means that at the very least, identifiers used in the MTS should be unique within their processing environment and life span.

Sometimes it is convenient to use an identifier which will always be unique, no matter where or when it is encountered. This is the case for objects which have an indefinite existence and can be referred to from any part of the interconnected CBMS environment. It is not strictly necessary to be able to generate such an identifier, but doing so can be a considerable convenience. Therefore, the ability to generate globally unique identifiers is a value-added feature.

### 3.2.5 MTS Addressing

Kernel:

The MTS address of an MTA is the concatenation of a network-wide host identifier with a locally-allocated identifier. The MTS address of a UA or other entity which is not an MTA is the concatenation of an MTA address with an additional identifier known to that MTA.

Value-added features:

A global network identifier may be prefixed to the MTS address to allow internetwork addressing.

Rationale:

For the purposes of communications, an MTA can be modeled as a process resident on a host. Using a concatenation of host and process identifiers to form an MTA's MTS address allows the address to be easily mapped into a session address. This is very useful when a UA or MTA wishes to establish communication with a given MTA.

Communications with UAs are always relative to the MTA to which the UA is connected. For example, all messages to be delivered to a given UA are transferred to the same "last" MTA, and that MTA does the delivering. In order to transfer a message to a UA through the MTS, it is necessary to know which is the "last" MTA. Addressing UAs by concatenating the MTS address of the "last" MTA with an identifier known by it makes this easy to know. When that MTA gets the message, it uses the additional identifier in order to determine how to contact the UA.

The use of network identifiers in forming MTS addresses is value-added because most current CBMSs operate in single-network environments.

### 3.2.6 Message Routing

Kernel:

An MTA shall be able to determine the next step in the route a message must take in order to cross the MTS and be delivered to its recipient's UA.

Value-added features:

An MTA shall be able to determine the complete route a message must take in order to cross the MTS and be delivered to its recipient's UA.

Rationale:

At the very least, an MTA must be able to determine the next entity to take responsibility for a message in the MTA's possession. Without this ability, messages could enter the MTS and "disappear." The ability to calculate an entire route at one time can be useful, but is not strictly necessary.

### 3.2.7 Message Buffering and Storage

Kernel:

The MTS buffers messages reliably while they are in transit between UAs.

Value-added features:

The MTS provides reliable permanent storage for messages.

Rationale:

Buffering of messages in transit is vital to the correct operation of the MTS. Permanent storage facilities may be used to support requests for permanent storage from UAs lacking such facilities.

### 3.2.8 Accounting

Kernel:

The MTS makes information available about its usage for accounting purposes.

Value-added features:

The MTS provides accounting functions.

Rationale:

If the MTS cannot perform billing and settlement, it must at the very least provide the information that serves as a basis for them.

### 3.2.9 Acknowledgments

Kernel:

MTAs must be able to generate positive and negative acknowledgments to MTS transactions.

Value-added features:

No acknowledgments value-added features.

Rationale:

Some operations which the MTS performs cannot be assumed to be complete without a positive acknowledgment. Message cancellation is an example of this. Other MTS operations are assumed to have succeeded unless there is a negative acknowledgment. Message delivery is an example of an operation which requires the use of a negative acknowledgment to signal failure. In this case, the originating UA assumes, after a certain amount of time, that a message which it posted has been delivered to the receiving UA. If that is not true, the MTS gives the originating UA a negative acknowledgment to the posting operation.

Within the MTS, MTAs must be able to acknowledge protocol transactions with their peers. This is necessary in any protocol transaction whose success or failure must be signalled before the transaction can be said to be complete. The transfer of a message between two MTAs is an example of this. The MTA which receives the message must tell the sending MTA that it is in possession of the message and has accepted it.

### 3.2.10 Security

Kernel:

Messages can be delivered to the UA only at the address specified on the message envelope. (Messages are forwarded by changing the destination information on the message envelope.)

Value-added features:

1. The name-to-address mapping for the recipient of a message is verified immediately before an MTA tries to deliver the message. If the address associated with the name has changed, the message is rerouted to the new address.

2. Measures are taken against unauthorized use of the MTS.

3. Transmitted data is encrypted and decrypted.

4. Messages are routed through specified MTAs.

Rationale:

In most situations the security policy calls only for delivering a message only to the address on its envelope. This policy is supported by the protocol kernel.

Sometimes a more stringent security policy is in effect. Several measures are possible, encryption, special routing, or verification of the identity of a receiving UA at delivery time. These features are included in the value-added portion of the protocol.

## REFERENCES

[1] J. Burrus.
Features of Transport and Session Protocols.
NBS Report ICST/HLNP-80-1, U.S. Department of Commerce/National Bureau of Standards, March, 1980.

[2] S. Clopper.
Features of the File Transfer Protocol (FTP) and the Data Presentation Protocol (DPP).
Draft Report ICST/HLNP-80-6, U.S. Department of Commerce/National Bureau of Standards, September, 1980.

[3] S. Clopper.
Service Specification of Data Presentation Protocol (DPP) and File Transfer Protocol (FTP).
Technical Report ICST/HLNP-80-9, U.S. Department of Commerce/National Bureau of Standards, October, 1980.

[4] Institute for Computer Sciences and Technology.
Federal Computer Network Protocol Standards Program: An Overview.
Technical Report, U.S. Department of Commerce/National Bureau of Standards, 1980.

[5] National Bureau of Standards.
Specification of Message Format for Computer Based Message Systems.
Report ICST/CBOS-81-??, US Department of Commerce / National Bureau of Standards, October, 1981.

[6] Peter Schicker.
The Computer Based Mail Environment: An Overview.
Technical Report, Bell-Northern Research Ltd., Ottawa, Ontario, Canada, December, 1979.

[7] CCITT Study Group 8.
Control Prpocedures for the Teletex Service.
Draft Recommendation S.62, CCITT, 1980.

# INDEX

Document No. 0300008

# BBN O/S Mail System Tutorial

**BBN Communications Corporation**

A Subsidiary of Bolt Beranek and Newman Inc.

# BBN O/S Mail System Tutorial

October 1983

Document No. 0300008
Revision 2.0
October 1983

CONTENTS

# PREFACE

## Scope

This is a combined tutorial and reference manual for the BBN-UNIX electronic mail system, which is comprised mainly of the msg program for reading and filing mail, and the sndmsg program for sending mail. This software is a standard part of the UNIX operating system as modified and extended by Bolt Beranek and Newman (BBN). It is not to be confused with the much more sophisticated InfoMail electronic mail and information management system, which is also available from BBN Communications Corporation as a self-contained product.

## Purpose

The chief purposes of this manual are:

- to enable new users of the mail system to learn quickly how to send and receive mail, and how to file messages in UNIX files when that is appropriate
- to enable users who send and receive electronic mail regularly to learn more advanced features and various abbreviations and shortcuts supported by the software
- to provide a comprehensive reference to mail system commands and functions for all users

## Audience

The intended audience for the tutorial material in Chapters 2 through 4 are new users who are at least somewhat familiar with data manipulation on the UNIX operating system. Readers of the reference material in the remaining chapters are expected to be familiar with the basics of sending and receiving electronic mail as described in the tutorial sections.

## Organization

- o Chapter 2 describes how to send mail to other users with the sndmsg program.
- o Chapter 3 describes how to read mail with the msg program.
- o Chapter 4 briefly describes the news facility for posting and reading public notices.

These three sections are sufficient to get started using the mail system. If you are going to send or receive mail routinely, however, you should read about the more advanced features of the mail system in the remaining sections.

- o Chapter 5 gives a detailed description of the msg program for reading and manipulating incoming mail.
- o Chapter 6 describes advanced features of the sndmsg program not presented in Chapter 2.
- o Chapter 7 describes several specialized utilities that are occasionally useful.
- o Appendix A is a summary of msg commands.

## Related Documents

This manual assumes that you have previously used the UNIX operating system. If this is not the case, you should read the document BBN-UNIX User's Guide (formerly entitled "An Introduction to Systems and Utilities") and get some familiarity with the UNIX operating system before proceeding.

## Documentation Conventions

In this description, the names of keys on the keyboard and of ASCII characters are printed in angle brackets, as for example <erase> and <RETURN>. The abbreviations used are as follows:

| | |
|---|---|
| <BOF> | Beginning Of File |
| <BOL> | Beginning Of Line |
| <BS> | Backspace |
| <RETURN> | Carriage Return |
| <DEL> | Delete |
| <EOF> | End Of File |
| <EOL> | End Of Line |
| <ESC> | Escape |
| <LF> | Line Feed |
| <TAB> | Tab |

The carat <^> is the abbreviation for the <CTRL> key when it is used as a shift. For example, <^D> means to type a <D> while simultaneously holding down the <CTRL> key.

# CHAPTER 1
# INTRODUCTION

The BBN-UNIX mail system provides a coherent set of utilities for processing electronic mail. It allows you to manipulate individual messages separately or in groups. The system is simple enough to learn easily if you are a casual user, yet sophisticated enough to enable you to cope regularly with large volumes of electronic mail.

The two major programs that make up the mail system are sndmsg, for sending messages to other users, and msg, for reading mail that other users have sent to you. In addition to these, the mail system provides a news facility for posting and reading public notices; the splmsg program for splitting over-large mailboxes; and the rr and rrh programs for reading a mailbox in reverse order. (These last are used to implement the news command.)

You may temporarily exit to a UNIX subshell from either the msg or the sndmsg program to use other UNIX facilities, and then return to the mail program at the point of interruption. In particular, you may edit a message with an editor such as ed or Pen before sending it with the sndmsg program.

# CHAPTER 2
# Sending Mail

There are a number of ways to send mail with the sndmsg program, ranging from a highly interactive mode to one in which the program prompts you step by step for all the information it needs, to one in which you type all pertinent information on the command line for greater speed and efficiency. In this section we describe the interactive mode, since it provides more feedback to the user.

The simplest way to send mail is to type the name of the program as a UNIX command when you are at the UNIX Shell level, as follows:

```
     sndmsg
```

(Read an introductory document such as the BBN-UNIX User's Guide if you do not understand how to enter UNIX commands.) The sndmsg program responds by printing:

```
     To:
```

This prompt is a request for you to type a list of recipients, using their log-in names, and separating the names with commas. Normally, the list is terminated by typing <RETURN> (carriage return). If you wish to continue typing on the next line, you should type a comma as the last character on the "To:" line. This tells the sndmsg program to expect additional list entries. The list may be continued onto as many lines as necessary, with all but the last line terminated by a comma. Spaces and tabs may be used to separate the names and commas in the list, and have no meaning to the sndmsg program. However, there must be no space after a comma that signifies the continuation of a list onto the next line.

After you have specified the list of primary recipients, the sndmsg program prints the following prompt:

```
     Cc:
```

This prompt is a request for you to type a list of secondary ("carbon copy") recipients, if any, just as you typed the names of the primary recipients, above. Any recipients listed on either the "To:" or "Cc:" lists receive your message. The only difference is the heading after which the recipient sees his or her name listed. However, at least one name must appear in the "To:" field, while the "Cc:" field may be omitted by typing a line consisting of only a carriage return.

After you have specified the list of secondary recipients, the sndmsg program prints:

```
     Subject:
```

This prompt invites you to type an arbitrary string of text summarizing the contents of the message you are about to send. If the string is so long that it cannot fit onto one line, you may type a '\' (backslash) as the last character on the line, and then continue typing on the following line. The text in the "Subject:" field is not interpreted by the sndmsg program in any way -- its sole purpose is to inform the recipients of the message. Again, this field may be entirely omitted by typing a line consisting of only a carriage return.

You are now ready to type the body of your message. The sndmsg program indicates its readiness to accept text by printing:

```
     Type message
     -----
```

and leaving the cursor or printhead at the first column of a new line. Whatever you type now appears in the body of the message. When you have finished, type a <^D> or <^Z>. (That is, while holding down the <CTRL> key as a shift, press the <D> or <Z> key.) The sndmsg program responds by printing:

```
     -----
     Disposition (type ? for help):
```

If you are satisfied with the message as it stands, type

```
     s
```

or simply a carriage return. The sndmsg program attempts to send a copy of your message to each recipient you have listed, and informs you of the result of each attempt. The sndmsg program asks you to retype or delete any recipients that it cannot find, as described in Section 6.2. If, even after all recipients have been verified or corrected, an attempted delivery fails (perhaps because of some system problem), the sndmsg program appends the body of your message to a file with a name like "msg12Feb162730" in your home directory. The name of the file is different from the names of all other files in your home directory. Failure to complete one of the deliveries does not, however, prevent delivery to other recipients.

The other options presented by the sndmsg program are discussed in section 6.2.

## 2.1 A Brief Example

Suppose you wish to send a reminder to a group of people, with a "carbon copy" of the message going to yourself. To do so, use the following sort of dialogue with the system (text typed by the user is underscored):

```
$ sndmsg<RETURN>
To: john, joe,bob<RETURN>
Cc:  yourname<RETURN>
Subject: "Tomorrow's" meeting<RETURN>
Type message
-----
...will be held in the 4th floor conference room<RETURN>
at 2:00.
        Yourname
<^D>
-----
Disposition (type ? for help): <RETURN>
Mail to john: Sent
Mail to joe: Sent
Mail to bob: Sent
Mail to yourname: Sent
$
```

# CHAPTER 3
# READING YOUR MAIL

Whenever you log in, the system checks to see whether you have any messages in your mailbox. If you do, the msg system tells you:

```
     You have new mail
```

or something to that effect, immediately after you are shown the message of the day. To read your mail, just type:

```
     msg
```

when you are at the Shell level. The msg program displays a list of messages that have arrived since the last time you read your mail. Each entry in the list takes up one line, and summarizes the information contained in the header of the message.* For example, if George had two new messages waiting, the msg program would print something like this when it started up:

```
     MSG: type ?<RETURN> for help
     New messages:
     +    2  12 Feb 1981  Bill Jones <wjones a  FORTRAN Documentation
     +    3  12 Feb 1981 >john at BBN-UNIX, jo  Tomorrow's meeting
     3 messages in file /usr/yourname/mailbox
     <-
```

* The header is the group of lines before the message body which tells who sent the message, when it was sent, the subject of the message, and possibly other information.

The "+" at the beginning of a line tells you that the message so marked is new. Other symbols may appear in this position, as will be explained in section 5.3.1. The number in the second column is the message number, with which you can refer to a message in subsequent commands.

After showing you the condensed headers, the msg program prints the symbol "<-". This is the prompt, which tells you that the msg program is ready to accept commands. The most important of these commands is type, which lets you look at the contents of messages. For example, to read the second new message, whose message number is 3, you would say:

```
<- type 3<RETURN>
```

and the msg program answers:

```
           Number: 3  Length: 214 bytes
        Date: 12 Feb 1981 18:19:13 EST
        From: Your Name <yourname at BBN-UNIX>
        Subject: Tomorrow's meeting
        To: john at BBN-UNIX, joe at BBN-UNIX, bob at BBN-UNIX
        Cc: yourname at BBN-UNIX

        ...will be held in the 4th floor conference room
        at 2:00.
                Yourname

        <*>
```

Similarly, you can look at other messages, both old and new, using the type command. To get a list of all the messages in your mailbox, including old ones, use the command:

```
     <- headers all
```

which in this case might result in the following output:

```
            1  23 Jan 1981  John Wilson <jwilson  Compiler blowing up
     +      2  12 Feb 1981  Bill Jones <wjones a  FORTRAN documentation
            3  12 Feb 1981  >john at BBN-UNIX, jo  Tomorrow's meeting
     3 messages in file /usr/yourname/mailbox
     <-
```

Message number 3 is no longer marked "new" because you just looked at it.

When you want to exit from the msg program you have three options:

- o Give the _quit_ command, to leave your mailbox in the same state it was in before you entered the msg program.
- o Use the _exit_ command to move all your messages to the file .../savebox (.../ means it is in your home directory), leaving your mailbox empty.

We describe the third option in the next section, after we have seen how to modify a mailbox.

# CHAPTER 4
# THE NEWS FACILITY

The news facility is a mechanism for posting and reading public notices. These notices generally consist of system announcements, unveiling of new programs, requests for comments, and the like. To send a message to the news program, just use the sndmsg program as already described, giving "news" as the name of the recipient. For example, you might type:

```
$ sndmsg<RETURN>
To: news<RETURN>
Cc: <RETURN>
Subject: Mail system<RETURN>
Type message
-----
A new version of sndmsg has been installed.
Please report any bugs to me. Thanks.
        Yourname<RETURN>
<^D>
-----
Disposition (type ? for help): <RETURN>
Mail to file /usr/news/mailbox: Sent
$
```

The meaning of the message "Mail to file /usr/news/mailbox: Sent" will become clear after we have discussed aliases in section 6.4.

A program called news enables you to read news items. You invoke it simply by typing:

```
news
```

at the Shell level. The news program prints out the news items in reverse order (i.e., newest item first). For each item, it prints the header fields and asks you whether you wish to see the message body, as in

```
     Date: 24 Feb 1981 16:15:01 EST (Tuesday)
     From: George Jones <george at BBN-UNIX>
     Subject: Mail system

     Option ? :
```

If you then type <RETURN> (or a line beginning with "y"), the news program prints the message, and then goes on to the header for the next most recent item. If you type "s", the news program skips the message and goes on to the next header. If you type "b", the news program skips back to the previous message, and if you type "p", (for "put") the news program asks you for the name of a file into which the message you are looking at should be saved. You can type "?" to get a list of the available options. If you type anything else, the news program terminates and returns you to the Shell.

# CHAPTER 5
# MSG REFERENCE MANUAL

Of course, there are many other operations that you might wish to perform on messages. Often, you will want to discard a message after reading it, or save it in some file other than your primary mailbox. Or you may want to forward a message to someone else. We describe these and other useful operations in this section. First, though, we describe in a general way how to put together a msg command.

## 5.1 Command Structure

Each msg command consists of a verb followed by zero or more arguments. When typing the command, you need only type the first letter or letters of the name, giving the shortest unambiguous abbreviation. Thus

```
h all
```

is equivalant to the command

```
headers all
```

since there is only one command that starts with the letter h.

The several argument types are defined according to the different contexts in which they may occur. The most common type of argument is the message sequence, which specifies the set of messages to which a command is to be applied. These are described in the next section. Another common argument type is the file name, used to specify a file in the UNIX file system. Wherever a file name argument may occur, the msg program recognizes and expands shell variables, that is, any words that start with the character. (To prevent this special interpetation of "$", precede it by the "\" (backslash) character.) Finally, there are several other argument types that are only used in certain commands. We discuss these when we present the applicable commands later in this chapter.

## 5.2 Message Sequences

We have already seen one simple type of message sequence: the message number, which refers to a single message. This is a special case of the numbered message sequence. The numbered sequence consists of one or more message ranges separated by commas. A message range, in turn, consists of either a single message number, or two message numbers separated by a colon or a dash (minus sign). The latter refers to all the messages between and including the first and second numbers.

A few examples should clarify this. Each of the commands in the following table prints the headers for a set of messages. Note that the order in which messages are specified is significant.

| Command | Messages affected |
|---|---|
| h 2 | 2 |
| h 1-4 | 1, 2, 3, and 4 |
| h 3,5:7,9 | 3, 5, 6, 7, and 9 |
| h 8,4-2 | 8, 4, 3, and 2 (in that order) |

Alternatively, instead of specifying messages by number, you can select messages based on whether or not they meet a certain criterion. There are fourteen different criteria or message classes that the msg program knows about. Each class is specified in a command by typing its name in place of a numerical sequence. As in the case of command verbs, you need type only the shortest unambiguous prefix of the name. Since all of the class names start with a different letter, it always suffices to type just a single letter. Below, we give the names of the classes, and explain which messages each class refers to.

| | |
|---|---|
| all | All messages. |
| current | The current message. This is the default used by the delete command if no message sequence is given explicitly. Generally, the current message is the last sage examined by an earlier command. The commands that set the current message are as follows:<br><br>answer<br>backup<br>forward<br>go<br>list<br>nofflist<br>move<br>next<br>put<br>read<br>type<br>redistribute |
| deleted | All deleted messages, that is, all messages for which the headers command shows a "*" in the first column. This is the inverse of the undeleted class (see below). |
| examined | All messages which have been examined, that is, all messages for which the headers command does not show a "+" or a "-" in the first column. This is the inverse of the not [examined] class (see below). |
| from \<string\> | All messages for which \<string\> is found as a substring of the "From" field of the header, not distinguishing upper-case and lower-case letters. If \<string\> contains any blanks or tabs, it must be enclosed in double quotes. |
| inverse | All messages in reverse order. |
| last | Message sequence used in the previous command. This is the default message |

sequence used by all but the delete command when no message sequence is given explicitly.

| | |
|---|---|
| not | All messages which have not been examined, that is, all messages for which the headers command shows a "+" or a "-" in the first column. This is the inverse of the examined class. |
| old | All old messages, that is, all messages for which the headers command does not show a "+" in the first column. This is the inverse of the recent class (see below). |
| recent | All new messages, that is, all messages for which the headers command shows a "+" in the first column. This is the inverse of the old class. |
| subject &lt;string&gt; | All messages for which &lt;string&gt; is found as a substring of the "Subject" field of the header, not distinguishing upper-case and lower-case letters. If &lt;string&gt; contains any blanks or tabs, it must be enclosed in double quotes. |
| to &lt;string&gt; | All messages for which &lt;string&gt; is found as a substring of the "To" or "Cc" fields of the header, not distinguishing upper-case and lower-case letters. If &lt;string&gt; contains any blanks or tabs, it must be enclosed in double quotes. |
| undeleted | All undeleted messages, that is, all messages for which the headers command does not show a "*" in the first column. This is the inverse of the deleted class. |
| zlast | The last (i.e., highest-numbered) message. (Mnemonic value: z is the last letter of the alphabet.) |

As an example, suppose our mailbox contains the following messages:

```
     1  23 Jan 1981  John Wilson <jwilson  Compiler blowing up
```

```
+    2  12 Feb 1981  Bill Jones <wjones a  FORTRAN documentation
     3  12 Feb 1981 >john at BBN-UNIX, jo  Tomorrow's meeting
```

The table below shows the messages referred to by a series of headers commands.

| Command | Messages affected |
|---|---|
| h examined | 1 and 3 |
| h last | 1 and 3 |
| h s fortran | 2 |
| h f jones | 2 |
| h recent | 2 |
| h i | 3, 2, and 1 (in that order) |
| h | 3, 2, and 1 (default is last sequence used) |

It should be noted that only one kind of message sequence can be used in a command. Thus it is not possible to select the intersection or the union of two classes, nor to specify a class plus some numbered messages. For example, all of the following commands have illegal arguments:

```
headers ex,2-5
headers exam,recent
headers 1-5,from jones
```

## 5.3 Description of Commands

With the preliminaries out of the way, we are now ready to discuss the command set in detail. We group the commands somewhat arbitrarily according to their major functions. In the following six sections we consider these categories of commands:

1) Examining messages - how to look at the contents of your mailbox.
2) Responding to messages - what you can do after you have read a message.
3) Modifying your mailbox - how to make reversible changes in the state of your mailbox.
4) Updating your mailbox - how to make the changes caused by group (3) commands permanent.
5) Utilities - generally useful commands, not directly related to mail.
6) Miscellaneous - other commands for dealing with

messages and mailboxes.

Most of the commands described below take a message sequence as one of their arguments. Instead of stating this for each such command, we simply refer to "the specified messages" or the "specified message sequence", implying that the command takes a message sequence as an argument. Where a command may be given additional arguments, we describe the syntax in order to show the proper order of the arguments.

## 5.3.1 Examining messages

When the msg program starts up, it performs the command headers recent. You may at any time invoke the headers command yourself to see what messages are in your mailbox. This command prints one line for each specified message, summarizing the information contained in the header fields. The general format of the output** is shown below:

```
     x    #  <date sent>  <sender / recipient>  <subject>
```

These symbols have a "precedence", such that "recent" overrides "unexamined", and "deleted" overrides both "recent" and "unexamined". In other words, any unexamined message that is recent shows a "+" in the first column, and any message at all that has been deleted shows a "*" in the first column.

The symbol in the second field of the line (shown here as "#") is the sequential message number, used to uniquely identify the message in the msg program commands. The third field gives the date the message was sent (the first eleven characters of the Date: field in the header). The fifth field shows the first thirty-six characters of the header's Subject: field. The fourth field usually indicates who sent

** The character in the first column (shown here as "x") tells you the state of the message: "+" means the message is recent (new), "*" means the message has been deleted, and "-" means that the message has not been examined. The precise meaning of "recent" is that the message was added to the mailbox (by the sndmsg program) since the last time the mailbox was examined and updated by the msg program, using the overwrite, exit, or xexit command.

the message by printing the first twenty characters of the header's From: field. However, if you were the sender of the message, the msg program instead prints the symbol ">" followed by the first twenty characters of the header's To: field. We saw a case of this in the example on page 5, where the headers command printed the following line for the third message in our mailbox:

```
+    3  12 Feb 1981 >john at BBN-UNIX, jo  Tomorrow's meeting
```

Use the type command to print messages on your terminal. For each of the specified messages, it prints a line giving the message number and the message length in characters, then the message header and body, and finally a line consisting of the end-of-message symbol

```
     <*>
```

This symbol indicates the end of the message. As a side effect, the type command sets the current message to the last message it prints. (See above under Message Sequences for a discussion of the idea of the current message.) Because the type command ignores deleted messages, the only way you can read a deleted message is to first undelete it. (The same restriction applies to the put, list, nofflist, and move commands--see below.) For the same reason, it cannot set the current message to the number of a deleted message.

If the message is more than one screenful, use <^S> to stop it from scrolling off your screen. You can also pipe the message through the UNIX more program or the variant page program by typing:

```
     pipe <message sequence> more
```

The more program displays one screenful at a time, moving on to the next screenful when you press <SPACE BAR>, or to the next line when you press <RETURN>.

The next command prints the next message on your terminal (next is with reference to the current message). It also sets the current message to the number of the message printed.

The backup command prints the previous message, and sets the current message to the number of the message printed.

The current command tells you what the name of the mailbox file is and what the current message number is, as in the following typical example:

```
     Current message is 2 of 3 messages in file /usr/george/mailbox
```

The type, next, and backup commands cause the messages they print to be marked as "examined".

### 5.3.2 Responding to messages

After you have read a message, there are several actions you might wish to take. For example, you may like to keep all messages on a particular subject together in one message file. You can use the list, nofflist, put, and move commands to distribute messages to subsidiary mailboxes. The syntax of these three commands is

```
     <cmd> <message sequence> <filename>
```

where <cmd> is list, nofflist, put, or move, and <filename> is a UNIX file name that may contain references to shell variables. The list command acts just like a headers command followed by a type command, all directed to a file. First it writes the one-line header for each specified message, followed by a form feed. Then, for each message, it writes a line giving the message number and the message length, followed by the header and body of the message, followed by a line consisting of the end-of-message symbol

```
     <*>
```

If the file does not exist, it is created. If the file already exists, the text is appended to the end of the file. This command is useful when the resulting message file is to be printed on a line printer, using a UNIX Shell command like

```
     $pr msgfile | print
```

The printed output would contain all the headers on the first page, followed by the messages themselves on succeeding pages. The nofflist command performs the same function, except that instead of inserting a form feed after each message, the msg program inserts six blank lines. This command can be used to save paper when printing out your mailbox.

Similar in function is the put command, which writes the specified messages in mailbox format. That means that messages stored with the put command can later be processed by the msg program. Finally, the move command has the same effect as the put command followed by the delete command. All three of these commands mark a message as "examined"; the move command also marks it as "deleted." If the file specified with a put command or move command does not already exist, the msg program asks you for confirmation before creating the file. Typing y for "yes" or just <RETURN> instructs the msg program to proceed. Any other response aborts the command without creating the file.

Often you will want to respond to a message by sending a reply to the author, and possibly to others. The answer command is provided for this purpose. The syntax of this command is

```
     answer <message number> <answer option>
```

where <message number> is a single message number, and <answer option> is a string (of which only the first letter is significant) indicating to whom the reply should be sent. In place of a <message number> you may specify one of several mnemonic strings: beginning (of file - message number 1), current (the current message number), last, zlast, or end (of file). The last three all refer to the highest-numbered message. You need only type the first character of any of these strings. If you omit <message number>, you answer the current message by default.

The possibilities for <answer option> are:

| String | Reply is sent to |
|---|---|
| from | Sender of original message only |
| to | Sender and all recipients in "To:" list |
| receivers | Same as to |
| cc | Sender and all recipients, including "Cc:" list |
| all | Same as cc |

If you omit the <answer option>, from is used by default.

If a message has a "Reply-to:" field, the reply is sent to anyone listed there, rather than to the user given in the "From:" field.

After you have typed the answer command, the msg program prints the one-line header for the specified message in the format used by the headers command. The msg program then asks you to type in the names of any others who should receive your reply. After entering this list (it may be empty), type <RETURN>. The msg program then starts up the sndmsg program, which you may use in the usual way to compose and send the body of your reply. When you have finished, and your message has been sent off, the msg program prints its prompt to indicate that it is ready to accept further commands.

Occasionally, you may receive messages that should have been sent to someone else, either in addition to or instead of yourself. The redistribute command allows you to copy messages to one or more other users. The syntax is

```
redistribute <message sequence>
```

The msg program prints the one-line header for the specified messages (in the format used by the headers command) and prompts you with

```
To:
```

At this point you should enter the list of users who should receive the redistributed messages.

After you have done this and the messages have been delivered to those you specified, the msg program is ready to accept more commands. The new recipients get a copy of each message that looks exactly like the original, except that each header contains three new fields:

```
Resent-by:
Resent-to:
Resent-Date:
```

The Resent-by: field contains your user name, since you initiated the redistribution, the Resent-to: field contains the list of names you typed in, and the Resent-date: field indicates the time that the redistributed messages were sent off.

If anyone replies to a redistributed message using the answer command with either the to answer option or the all answer option, the reply goes to those on the Resent-to: list. Note that the redistribute command does not delete the messages from your mailbox.

Similar in purpose to the redistribute command is the forward command. The syntax is

```
     forward <message sequence>
```

This command might be thought of as a combination of the answer command with the redistribute command. As with the redistribute command, the msg program prints the headers for the specified messages, and then prompts you for a list of those to whom the messages are to be forwarded. It also prompts you for a "Cc:" list and a "Subject:" line. The latter two fields may be omitted by simply typing <RETURN>. If the subject field is omitted, it is copied from the subject field (if any) in the first forwarded message. The "To:" line must not be omitted. The msg program then starts up the sndmsg program, and the sndmsg program prompts you in the usual way by printing

```
     Type message
     -----
```

If you wish to add any commentary after the forwarded messages, you can do so at this point. When you are done, finish up by typing <^D> and sending the message off as described in section 2. The sndmsg program delivers the message and exits, leaving you back inside of the msg program. The recipients of the forwarded messages receive a single message that looks something like this:

```
     [Header including your "To:", "Cc:", and "Subject:" fields]
     ----BEGINNING OF FORWARDED MESSAGES----
     [Complete text of forwarded messages, concatenated together]
     ----END OF FORWARDED MESSAGES----
     [Additional text, if any, that you entered in the sndmsg program]
```

For instance, if we were to forward messages 1 and 2 from the example in section 3 to a user named smith, the latter might receive a message that looked like this:

```
Date: 25 Feb 1981 21:55:28 EST (Wednesday)
From: Your Name <yourname at BBN-UNIX>
Subject:  Interesting messages
To: smith at BBN-UNIX


----BEGINNING OF FORWARDED MESSAGES----
Date: 24 Feb 1981 16:24:15 EST (Tuesday)
From: John Wilson <jwilson at BBN-UNIX>
Subject: Compiler blowing up
To: yourname at BBN-UNIX

The C compiler blows up on the following program:

main ()
{       exit(0); }

/John

Date: 12 Feb 1981 17:49:58 EST (Thursday)
From: Bill Jones <wjones at BBN-UNIX>
Subject: FORTRAN documentation
To: yourname at BBN-UNIX

George,
Do you know if any documentation exists for the
FORTRAN system we are running?

                              Bill


----END OF FORWARDED MESSAGES----
Thought you might want to have a look at these.
        Yourname
```

The final command in the present category is the command sndmsg, which simply starts up the sndmsg program. You may specify a full set of arguments, just as if you were running the sndmsg program from the Shell. Shell variables are expanded in the arguments to this command. All of the commands described in this section, except the sndmsg command, set the current message to the number of the last message affected.

## 5.3.3 Modifying your mailbox

The following commands directly modify the state of your mailbox:

```
delete
undelete
mark
unmark
sort
```

If you delete a message, the type, list, nofflist, put, and move commands subsequently ignore it Furthermore, when your mailbox is updated (see next section) the message is permanently expunged, never to be seen again. The undelete command undoes the effect of a delete command. (The system does not give you an error message if you undelete a message that was not deleted). These commands determine whether or not a message is included in the "deleted" and "undeleted" classes of message sequences (as described in section 5.2).

The mark and unmark commands set the state of the specified messages to "examined" and "unexamined" respectively. Thus they determine whether or not a message is included in the "examined" and "not [examined]" classes of message sequences.

The sort command reorders the messages in your mailbox according to one of three sorting criteria. The syntax of this command is

```
sort <sort option>
```

where <sort option> may be one of the strings date, from, subject, or an abbreviation of one of these strings. After this command is executed, the messages are sorted on the specified field. For example, after you give the command

```
sort subject
```

a message whose subject field is "Lost files" would have a message number greater than a message whose subject field is "Compiler bugs." For all three sorting options, the sort is based strictly on lexicographical ordering. Therefore, a sort by date may not work if the messages in your mailbox have varying date formats.

### 5.3.4 Updating Your Mailbox

While you are inside the msg program, examining, deleting, undeleting, marking, and unmarking messages, no changes are made to your mailbox file itself. The msg program simply keeps track of the state of each message internally.

After you have finished looking at your mailbox, you will either want to leave the msg program and go back to the Shell, or switch your attention to another mailbox. Before doing either of these, you will probably want to make any changes in the state of your mailbox permanent. Three commands are provided for this purpose. ix recent (message class) The overwrite command preserves the current state by expunging all deleted messages, and returning all undeleted messages to the mailbox. The msg program then tells you how many messages your mailbox now contains. Any messages which had been shown as "recent" (i.e., with a "+" in the first column) or "not examined" (with a "-" in the first column) are now shown as "not examined" or "examined," depending on whether or not you looked at them. The overwrite command never leaves a message marked as "recent." The effects of any sort commands are preserved, since the undeleted messages are written out in the current numerical order as shown by a headers command.

After you have given an overwrite command, you may switch to another mailbox by using the read command (see below).

The xexit command (abbreviated x) performs an overwrite followed by a quit (see below). That is, it expunges deleted messages, returns undeleted messages to your mailbox, and exits back to the Shell. The exit command (abbreviated e) is similar, but instead of returning undeleted messages to your mailbox, it deposits them in the file named savebox in your home directory.(Messages are not appended to the savebox file when you start the msg program up with a file name as argument, that is, when you are not processing your primary mailbox.)

Of course, you may wish to leave the state of your mailbox unmodified. You can do this by using the read command without first giving the overwrite command, or by using the quit command rather than the exit or xexit command.

### 5.3.5 Utility Commands

You may regularly read two or more mailboxes (for example, your main mailbox and a mailbox which receives bug reports). Or, you may keep your subsidiary mailboxes in a subdirectory. In these cases, it is convenient to be able to change to a different directory from within the msg program, in order to avoid typing full pathnames. The command cd (change directory) is provided for this purpose. This command works in exactly the same manner as the Shell command that bears the same name. It takes a UNIX pathname as an argument, and tries to change to the directory specified. If the argument is omitted, it changes to your home directory. If an argument is present, it may contain references to Shell variables. As in the Shell, the chdir command is a synonym for cd.

It is frequently useful to run UNIX commands from within the msg program. For example, you might want to run the UNIX ls command to see what message files were contained in a directory. The shell command (abbreviated sh) provides a way to do this. If you type

```
     shell <cmd> <arg1> <arg2> ...
```

the command <cmd> is executed with arguments <arg1> <arg2> ..., exactly as if you typed it at the Shell level. If the character "&" is typed at the end of the command line, the command is run in the background. That is, control returns to the msg program before the Shell command has terminated. If you give the shell command with no arguments, the msg program starts up an interactive Shell for you. You may then run any number of UNIX commands. When you exit from the Shell (usually with <^D>), the msg program is ready to read more msg commands.

Two other commands which are available are the sdate command, which prints the current date and time, and the ? (question mark) command, which prints a brief description of the msg commands.

### 5.3.6 Miscellaneous commands

When you are through processing the messages in one mailbox, you may want to switch your attention to another mailbox. One way to do this is to quit out of the msg program and go back to the Shell, and then to start it up again on the other mailbox. You can accomplish exactly the same effect with the read command. This command takes a UNIX filename as an argument, reads the messages in, and shows you the headers for any recent (new) messages it contains. That is, it behaves exactly as if the msg program were starting up with the specified file as your mailbox. It should also be noted that you can specify a file other than your primary mailbox by giving the msg program an argument when you start it up from the Shell, as in

```
     msg <file>
```

where <file> is the name of the mailbox you want to examine.

The pipe command runs a UNIX command, using the text of the specified messages as the command's standard input. Its syntax is

```
     pipe <message sequence> <UNIX command>
```

A typical use is

```
     pipe <message sequence> lpr
```

which prints the specified messages on the line printer. If the character "&" is typed at the end of the command, it is executed in the background. That is, control returns to the msg program before the command terminates.

The go command takes a message number as an argument, and changes the "current message" to be the specified one. The quit command terminates the msg program and returns you to the Shell without updating your mailbox, so that the next time you run the msg program you will find your mailbox in exactly the same state as before, except for the possible addition of new messages you have received.

## 5.4 Command Completion

As we have seen, there is a large number of msg commands, and a wide variety of argument sequences. To help you use commands correctly, the msg program provides a feature called command completion. This facility allows the msg program to show you what command you are typing, and to prompt you for arguments.

At any point while you are typing in a command, you can press the escape key, usually labelled <ESC> on the keyboard. If you have typed an ambiguous abbreviation of a command name or argument, the msg program types out the rest of the name or argument, and then prompts you for the next argument. Otherwise, the msg program rings the bell on your terminal. If you immediately type another <ESC>, the msg program uses the default value, if any, for that argument (showing you what the value is), and goes on to prompt you for the next argument. Typing <ESC> after you have entered the last argument has the same effect as typing <RETURN>, i.e., it tells the msg program to go ahead and execute the command. At any time before you type <RETURN> or the final <ESC>, you can cancel the command you are entering by pressing the interrupt key (usually <RUBOUT> or <^C>).

In the following examples, assume the current message number is 3. The text typed by the msg program is shown underlined. The places where the user types <ESC> are shown with a "$", although the <ESC>s do not actually echo.

```
     <- pu$t (message sequence) ol$d messages into (file name) mbox$
     <- re$<bell>     [bell rung - command name is ambiguous]
     <- red$istribute (message sequence) $3
          3  12 Feb 1981 >john at BBN-UNIX, jo  Tomorrow's meeting
     To:
```

Because the current message is number 3, the last <ESC> causes the system to insert "3" in the msg command line.

Many commands allow you to omit some arguments. For example, a command that takes a message sequence as an argument uses either the last message sequence or the current message as a default, depending on the command. Other argument types, such as filenames, have no defaults, and must always be supplied. Thus if you type the command

```
     <- put 2-4<RETURN>
```

the msg program prints

```
     (file name)
```

If you do not supply a file name at this point, the msg program refuses to execute the command. Thus, if you want to do the minimum amount of typing, and you always want to use default values for arguments, you can just type the shortest unambiguous abbreviation of a command name followed by <RETURN>. The msg program always prompts you for those arguments that have no defaults.

## 5.5 Command-Line Options

As we have already mentioned, you can give the msg program an argument on the command line to tell it to use a mailbox other than your primary mailbox. Thus, for example, typing the command

```
     msg savebox
```

at the Shell level has the same effect as just typing

```
     msg
```

and then immediately giving the msg program the command

```
     read savebox
```

Using the Shell command is quicker of course, and avoids printing out irrelevant information about new messages in your primary mailbox.

There is one other way to invoke the msg program from the Shell. The general form is

```
     msg <file> <command>
```

where <file> is the name of a mailbox, and <command> is a valid msg command. When invoked in this manner, the msg program does not print its introductory message, and does not show the headers of new messages. It simply runs the single <command> specified on <file>, and then exits. If the symbol "-" (dash) is used for <file>, then the command applies to your primary mailbox. For example, the command

```
     msg - h a
```

would print out the headers for all the messages in your primary mailbox, and immediately return you to the Shell.

## 5.6 Broken Mailboxes

Although it should never happen, a mailbox file may occasionally be slightly damaged, so that it is no longer in a valid format. The msg program detects this situation soon after startup when it is reading the messages. It offers to repair the mailbox with the following message:

```
     Bad format in msg file mailbox at line *** 225 0 in message 2
     Reading of file discontinued.
     Attempt repairs? [confirm]
```

If you type "y" at this point, the msg program first copies the mailbox file to a file of the same name ending in "~". It then attempts to reconstruct the bad file. Usually, it succeeds, and gives you a message like the following:

```
     backing up file mailbox to file mailbox~
     Attempting repairs on msg 2...success.  Msg 2 fixed
     3 messages in file mailbox.
```

On some occasions, you may find that two or more messages have been combined after such a repair attempt.

# CHAPTER 6
# OTHER FEATURES OF THE SNDMSG PROGRAM

## 6.1 Including a File in a Message

To include some text from a file in your message, you can type <^F> or <^B> (both have the same effect). The sndmsg program prompts you with

```
     Type file name:
```

at which point you should enter a valid UNIX file name and type <RETURN>. The contents of the file you specified are appended to the end of the text you have typed in so far. The sndmsg program indicates this by typing

```
     has been included
```

At this point, you may continue typing. Any text you enter is appended after the text of the included file. Note that any number of files can be included in a single message.

## 6.2 Editing your message

It is entirely possible that you may make some errors while you are typing in the text of your message. You can of course, use your normal erase and kill characters to edit your text in the usual manner. In addition, the sndmsg program itself recognizes several special characters. They may be typed at any time while you are entering the body of the message.

To delete an entire word from your text, type <^W>. Any spaces and tabs after the last word you typed is erased, followed by the last word itself. You can only delete words on the current line. If you want to see what your message looks like so far, type <^T>. This causes the sndmsg program to print the header and body of the message on your terminal.

If you need to do substantial editing on your message body (for example, the changes affect more than one line), you have to use a text editor to make the corrections. To invoke an editor, type <^E>. The sndmsg program tells you to

```
     enter editor name:
```

You should type in the name of your favorite text editor, followed by <RETURN>. In a moment you find yourself in the editor, with your message body as the initial editing buffer. After you have made any desired changes, write out the buffer to save the editing session, and then quit. (When using the Pen editor, simply exit with the Exit command <^X^Z> and the editor automatically "saves" or write out the buffer to a msg file.) When you exit from the editor, you are back in the sndmsg program, which tells you to

```
     continue typing now
```

Anything you type now is appended to the end of the just-modified message body.

Some people prefer always to compose their messages entirely within an editor. There are two ways to do this, both of them described in the next section.

The last special character recognized by the sndmsg program is the interrupt character (usually either <RUBOUT> or <^C> in UNIX). When you type <INTERRUPT>, the sndmsg program aborts and places the body of your message in a uniquely named file in your home directory. The file name is of the form

```
     msg12Feb162730
```

You can later start up the sndmsg program again and read in the contents of that file with <^B> or <^F>, or with

```
     < filename
```

on the command line when you invoke the msg program.

## 6.3 Terminating The sndmsg program

In our discussion in section 2, we saw that when you type <^D>, it signals the sndmsg program that you have finished entering the body of your message. At that point, the sndmsg program prints

```
     -----
     Disposition (type ? for help):
```

At the time we said you should just type <RETURN> to tell the sndmsg program to deliver your message. However, if you wish to make corrections to the header fields, you may do so at this point. The corrections you may make include additions to and deletions from the "To:" and "Cc:" lists, and replacement of the subject line. To add new recipients to the "To:" list, type

```
             addto <list><RETURN>
     or
             to <list><RETURN>
```

where <list> is a comma-separated list of addressees. The new addressees are added to the header of the message, and the specified users also receive copies of the message. To add recipients to the "Cc:" list, type "addcc" or simply "cc" instead of "addto" or "to". To delete a set of recipients from the "To:" list, type

```
             delto <list><RETURN>
     or
             dto <list><RETURN>
```

where <list> is the comma-separated list of recipients whom you want removed from the "To:" list. Upper and lower case are not distinguished here in either user or host names. To delete recipients from the "Cc:" list, type "delcc" or "dcc" instead of "delto" or "dto". To replace the subject field with a new one, type

```
            subject <text>
```

where <text> is any string of characters terminated by <RETURN>. For all these options, if you omit the argument, the sndmsg program prompts you to enter the argument on a separate line. If you use "delto" to empty the "To:" list, the sndmsg program asks you to give at least one addressee for that list.

The other options you have are as follows:

h h (for header) causes the sndmsg program to display the header of the message. Any editing changes you have made are visible at this time.

m (for message) causes the sndmsg program to display the header and the body of the message.

a (for abort) has the same effect as typing interrupt, as described above: that is, the message body is placed in a uniquely named file in your home directory, and the sndmsg program terminates.

q (for queue) causes the sndmsg program to fork off a subprocess to deliver the mail. The original (parent) process immediately returns to the Shell. This means that you do not have to wait for all the deliveries to finish before you get back to the Shell. Also, the sndmsg program does not report to you the results of each delivery. If any delivery fails, the unsent message is still mailed back to you. Note that mail to a foreign network host is always queued. (See the next section.)

Typing "e" (for edit) has exactly the same effect as typing <^E> while entering text. Typing "i" (for "improve spelling") causes the UNIX spell program to be run on your message. If any spelling errors are found, the edtypo program is run to allow you to correct them. Typing "!" places you in a Shell, where you can execute UNIX commands and then return to the sndmsg program. If you type

```
     !<command>
```

then the UNIX <command> is executed. Finally, typing "?" causes a list of one-line descriptions of the options to be printed.

When you finally instruct the sndmsg program to deliver the message to the intended recipients, the sndmsg program attempts to verify the existence of all the users before delivering the mail to any of them. Anytime it finds a non-existent user (or an unknown host name in the case of network mail) the sndmsg program types something like

```
     Can't find user smythe.
```

or

```
     "bbn-unx" is an unknown host in address "jones at bbn-unx".
```

followed by

```
     Retype addressee or press <RETURN> to delete:
```

at this point you should specify a new addressee, or if you prefer, you may eliminate the incorrect address entirely by simply typing <RETURN>. If doing so causes the "To:" list to become empty, the sndmsg program asks you to enter at least one address for the "To:" list.

Before delivering any copies of the message, the sndmsg program removes any duplicate occurrences of an address. For example, if "smith at BBN-UNIX" is found on both the "To:" and "Cc:" lists, the program removes its occurrence on the "Cc:" list. If this was the only address on the "Cc:" list, then that list disappears entirely from the message header.

## 6.4 Command-Line Options

In section 2 we saw how you can let the sndmsg program prompt you for all the information it needs. As you become more adept at using the sndmsg program, you may find this method somewhat tedious. You may instead specify most or all of the information on the command line when you start up the sndmsg program. The general form of the command is

```
     sndmsg [[-to] to-list] [-cc cc-list] [-subject subject] \
     [-file filename] [-editor [editor-name]]
```

The command sndmsg and the keywords "-to," "cc," "-subject," and "-editor" must be typed literally, although the keywords may be abbreviated to one letter. We have split the command across two lines for readability. If you must continue the command onto more than one line, you must terminate all but the last line with the character "\," as we have done above. This is the general mechanism for telling the Shell that a logical command line extends over several physical lines.

Items shown in brackets are optional. Thus the simplest way to invoke the program is just to type

```
     sndmsg
```

as we have already seen. On the other hand, we could have sent the sample message in section 2.1 by typing the command

```
     sndmsg john,joe,bob -c george -s Tomorrow\'s meeting
```

and then typing the body of the message when the sndmsg program prompted us with

```
     Type message
     -----
```

In fact, if we had placed the text of the message we wanted to send in a file called, say, meet, we could have completed the entire the sndmsg session by simply typing

```
     sndmsg john,joe,bob -c george -s Tomorrow\'s meeting <meet
```

using the Shell's input-redirection facility to supply the message body. Note that we had to escape the character "'" with a backslash to prevent it from being interpreted by the Shell. Notice also that we have abbreviated the keywords "-cc", and "subject;" we could have abbreviated the optional keyword "-to" to "-t," but we chose the even simpler alternative of omitting it. In general, any of the keywords may be shortened to as little as one letter, and the keyword "-to" may be omitted entirely if the to-list is the first argument given.

The to-list and the cc-list are lists of user names, separated by spaces or commas. Each user name is either a simple log-in name, if the recipient is on the local machine, or a log-in name and host specifier, if the recipient is on a foreign network host. There are four ways to specify a network address:

```
user @ host       (three command-line arguments)
user@host         (one argument, no spaces)
user -at host     (three arguments)
"user at host"    (one argument, must be enclosed in quotes)
```

If your line-kill character is the character "@" (this is the UNIX default), then you must precede any other use of this character on a command line with a "\" (backslash). Otherwise, you would wipe out the command line by typing "@".(However, when you are entering addresses in interactive mode, the sndmsg program turns off your kill character if it is "@". This means that you can enter this character without escaping it with "\", but it also means that you do not have any way to erase the line you are typing in interactive mode. When you begin entering the text of your message, the sndmsg program restores the meaning of "@".) If the to-list or cc-list on the command line begins with the keyword -at, the host name following is applied to all recipients on the list up to the first one with an explicit -at (or @) host suffix. This default host may be changed within a name list by using

```
-host host
```

at any point within the list. Note that the default host name does not carry across from one name list to another. Several examples are given below:

```
sndmsg -to jones@bbna smith
```

sends mail to user jones at host bbna, and to smith on the local machine, while

```
sndmsg  -to -at bbnd jones smith martin @ bbnc -cc wilson
```

sends mail to users smith and jones at host bbnd, martin at host bbnc, and a copy to wilson on the local machine.

There are two more command-line options that we have not yet described. If you include

```
     -file filename
```

in the command, the file filename is inserted as the first portion of the message to be sent, and anything you type afterwards is appended to the end of that material. This has the same effect as starting up the sndmsg program and immediately using <^F> to read in <filename>. The

```
     -editor editorname
```

option specifies the name of a text editor that the sndmsg program should start up as soon as it is ready to accept text. You must compose your entire message within the text editor, since when you write the buffer and quit out of the editor, the sndmsg program immediately asks

```
     Disposition (type ? for help):
```

and you have no further opportunity to add text (without going back into the editor). This option also overrides the editor option in .msg.profile (see below).

## 6.5 Miscellaneous Features

When the sndmsg program starts up, it looks for a file called .msg.profile in your home directory. If such a file exists, the sndmsg program checks for the presence of two special keywords, each of which must occur on its own line. One of these keywords is sndmsgcc, which causes you to automatically receive a "carbon copy" of every message that you send. Similarly, msgcc in the .msg.profile file causes the msg program to give you an automatic "carbon copy" of every message you answer or forward. The third possible keyword in the .msg.profile file is editor, followed by the name of a text editor. If the sndmsg program finds an editor line it immediately puts you into the specified editor. The effect is the same as if you had used the -editor option on the command line. However, if you give the command-line option and you have an editor line in .msg.profile, the command-line option takes precedence. Furthermore, if you omit the editor name from the command-line option, that effectively negates the editor line in .msg.profile. That is, typing

```
     sndmsg -editor
```

ensures that your message text is read directly by the sndmsg program, rather than by a text editor.

There are several ways to specify a set of recipients for a message, other than simply giving their names in the to-list or cc-list. A name ending in a ":" (colon) is interpreted as the name of a file which in turn contains the names of recipients. This to-list file may contain one or more lines, each of which is a comma-separated list of addressees. There is also a database of mailing lists maintained in the file /etc/net/aliases. Each line in this file is of the following form:

```
     aliasname:user1,user2,user3,...
```

Just before the sndmsg program delivers a message, it checks the name of each recipient against all the lines in /etc/net/aliases. For each line where the name to the left of the colon matches the recipient name, the latter is replaced by the list of names to the right of the colon. These names may contain further aliases, which the program then expands, and so on to any level. Finally, a name ending in a "*" (star or asterisk) is interpreted as the name of a UNIX file to which the message should be appended. The major uses for these features are:

a) to have all your mail on one machine forwarded to another machine, as in

```
            george:george@bbnu
```

b) to establish a "pseudo-user", as in

```
            news:/usr/news/mailbox*
```

which allows users to mail to "news" even though there is no one with that log-in name;

c) to allow other forms of your name to be used in a name list, as in

```
            rob:rjones
```

d) to establish a mailing list, that is, a list of users who should receive all the messages on a given topic, as in

```
             info-terms:george,jones,wilson,fred,bill
     or
             msgbugs:smith,/usr/bugs/mailbox*
```

Note that this last alias causes mail to be sent both to a user and to a file.

e) to append "cc" copies of your messages to files of your choice, if you keep all messages on certain topics for future reference.

# CHAPTER 7
# OTHER MAIL-RELATED PROGRAMS

There are three small utility programs associated with the mail system. The first is called splmsg, and is used to split up a mailbox when it is too large for the msg program to handle. The general form of the command is

```
     splmsg infile outfile
```

If outfile is omitted, then nmailbox is used. If infile is omitted, then the file mailbox in your home directory (which is your system mailbox) is used. The splmsg program splits infile up into groups of fifty messages. The first group is written to the file outfile, the second group to the file outfile, and so on. The file infile is not modified.

Another program is called rr. It simply reads a mailbox in reverse order. For each message, it prints the header fields and then asks you whether you want to proceed. For example, if we ran the rr program on the mailbox from section 3, we would see the following output:

```
     Date: 24 Feb 1981 16:15:01 EST (Tuesday)
     From: Your Name <yourname at BBN-UNIX>
     Subject: Tomorrow's meeting
     To: john, joe, bob
     Cc: yourname

     Option ? :
```

When the rr program prints "Option ? :", you can type <RETURN> to see the body of the message, "s", to skip the message and see the next header, "b" to skip back to the previous message, "p" to put the message in a file, or any other character to quit out of the program. This program is very useful for such tasks as reviewing a long list of accumulated bug reports. A slight variation called rrh does not print the "To:" and "Cc:" fields of the headers, but is otherwise the same as the rr program. An example of the use of this program is the one-line shell script:

```
     rrh /usr/news/mailbox
```

which implements the UNIX news command.

# APPENDIX A
# MSG COMMAND SUMMARY

Usages:

```
     msg
```

Read mail from system mailbox (i.e., the file homedir/mailbox, where homedir is your home directory).

--------------------

```
     msg <filename>
```

Read mail from the mailbox file called filename.

--------------------

```
     msg <filename> <command> <args>
```

Execute the msg command <command> with the arguments <args> on the mailbox filename. Return to the Shell after executing the command.

--------------------

```
     msg - <command> <args>
```

Execute the msg command <command> with the arguments <args> on your system mailbox. Return to the Shell after executing the command.

--------------------

Commands to msg consist of one verb followed by zero or more arguments. The characters typed may be the minimum unambiguous prefix of the command name. Typing <ESC> causes command completion. Capitalized characters show the minimum unambiguous form of a command. The commands accepted are:

Answer <message number> <answer option>

Send replies to those specified in the answer option. If the answer option is not specified, the from option (described below) is assumed. Possible answer options are:

| | |
|---|---|
| From | Reply only to the sender of the message. |
| To | Reply to all those in the "To" list and the sender of the message. |
| Receivers | Same as To. |
| Cc | Reply to all who received the message including the sender. |
| All | Same as Cc. |

The answer command also asks for additional addresses in case anyone should be added to the mailing list.

Backup

Type the previous message.

CD <directory name>

Change the current directory to the one specified. If <directory name> is omitted, change to the home directory.

CHdir <directory name>

Synonym for the cd command.

CUrrent

Type the number of the current message, the total number of messages in the file, and the name of the message file.

Delete <message sequence>

Delete the specified message(s).

Exit

Exit, delete all deleted messages from the mailbox file and move the others to savebox.

Forward <message sequence>

Send the message(s) to a different address(es). Sndmsg is called to mail the forwarded messages.

Go to <message number>
Make the specified message the current one.

Headers <message sequence>
Type the header information on the specified message(s).

List <message sequence> <file name>
List copies of the specified message(s) on the file designated, without message separators. Separate each pair of messages with a form feed.

MArk <message sequence>
Mark a message as having been examined.

MOve <message sequence> <file name>
Move the specified message(s) to the designated file. The message(s) are deleted from the original file. This function is equivalent to put plus delete.

NExt
Type the next message.

NOfflist <message sequence> <file name>
List copies of the specified message(s) on the file designated, without message separators. Separate each pair of messages with six blank lines.

Overwrite
Overwrite and re-read the current file; this makes any requested deletions effective.

PIpe <message sequence> <shell command>
Writes out the messages specified by <message sequence> to a pipe which is used as the input to the shell command.

PUt <message sequence> <file name>
Put copies of the specified message(s) to the file designated, with message separators (i.e., in mailbox format).

Quit
   Quit, leaving everything unchanged. Another way to quit msg is to type two <EOT>s (usually <^D>). The first <EOT> elicits a "update message file?" prompt. One can either type a y to update the message file, or the second <EOT> to completely quit the program without updating.

REAd <file name>
   Read the mail from the file designated. From now on all operations will be on the designated file, as if the file had been specified as an argument to msg.

REDistribute <message sequence>
   Like the forward command, but the specified messages are forwarded unchanged, except for the addition of three lines in the headers, indicating when, by whom, and to whom each message was redistributed.

SDAte
   Print current date.

SHell <shell command>
   Execute the specified command in a sub-shell.

SNdmsg <sndmsg arguments>
   Execute the sndmsg program.

SOrt <sort option>
   Sort the mail by the given option. Valid options are: date, from, subject, number.

Type <message sequence>
   Type the message(s) specified.

UNDelete <message sequence>
   Undelete the message(s) specified if they were already marked as deleted.

UNMark <message sequence>
   Unmark messages to indicate not examined.

Xexit
   Overwrite the message file and then exit.

?
   Help. Prints a brief description of the commands.

A <message number> is either a single number or one of the following letters:

| | |
|---|---|
| b | the first message |
| c | the current message |
| e | the last message |
| l | the last message |
| z | the last message |

If a <message number> is omitted from a command, the current message number is used by default.

A <message sequence> is one of the following:

A single number n means the nth message.

Two numbers separated by a colon or dash (minus sign) means all messages from the first message through the second. If the first number is less than the second, then messages are operated on in ascending order; if the first is greater than the second and the numbers are separated by a colon then they are operated on in descending order.

Additionally, a message class is acceptable as a <message sequence>. Letters are used to abbreviate the following classes:

| | |
|---|---|
| a | all messages |
| c | current message |
| d | deleted messages |
| e | examined messages |
| f | all messages from a specified user |
| i | all messages in inverse order |
| l | last message sequence used |
| n | not-examined messages |
| o | old messages |
| r | recent messages |
| s | all messages about a specified subject |
| t | all messages to a specified user |
| u | undeleted messages |
| z | last message. |

Operations such as list, put, and move which involve a file append the text to the file if it already exists. Any legal file name may be specified.

## INDEX

## Reader's Comments Form

---

**BBN O/S Mail System Tutorial**

Document Number 0300008-20

---

Your comments and suggestions will help us improve the quality and usefulness of this publication. All comments and suggestions become the property of BBN Communications Corporation.

---

Technical or editorial errors (include page number):

---

Suggestions for improvement:

---

Please print identifying information below:

Name: ______________________ Date: __________

Title: ______________________

Company Name: ______________________

Address: ______________________

City: ______________ State: ______ Zip: ________

FOLD HERE - TAPE CLOSED

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**BUSINESS REPLY MAIL**

FIRST-CLASS PERMIT NO. 36450 CAMBRIDGE, MA

POSTAGE WILL BE PAID BY ADDRESSEE

BBN COMMUNICATIONS CORPORATION
50 Moulton Street
Cambridge, MA 02238

**ATTN: Documentation Department**

FOLD HERE - TAPE CLOSED

|SP| = SPACE-bar. Used as a separator with MAILSYS commands.

|ESC| = ESCAPE-key. Used to "extend" MAILSYS commands. As soon as you have typed enough characters to disambiguate a command or an object, |ESC| causes MAILSYS to print the rest of the word and if appropriate, some "noise words" that tell you what to type in next.

Lowercase words enclosed in angle brackets, such as <addressee list>, <file-name> and <name>, are used to indicate descriptions of information that must be supplied by the user.

```
Ex.: >>SUB|ESC|ject: <text-string>|CR|
```

means that after you type the beginning of the command Subject: and press the ESCAPE-key, the computer pauses and waits for you to type in the text of the Subject:-field. You must end the CARRIAGE-RETURN-key.

-----

## D. MODES OF INPUT: FULL, RECOGNITION AND ABBREVIATION

MAILSYS commands, like TENEX commands, give you a choice of three types of input.

FULL INPUT MODE: Type in the full name of the command:

```
>QUIT |CR|   or   >QUIT|CR|
```

RECOGNITION INPUT MODE: Type the minimum number of characters and then press the ESCAPE-key. The MAILSYS system then prints out the rest of the command designation:

```
>Q|ESC|uit |CR|   or   >Q|ESC|uit|CR|
```

ABBREVIATED INPUT MODE: Type the minimum number of characters necessary for MAILSYS to recognize the command:

```
>Q |CR|   or   >Q|CR|
```

As these examples show, you may type commands without inserting a space between the command name and the |CR|. Alternatively, you may insert one, and only one, space before the |CR|.

-----

## E. COMMANDS WITH MULTIPLE PARTS

The QUIT command has only two parts: the command name and the text string. Other commands can have several parts.

RECOGNITION INPUT MODE. Press the ESCAPE-key after each part of a command. The command for filing a message is:

```
>FI|ESC|le (messages)<message-no.>|ESC| (on file)<filename>|CR|
```

The phrases "(messages)" and "(on file)" are "noise words" that the system prints to prompt you to type in the required information.

ABBREVIATED INPUT MODE: Replace the ESCAPE-key |ESC| by a SPACE-bar |SP|:

```
>FI|SP|<message-no.>|SP|<filename>|SP||CR|
```

which appears on the page as:

```
>FI <message-no.> <filename>
```

No noise words are shown.

If you type |CR| before you type enough characters for recognition, or make any other mistake in typing a command, the MAILSYS system prints a question mark ?, aborts the command, and starts over with a new ready signal:

```
>LS|CR|
?
>
```

If you type too few characters for recognition and then press |ESC|, MAILSYS rings a bell and waits for more input:

```
>S|ESC| "br-r-ring!"
```

To make examples short and clear, the MAILSYS commands are usually shown typed in the abbreviated mode, but the full command name is printed.

```
>ST|ESC|atus
```

MAILSYS extends the command. Since the Status Command does not take an object, MAILSYS waits for the |CR|. If you type a question mark:

```
>STatus ?
(Abort with |DEL|: confirm with |CR|)
```

-----
F. HOW TO CORRECT SIMPLE MISTAKES:
|Control-A| deletes a single character. If you make a mistake in typing one or a few characters, you can correct your work, one caracter at a time, with the |Control-A| signal. |Control-A| deletes the most recently typed character, and indicates the deletion by printing the backslash \ followed by the character deleted.

```
MAIO|Control-A|\OLSYS
```

In the printout at your terminal, this appears as

```
MAIO\OLSYS
```

The string that goes to the computer is

```
MAILSYS
```

-----
G. THE PANIC BUTTON:
|Control-E| aborts any MAILSYS command and allows you to type in a new command.
-----------

I-SUMMARY:-MESSAGE-HANDLING-COMMANDS

> = prompt signal for MAILSYS at top "command level".

```
>Summarize <sequence> <destination>|CR|
>Summarize|CR| = Summarize RECENTMESSAGES TTY:|CR|
          Always prints according to STEMPLATE.

>Survey <sequence> <destination>|CR|
>Survey|CR| = Survey CSEQUENCE TTY:|CR|
          Always prints according to STEMPLATE.

>List <sequence> <destination>|CR|
>List|CR| = List CSEQUENCE LPT:|CR|
          Prints Index page according to STEMPLATE, followed by
          messages printed according to PTEMPLATE.  Each message
          begins on a new page.

>Print <sequence> <template> <destination>|CR|
>Print|CR| = Print CMESSAGE PTEMPLATE TTY:|CR|
          General command for printing messages.  Sets CMESSAGE
          to last message on <sequence>.

>|LF|     Changes CMESSAGE to the next message on CSEQUENCE and
          Prints it according to PTEMPLATE on your terminal.

>NEXT|CR| = |LF|

>^        "Reverse" of |LF|.  Changes CMESSAGE to the preceding
          message on CSEQUENCE and Prints it on your terminal.

>File <sequence> <destination)|CR|
>File|CR| = File CMESSAGE FDESTINATION|CR|
          Command for creating MAILSYS-readable message-files.
          FDESTINATION is a TENEX file name that is NOT TTY: or
          LPT:.  MAILSYS standard SWITCH setting for File does
          not Delete <sequence> from CMESSAGE-FILE.

>Move <sequence> <destination> |CR|
>Move|CR| = Move CMESSAGE FDESTINATION|CR|
          = File, except that MAILSYS Standard SWITCH setting for
          Move asks whether to Delete <sequence> from
          CMESSAGE-FILE.

>Mark <sequence> <status>|CR|
>Mark <status>|CR| = Mark CMESSAGE <status>|CR|
 where <status> may be
  SEEN
  UNSEEN
  DELETED
  UNDELETED

>Delete <sequence>|CR|
>Delete|CR| = Delete CMESSAGE|CR|
          Marks messages DELETED.

>Undelete <sequence>|CR|
>Undelete|CR| = Undelete CMESSAGE|CR|
          Removes DELETED markings.

>Expunge|CR|
          Physically removes messages marked DELETED and
          renumbers the CMESSAGE-FILE.
-----------

II-SUMMARY:-DRAFT-EDITOR:-DRAFT-CREATING-AND-EDITING-COMMANDS

>Compose|CR|
          Prompted composition

>Reply <message-no.>|CR|
>Reply|CR| = Reply (CMESSAGE)|CR|
        Prompted composition

>Forward <sequence>|CR|
          Prompted composition

>Send|CR| (Confirm)|CR|
          Places a copy of CDRAFT in queue for MAILER to pick up
          and deliver as an outgoing message.  Standard MAILSYS
          Switch Setting erases contents of CDRAFT.

>Mailer|CR|
          Forces immediate delivery of queued outgoing messages.

>Mailstat|CR|
          Lists queued and undeliverable outgoing messages.


-----
II-A. DRAFT-EDITOR

>Create DRAFT|CR| = Create|CR|
or
>Edit CDRAFT|CR|
          Calls DRAFT EDITOR for composition of CDRAFT; does not
          erase pre-existing contents of CDRAFT.
>> = Subcommand prompt signal

>>Show <All, Headers or Field-name>|CR|
>>Show|CR| = Show All|CR| = Show CDRAFT|CR| at command-level.
          Shows fields as they will be transmitted.

>>Show TYPEDFORM <All, Headers or field-name>|CR|
>>Show TYPEDFORM|CR| = Show TYPEDFORM All|CR|
          Shows fields exactly as typed.

>>Erase <All, Headers or field-name>|CR| (Confirm)|CR|
>>Erase|CR| = Erase All|CR| = Erase CDRAFT|CR| at command level.
          Erases contents of fields.

>>Compose|CR|
          Prompted composition.  Also top-level command.

>>Forward <sequence>|CR|
  Forward|CR| = Forward CMESSAGE|CR|
          Prompted composition.  Also top-level command.

>>Reply <message-no.>|CR|
>>Reply|CR| = Reply CMESSAGE|CR|
          Prompted composition.  Also top-level command.

>>Include <sequence>|CR|
>>Include|CR| = Include CMESSAGE|CR|
          Puts messages in Text:-field.

>><field-name>|CR|
  where <field-name> may be:
   From:
   To:
   Cc:
   Subject:
   Bcc:
   Message-ID:
   In-Reply-To:
   References:
   Keywords:
   Precedence:
```

```
   Message-Class:
   Special-Handling:
   Fcc:
   Text:
         Creates field, or adds to current contents of existing
         field.

>>Save-Field <All, Headers or field-name>
 (on file) <TENEX file name>|CR|
         Stores contents of field in file.

>>Add-File (from file) <TENEX file name> (to) <field-name>|CR|
         Appends contents of file to field.

>>Restore-Draft (from file) <TENEX file name>|CR|
         Can be used only with file created by Save-Field All or
         Save-Field Headers.  Appends contents of each field in
         file to the corresponding field of CDRAFT.

>>Teco <field-name>|CR|
>>Teco|CR| = Teco TEXT|CR|
         Calls the TECO Editor.  Return to MAILSYS DRAFT-EDITOR
         by typing ;H|ESC| to * prompt.

>>Neted <field-name>|CR|
>>Neted|CR| = Neted TEXT|CR|
         Calls the Neted Editor.  Return to MAILSYS DRAFT-EDITOR
         by typing QUIT |CR| or SAVE<filename>|CR| to prompt.

>>Xed <field-name>|CR|
>>Xed|CR| = Xed TEXT|CR|
         Calls the Xed Editor.  Return to MAILSYS DRAFT-EDITOR
         by typing E|CR| or Q|CR| to prompt.

>>Format <field-name>|CR|
>>Format|CR| = Format TEXT|CR|
         Evens up lines within paragraphs.

>>Send|CR|
         Also top-level command.

>>Mailer|CR|
         Also top-level command.

>>Mailstat|CR|
         Also top-level command.

>>Print <sequence> <template> <destination>|CR|
>>Print|CR| = Print CMESSAGE PTEMPLATE TTY:|CR|
         Also top-level command.

>>Survey <sequence> <destination>|CR|
>>Survey|CR| = Survey CSEQUENCE TTY:|CR|
         Also top-level command.

>>Done|CR|
         Does not change contents of CDRAFT.  Returns you to top
         command level.
-----------
```

III-SUMMARY:-UTILITY-COMMANDS

```
>Quit|CR|
         Returns you to the TENEX Exec.  MAILSYS Standard SWITCH
         Setting for Quit asks whether to Expunge CMESSAGE-FILE.
```

```
>Exit|CR|
          = Quit, except that MAILSYS Standard SWITCH Setting for
          Exit does not Expunge CMESSAGE-FILE.
```

-----
## III-A. DOCUMENTATION COMMANDS

```
>News|CR|
          Prints latest changes in MAILSYS.

>Help|CR|
          Prints introductory paragraph for new users.

>Describe <topic> <destination>|CR|
          Detailed documentation.  Type Describe? for list of
          topics.
```

-----
## III-B. DEBUGGING COMMANDS

```
>Version|CR|
          Prints information useful in debugging MAILSYS.

>Suggestion|CR|
          Prompted composition.  Creates and immediately sends
          Suggestion-message to MAILSYS design team.
```

-----
## III-C. TENEX EXECUTIVE COMMANDS

```
>Exec|CR|
          Allows you to use the TENEX Exec in a "lower fork".
          Return to MAILSYS by typing Quit to the @ prompt
          signal.

>Run <TENEX file name>|CR|
          Executes a TENEX Exec program.

>Daytime|CR|
          Prints date and time on User's terminal.

>Directory <TENEX file name>|CR|   Prints names of files.
>Directory|CR|
          Prints list of files in User's TENEX file directory.

>QFD <TENEX file name>|CR|
          Prints all TENEX directory information for User's file
          directory.

>Check-Inbox|CR|
          Checks for new messages in INBOX.  Useful when you are
          in some other message-file.

>Jobstat|CR|
          Prints TSS Job No., name of LOGIN directory and
          terminal line.

>Talk     allows you to type a string of characters that are
          ignored by MAILSYS.  Terminates with |Control-Z|.

;         allows you to type a one-line string of characters that
          are ignored by MAILSYS.  Terminates with |CR|.
```
-----------

## IV-SUMMARY:-OBJECT-HANDLING-COMMANDS

-----

### IV-A. OBJECTS IN THE ACTIVE ENVIRONMENT

```
                          Setting at Start
                          of MAILSYS Session

CDRAFT                    [Empty]

CMESSAGE-FILE             CMESSAGE-FILE in PROFILE

Sequences:
 System:
  CMESSAGE (=.)           At Beginning of RECENTMESSAGES
  CSEQUENCE               ALLMESSAGES
  ALLMESSAGES (=*)
  NULLSEQUENCE
  NEWMESSAGES
  RECENTMESSAGES
  OLDMESSAGES
 User:

Filters:
 System:
  CFILTER                 CFILTER in PROFILE
  NULLFILTER
  SEEN
  UNSEEN
  DELETED
  UNDELETED
  TODAY
  YESTERDAY
 User:                    User-Created Filters in PROFILE

Templates:
 System:
  PTEMPLATE               PTEMPLATE in PROFILE
  STEMPLATE               STEMPLATE in PROFILE
  RCVD-FORM
  STANDARD
  SUMMARYFORM
  NULLTEMPLATE
 User:                    User-Created Templates in PROFILE

Destinations:
 System:
  FDESTINATION            [None]

SWITCHES:                 SWITCHES Settings in PROFILE
```

### IV-B. OBJECT-HANDLING COMMANDS

```
>Show <object>|CR|
>Show|CR| = Show All|CR|
 where <object> may be:
  All
  Names
  Current-Objects
  SWITCHES
  Profile
  the name of any object

>Show All <object type>|CR|
>Show All|CR| = Show All Objects|CR|
 where <object type> may be:
  Objects
  Fields
  Sequences
  Filters
  Templates
  Destinations

>Status|CR| = Show CMESSAGE-FILE|CR|

>Copy <object> (and name it) <name>|CR|
 where <object> may be
  CSEQUENCE
  CFILTER
  PTEMPLATE
  STEMPLATE
  Any User-Created sequence, filter or template.
          copies contents of <object> into nnew or existing
          object nnamed <name>; erases any pre-existing contents.

>Erase <object>|CR| (CONFIRM)|CR|
erases contents of
  CFILTER
  CTEMPLATE
  PTEMPLATE
  STEMPLATE
  FDESTINATION
erases entire object where <object is any User-created sequence,
          filter or template.

>Use (as) <current object> <object>|CR|
 where <current object> may be:
  CMESSAGE-FILE
  CMESSAGE
  CSEQUENCE
  CFILTER
  PTEMPLATE
  STEMPLATE
  FDESTINATION
          Changes the contents of <current object> to a copy of
          <object>.

Three commands, Get, Consider, and Jump-To, are special cases of
          Use:

>Get <TENEX file name>|CR|
   = Use (as) CMESSAGE-FILE <TENEX file name>|CR|

>Consider <sequence>|CR|
   = Use (as) CSEQUENCE <sequence>|CR|

>Jump-To <message-no.>|CR|
   = Use (as) CMESSAGE <message-no.>|CR|

>Create <object-type> <name>|CR|
 where <object-type> may be:
  Draft
  Sequence
  Filter
  Template
          Calls appropriate object editor.
          Create does NOT erase an object that already exists.

>Create|CR| = Create DRAFT|CR|
          See II-SUMMARY: DRAFT-EDITOR
```

```
>Edit <object>|CR|
 where <object> may be:
  CDRAFT
  CSEQUENCE
  CFILTER
  PTEMPLATE
  TEMPLATE
  SWITCHES
  PROFILE
  Any User-Created sequence, filter or template.
-----------

V-SUMMARY:-SEQUENCE-EDITOR

>Create Sequence <name>|CR|
>Create Sequence|CR| = Create Sequence CSEQUENCE|CR|
          Calls SEQUENCE-EDITOR, and inserts a temporary copy of
          sequence named <name> if it exists.  Otherwise, creates
          an empty temporary sequence.
or
>Edit <name>|CR|
          Calls SEQUENCE-EDITOR and inserts a temporary copy of
          sequence named <name> if it exists.  Otherwise, gives
          an error signal.
>> = Subcommand prompt signal

>>Show|CR|
          Shows entire temporary sequence.

>>Survey|CR|
          Prints Surveys of entire temporary sequence.

>>Erase <sequence:2>|CR| (Confirm)|CR|
          Erases from temporary sequence those message-nos.  that
          occur in both the temporary sequence and sequence:2.
>>Erase|CR|
          Erases entire temporary sequence.

>>Sort (by) <attribute>|CR|
          Sorts temporary sequence into alphabetic or numerical
          order.
  where <attribute> may be
   Message-No.
   Char-Count
   Date
   Rcvd-Date
   From-Field
   Message-ID
   Subject-Field

>>Add <sequence:2>|CR|
          Combines temporary sequence and sequence:2, eliminating
          duplicates.
>>Add|CR|
          Gives error report.

>>Intersect with <sequence:2>|CR|
          Erases all message-nos.  except those that occur in
          both temporary sequence and sequence:2.

>>Reverse|CR|
          Reverses temporary sequence.

>>|Control-E|
          Destroys temporary sequence, and does not change
          contents of pre-existing sequence named <name>.
          Returns you to top command level.

>>Done|CR|
          Causes temporary sequence to become new contents of
          sequence named <name>.  Destroys previous contents of
          <name>.  Returns you to top command level.
-----------

VI-SUMMARY:-FILTER-EDITOR

>Create Filter <name>|CR|
>Create Filter|CR| = Create Filter CFILTER|CR|
          Calls FILTER-EDITOR, and inserts a temporary copy of
          filter named <name> if it exists.  Otherwise, creates
          an empty temporary filter.
or
>Edit <name>|CR|
          Calls FILTER-EDITOR and inserts a temporary copy of
          filter named <name> if it exists.  Otherwise, gives an
          error signal.
>> = Subcommand prompt signal

>>Show|CR|
          Shows entire temporary filter.

>>Erase|CR| (confirm)|CR|
          Erases entire temporary filter.

>>Require <attribute>|CR|

>>Reject <attribute>|CR|

>>Ignore <attribute>|CR|

  where <attribute> may be
   SEEN
   DELETED
   <address-field> <address-list>
          Ex.: To: Watson, Holmes
   <header-field> <string list>
          Ex.: Subject: "Dancing men", "ciphers"
   where <string list> consists of one or more <quoted strings>.
          NOTE: A <quoted string> must be enclosed in quotes if
          it contains a space |SP|; it may not contain a |CR|.)

>>Before <date>|CR|

>>After <date>|CR|

>>On <date>|CR|
  where <date> may have the form
   4 JUL 76
   4-JUL-76
   July 4, 1976
   7/4/76

>>|Control-E|
          Destroys temporary filter, and does not change contents
          of pre-existing filter named <name>.  Returns you to
          top command level.

>>Done|CR|
          Causes temporary filter to become new contents of
          filter named <name>.  Destroys previous contents of
          <name>.  Returns you to top command level.
```

----------

## VII-SUMMARY:-TEMPLATE-EDITOR

```
>Create Template <name>|CR|
          Calls TEMPLATE-EDITOR, and inserts a temporary copy of
          template named <name> if it exists.  Otherwise, creates
          an empty temporary template.
or
>Edit <name>|CR|
          Calls TEMPLATE-EDITOR and insrts a temporary copy of a
          template named <name> if it exists.  Otherwise, gives
          an error signal.
>> = subcommand prompt signal

>>Show <line-number>|CR|
>>Show|CR|
          Shows entire temporary template

>>Erase (line number) <number>|CR|
>>Erase|CR|
          Erases entire temporary template.

>>Item-Insert (on) <line-number> (before) <item-number>
  (items) <item-group> |CR|
  where <item-group> is a series of one or more <items>,
          separated by spaces or commas.

>>Line-Insert (before) <line-number>|CR|
(Type template lines, ending with ^Z)
<item-text>|Control-Z|

where <item text> is a series of zero, one or more <items>,
          separated by either spaces, commas, or |CR|s.

<item> may be one of the following:

   Status
          prints -+

   Status+
          prints UNSEEN RECENT

   Verbatim
          prints entire message exactly as received.

   <field-name>
          prints contents of field.

   <field-name>+
          prints name of field followed by contents of field.

   Source
          prints FROM:-field, except that if the message is from
          the User (i.e., if the FROM:-field contains the name of
          the LOGIN directory), Source prints "To:" followed by
          the first name in the To: field.

   "<quoted string>"
          prints <quoted string> exactly as typed.  <Quoted
          string> may include spaces, but may not include |CR|s.

   Separate
          inserts a formfeed |Control-L| that starts a new page
          with the next <item> (or the next message if Separate
```

is at the end of the template).

```
  and <field-name> may be:
   Message-No.
   Char-Count
   Rcvd-Date
   Date:
   Sender:
   Subject:
   From:
   To:
   Cc:
   Info:
   Bcc:
   Message-ID:
   In-Reply-To:
   References:
   Keywords:
   Precedence:
   Message-Class:
   Classification:
   Special-Handling:
   Fcc:
   Other:                   Any other Header-field.
   Text:

>>|Control-E|
          Destroys temporary template, and does not change
          contents of <name>.  Returns you to top command level.

>>Done|CR|
          Causes temporary template to become new contents of
          template named <name>.  Destroys previous contents of
          <name>.  Returns you to top command level.
-----------
```

VIII-SUMMARY:-SWITCHES-EDITOR

```
>Edit SWITCHES|CR|
          Calls SWITCHES-EDITOR.  Places temporary settings of
          SWITCHES in editor.
>> = Subcommand prompt signal

>>Show <switch-name>|CR|
          Shows setting of individual Switch.
>>Show|CR|
          Shows settings of all SWITCHES.

>>Set <switch-name> (to be) <setting>|CR|
          changes temporary setting of individual SWITCH.

>>|Control-E|
          Destroys temporary settings of SWITCHES, and does not
          change previous settings.  Returns you to top command
          level.

>>Done|CR|
          Causes temporary settings of SWITCHES to become new
          settings.  Destroys previous SWITCH settings.  Returns
          you to top command level.
-----------
```

IX-SUMMARY:-PROFILE

-----

IX-A.  PROFILE OBJECTS

MAILSYS Standard Setting

```
Message-Files:
 Current:
  CMESSAGE-FILE       <USER>MESSAGE.TXT;1

Filters:
 Current:
  CFILTER            NULLFILTER
 User-Created:       [None Exist]

Templates:
 Current:
  PTEMPLATE          RCVD-FORM
  STEMPLATE          SUMMARYFORM
 User-Created:       [None Exist]

SWITCHES:        [X] = MAILSYS Standard Setting

 Message-File:
  INITIAL-SURV    [X]InitialSurvey        NoInitialSurvey
  REPORTNEWMSG    [X]Periodic&AtPrompt    AtPrompt            DelayReport
  REPORTFORM      [X]SurveyReport         BulletinRept        NoReport
  GET-DIRECTORY   [X]ConnectDirectory     LoginDirectory
  GET-EXPUNGE     [X]AskExpunge           Expunge             NoExpunge
 Message-Handling:
  DELETE          [X]AskConfirm           NoAskConfirm
  FILE            [X]NoDelete             Delete
  MOVE            [X]NoDelete             Delete
 Draft-Creating:
  FROM-NAME       [X]LoginDirectory       ConnectDirectory
  COMPOSE-SEND    [X]AskSend              Send                NoSend
  FORWARD-SEND    [X]AskSend              Send                NoSend
  FORWARD-COMTS   [X]AskComments          Comments            NoComments
  REPLY-SEND      [X]AskSend              Send                NoSend
  REPLY-COPIES    [X]NoCopies             AskCopies           Copies
  FORMAT          [X]NoJustify            Justify
  SEND-ERASE      [X]Erase                NoErase
  SEND-ARCHIVE    [X]NoArchive            Archive
 Utility:
  QUIT            [X]AskExpunge           Expunge            NoExpunge
  EXIT            [X]AskExpunge           Expunge            NoExpunge

MESSAGE-FILE-RECORD:
Record of the most recent time you entered a message file with
          MAILSYS; automatically generated and updated.
```

-----

## IX-B. PROFILE EDITOR

```
>Edit (object) PROFILE|CR|
          places a temporary copy of the Profile in the
          PROFILE-EDITOR.
>> = Subcommand prompt signal

>>Show <object>|CR|
          where <object> may be All, Names, Current-Objects,
          Switches, or the name of any Object.
>>Show|CR| = Show All|CR|

>>Show All <object-type>|CR| where <object-type> may be Objects,
          Filter, or Templates
>>Show All|CR| = Show All Objects|CR|
>>Erase <filter or template>|CR|
          Erases an object in the Profile.
  Erase|CR|
          Erases all User-created filters and templates.

>>Copy <object> (and name it) <name>|CR|
          copies an object in the Profile.

>>Import (from ACTIVE ENVIRONMENT to PROFILE) <object>
  (and name it) <name>|CR|

>>Export (from PROFILE to ACTIVE ENVIRONMENT) <object>
  (and name it) <name>|CR|

>>Use (as) <current object> <name>|CR|
 where <current object> may be
  CMESSAGE-FILE
  CFILTER
  PTEMPLATE
  STEMPLATE
          Changes the contents of <current object> to a copy of
          <object>.

>>Create <object-type> <name>|CR|
 where <object-type> may be filter or template.
          Creates an object in the Profile.

>>Edit <name>|CR|
where <name> may be
  CFILTER   PTEMPLATE
  STEMPLATE
  SWITCHES
  MESSAGE-FILE-RECORD
  Any User-created filter or template
          Edits an object in the Profile.

>>|Control-E|
          Destroys temporary profile objects and SWITCH settings.
          Does not change previous Profile.  Returns you to top
          command level.

>>Done|CR|
          Causes temporary profile objects and SWITCH settings to
          become new profile.  Returns you to top command level.

-----
IX-C.  FILTER EDITOR IN PROFILE
>>> = subcommand prompt signal
          See VI. FILTER EDITOR

IX-D.  TEMPLATE EDITOR IN PROFILE
>>> = subcommand prompt signal
          See VII. TEMPLATE EDITOR

IX-E.  SWITCHES EDITOR IN PROFILE
>>> = subcommand prompt signal
          See VIII. SWITCHES EDITOR

-----
IX-F.  MESSAGE-FILE-RECORD EDITOR IN PROFILE

>>Edit (object) MESSAGE-FILE-RECORD|CR|
          places a temporary copy of the MESSAGE-FILE-RECORD in
          the MESSAGE-FILE-RECORD-EDITOR.
>>> = Subcommand prompt signal
```

```
>>>Show <message-file number>|CR|
          Shows a single record.
>>>Show|CR|
          Shows entire list of message-file records.

>>>Delete <messsage-file number> |CR| (Confirm)|CR|
          Marks a single message-file record for deletion.
>>>Delete|CR|
          Gives error report.

>>>Undelete <message-file number>|CR|
          Removes deleted marking from a single marking.
>>>Undelete ALL|CR|
          Gives error report.

>>>|Control-E|
          Destroys temporary MESSAGE-FILE- RECORD.  Does not
          change previous MESSAGE-FILE-RECORD.  Returns you to
          PROFILE-EDITOR subcommand level.

>>>Done|CR|
          Causes temporary MESSAGE-FILE-RECORD to become new
          MESSAGE-FILE-RECORD Returns you to PROFILE-EDITOR
          subcommand level.
-----------
```

X-SUMMARY:-CHARACTERS

| | |
|---|---|
| \|Control-A\| | deletes single character |
| \|Control-B\| | <TENEX file name> inserts contents of file inside a CDRAFT field. |
| \|Control-C\| | TENEX interrupt character; returns you to the TENEX Exec |
| \|Control-E\| | MAILSYS interrupt character; Aborts current command and returns you to next higher command level in MAILSYS. |
| \|Control-O\| | Stops printout of text anywhere in MAILSYS. |
| \|Control-Q\| | deletes line of text. |
| \|Control-R\| | reprints line of text. |
| \|Control-S\| | reprints text in Text:-field. |
| \|Control-V\| | causes control character to be entered as ordinary character in text. Does not work for \|Control-C\|, \|Control-E\|, or \|Control-O\|. |
| \|Control-W\| | deletes word in text, name in address-field, or TENEX file name. |
| \|Control-Z\| | Ends Text:-field in DRAFT-EDITOR, and template lines in Line-Insert command in TEMPLATE-EDITOR. |

```
|ESC| = ESCAPE-key; extends commands and object names.
|ALT MODE| = ESCAPE-key
|CR| = Carriage return; ends top commands.
|RUBOUT| = DELETE-key
|LF| = LINEFEED-key; prints next message in CSEQUENCE on TTY:
^    = UP-ARROW; prints preceding message in CSEQUENCE.
|SP| = SPACE-bar character; separates arguments in commands.

?   QUESTION-MARK; Prints out your choices of possible commands,
     objects or other arguments at any point in MAILSYS, except
     in text or TENEX file names.
@  TENEX Exec prompt signal; precedes host names in
     addressee-lists.
>  MAILSYS prompt signal
**  = ALLMESSAGES in CMESSAGE-FILE.
%  = LASTMESSAGE in CMESSAGE-FILE.
/  Slash; separator between sequence and filter.
```

\ Backslash; printed out by |Control-A|.
. Period or dot; symbol for CMESSAGE.
, Comma; separator in some arguments.
: Colon; in sequences 5:10 means 5 through 10; following an
  object, shows that object is a field of CDRAFT.
; Semicolon; separator in sequences. Top-level single-character
  command: introduces a single line of characters that are
  ignored by MAILSYS. Used to "talk" over a TENEX Exec
  "Link".
- Minus sign; to be used in later versions of MAILSYS.
+ Plus sign; also used in TEMPLATE-EDITOR to show that name of
  field is printed out.

----------

^-UP-ARROW-Character

>^

The single-character command UP-ARROW ^ is the "reverse of LINEFEED |LF|; ^ changes the CMESSAGE to the preceding message on CSEQUENCE, and then prints it on the User's terminal, using the PTEMPLATE.

----------

;
Utility Command

>; (Characters to (CR) ignored) <string>|CR|

The single-character command ; allows you to type in a one-line <string> of characters that are ignored by MAILSYS. The string is terminated by |CR|.

The most common reason for using ";" is to "talk" to someone who has linked to you through the TENEX Exec sybsystem "Link".

The MAILSYS command "Talk" allows you to type in a multiple-line string of characters that has no effect on MAILSYS. The "Talk" string is terminated by |Control-Z|.

-----------

|LF|-LINEFEED-key
Message-Handling Command

The single-character commands LINEFEED |LF| and UP-ARROW ^ allow you to "browse" forward and back through CSEQUENCE, using PTEMPLATE and printing messages one by one on your terminal. These two commands cannot be directly mimicked by PRINT, because they allow you to step through the CSEQUENCE automatically, printing one message at a time.

>|LF| changes the CMESSAGE to the next message on the CSEQUENCE
    and then prints the new CMESSAGE on your terminal, using the
    PTEMPLATE.

>^ is the "reverse" of |LF|; ^ changes the CMESSAGE to the
    preceding message on CSEQUENCE, and then prints it.

----------

Add-File
Subcommand of Create Draft; Creates or Appends to a Draft-Field

>>Add-File <TENEX file-name> <field>|CR|

allows you to create any Draft-field from any TENEX file.
Add-File adds the contents of the file to the current contents

(if any) of the CDRAFT field.
----------

Address-Fields
Header-Fields; Subcommands of Create Draft

The address-fields are created, or added to, by the To:, Cc: and Bcc: subcommands. (For sending messages to files, see the Fcc: subcommand.) The address-subcommands have the form

```
>><address-subcommand> <addressee-list>|CR|
```

where <addressee-list> is a series of one or more <name spec>s, separated by commas, and <name spec> has one of the following forms:

<name>
    local user.

<name>(<string>)
    local user with an Attention Subfield.

<name> @ <host>
    user at some other ARPA host computer.

<name> @ <host>(<string>)
    user at some other host computer with an Attention Subfield.

@<host>, <name1>, <name2>, <name3>, ...
    several users at some other host computer.

@<host1>, <name1>, @, <name2>, @<host2>, <name3>(<string>)
    mixed list of local and remote users. Each name may be followed by an Attention Subfield. The host remains the same until another @ is inserted. An "@," means "at the local host."

    where <string> is any string of characters that does not contain a |SP|, |ESC|, |LF| or |CR|. Usually the Attention Subfield will contain names of people or groups, separated by commas.

The address-fields are shown in a standard form different from their typed form.

Smith @ BBN, @ HAWAII-ALOHA, Jones, Robinson (Special-Group)

is shown as:

SMITH at BBN-TENEX, JONES at HAWAII-ALOHA, ROBINSON at HAWAII-ALOHA (Attn: Special-Group).

To show an address-field exactly as typed, use the subcommand

```
>>Show TypedForm <addressee field>|CR|
```

In the current version of MAILSYS, it is possible to filter on a name at a host (local or remote), but not on the attention-subfield. However, the attention-subfields are displayed along with the rest of the Address-Field, and may be used to alert members of a group sharing the same message-file.

Each <name> and <host> is parsed as it is typed in. MAILSYS will refuse to accept an incorrect local name or host name. For a list of local names, consult your local computer center.

For a list of host names, consult the Arpanet Directory or type:

>To: @?

-----
GROUPNAMES

An <addressee-list> can be given a groupname:

<subcommand> <groupname>: <addressee-list>|CR|

where <groupname> is any string of characters that does not contain a |SP|, |ESC|, |LF| or |CR|.

Only the groupname followed by : appears in the address-field. To show the complete <addressee-list> as typed, use the subcommand Show TypedForm. The <addressee-list> is not transmitted when the Draft message is sent, and so does not appear in, and cannot be recovered from, the received message.
-----------

Archive
Switch Option and Subcommand of SEND

SEND-ARCHIVE SWITCH (Send subcommand overrides SWITCH)
ARPA Network Archiving Facility
1. NoArchive: Outgoing messages are not archived in the ARPA Network DATACOMPUTER. (MAILSYS Standard Setting)
2. Archive: Outgoing messages are automatically archived in the ARPA Network DATACOMPUTER for permanent storage.

   The DATACOMPUTER is an experimental, public-access information facility whose entire contents is accessible to all users of the ARPANET. Messages Archived in the DATA COMPUTER can be retrieved with the command Retrieve.

   WARNING!!! The Archive subcommand has no relationship to the Archive facility that may be implemented as part of your local TENEX system.
-----------

Bcc:
Header-Field; Subcommand of Create Draft

>>Bcc: <addressee-list>|CR|

Bcc: allows you to specify a list of addressees to whom "blind carbon copies" of the message are to be sent. The Bcc:-field appears only on copies sent to people on the Bcc: list and on the sender's display and file copies.

Type Describe Address-Fields|CR| for an explanation of <addressee-list>.

To continue the field to another line, type ,|CR| instead of |CR|. The comma is not inserted in the field. This subcommand adds to the current contents, if any, of the Bcc:-field.
-----------

Cc:
Header-Field; Subcommand of Create Draft

>>Cc: <addressee-list>|CR|

Cc: allows you to specify a list of addressees to whom "carbon copies" of the message are to be sent.

Type Describe Address-Fields|CR| for an explanation of <addressee-list>.

To continue the field to another line, type ,|CR| instead of |CR|. The comma is not inserted in the field. This subcommand adds to the current contents, if any, of the Cc:-field.

-----------

CDRAFT
Current Object

In this version of MAILSYS, there is always only one current Draft message, named CDRAFT, which is the message being created for sending.

Draft is the default object-type for the Create command, and CDRAFT is the default object for the Edit command. Type Describe DRAFT-EDITOR for information about creating and sending Draft messages.

------------

CFILTER Current-Object

The CFILTER is the default filter that my be used as part of the specification of a sequence.

The CFILTER comes into action only when the User types a sequence, followed by a slash /, followed ay an ESCAPE-key:

```
Ex.: >Print ALLMESSAGES/|ESC|CFILTER|CR|
or
Ex.: >Print|ESC|CSEQUENCE/|ESC|/CFILTER|CR|
```

However, if the USer default the <sequence> in a Print command, the CFILTER is not used:

```
Ex.: >Print|CR| = Print|ESC|CSEQUENCE|ESC|PTEMPLATE|ESC|TTY:|CR|
```

MAILSYS STandard Setting
CFILTER = Empty; passes any message

How to Change your CFILTER:

```
>Use CFILTER <filter>|CR|
 where filter may be
  a one-time, throwaway filter
  a ready-made filter
  a User-created, named filter
```

```
Ex.: >Use CFILTER From:Jones|CR|
     >Use CFILTER UNSEEN|CR|
     >Use CFILTER MYFILTER|CR|
```

-----------

Characters

|Control-A|
deletes a single character; does not delete back across unit parts of an addressee name in the To:, Cc:, or Bcc: fields, or a TENEX file designator in the Fcc:-field. Echoes as \ followed by the deleted character.

|Control-B| <TENEX file-name>
inputs text from a designated file while you are creating a message field; i.e., after you have given the command and

before you type the final |CR| (for a header-field) or
|Control-Z| (for the Text:-field). Does not echo.

|Control-C|
returns you to the TENEX Exec command level (prompt @).
Deleted messages are not expunged.

NOT RECOMMENDED. Type Exec|CR| to run the TENEX Executive in a lower fork if you want to continue in MAILSYS. Type Quit|CR| if you are finished with MAILSYS.

|Control-C| is safe to use only when MAILSYS has just issued the prompt character >, If |Control-C| is used at any other time during the operation of the system, the results of the command that is being executed cannot be guaranteed.

|Control-E|
aborts a command. Echoes as; ^E.

|Control-O|
Stops printout of text anywhere in the system. The command finishes its operation as if no interrupt had occurred. Echoes as ^O. |Control-O| is not equivalent to |Control-E|.

|Control-Q|
deletes a line of text in a text-string, or, used at the beginning of a line, deletes the previous line. Echoes as _ and a new line. Works the same way in To:, Cc:, Bcc:, and Fcc:-fields. In command strings, deletes the current part of a command.

|Control-R|
reprints the current line of text in a Header or Text:-field.

|Control-S|
reprints the text in the Text:-field that has been typed in since the last subcommand.

|Control-V|
preceding a character causes the character to be entered into a text string and not to have any control effect. Does not work with the interrupt characters |Control-C|, |Control-E| and |Control-O|. For example, if you are typing a Text:-field, |Control-Z| terminates the field. |Control-V| |Control-Z| causes the |Control-Z| to be entered as part of the text string, where it is shown as ^Z.

|Control-W|
deletes a single word in a text-string. Echoes as < followed by the first letter of the deleted word. Also deletes an entire addressee <name spec> or <TENEX file-name>.

|Control-Z|
acts as the terminator character for text in the Text:-field, and in the TEMPLATE EDITOR. Echoes as ^Z. (TELNET USERS: Change your escape character from |Control-Z| to something else before you use MAILSYS.)

Carriage Return |CR|
acts as the terminator for all HEADER fields (all message fields other than Text:). Causes execution of MAILSYS commands. In the Text:-field |CR| is the ordinary new-line character.

ALT-MODE-key (See ESCAPE-key |ESC|.)

DELETE-key |DEL| (Same as RUBOUT)
    deletes the current line in either text strings or commands.
    Echoes as XXX and a new line.  |DEL| is idempotent, i.e.,
    multiple |DEL|s have the same effect as one.

ESCAPE-key |ESC| (Same as ALT-MODE-key)
    triggers recognition and extension of MAILSYS commands, of
    names in the address-fields and of TENEX file-names.

LINEFEED-key |LF|
    changes the CURRENT MESSAGE to the next message in CSEQUENCE
    and prints it on your terminal.

Up-Arrow ^
    changes the CMESSAGE to the previous message in CSEQUENCE and
    prints it on your terminal.

RUBOUT-key (See DELETE-key |DEL|)

SPACE-bar character |SP|
    used to separate parts of MAILSYS commands in the abbreviated
    input mode.

QUESTION-MARK ?
    shows you the choices you have at any point in MAILSYS.

    >? gives a list of top-level commands.

    >SU? gives a list of commands that begin with SU.

    >SURVEY? gives a list of objects that can be use at the
    <sequence> position in a SURVEY command.  These include
    ready-made sequences, User-created sequences, if any, and
    CSEQUENCE.

    After listing your options, MAILSYS retypes the command and
    waits for you to type.  You can either complete the command
    with one of the words or characters listed, or you can abort
    the command with |DEL|.

    In a text string, ? is an ordinary character.  Do NOT use ?
    in a TENEX file-name; it will abort the type-in.

```
@  TENEX Exec prompt signal; precedes host names in
   addressee-lists.
>  MAILSYS prompt signal
*  = ALLMESSAGES in CMESSAGE-FILE.
%  = LASTMESSAGE in CMESSAGE-FILE.
/  Slash; separator between sequence and filter.
\  Backslash; printed out by |Control-A|.
.  Period or dot; symbol for CMESSAGE.
,  Comma; separator in some arguments.
:  Colon; in sequences 5:10 means 5 through 10; following an
   object, shows that object is a field of CDRAFT.
;  Semicolon; separator in sequences.  Top-level single-character
   command: introduces a single line of characters that are
   ignored by MAILSYS.  Used to "talk" over a TENEX Exec "Link".
-  Minus sign; to be used in later versions of MAILSYS.
+  Plus sign; also used in TEMPLATE-EDITOR to show that name of
   field is printed out.
```
-----------

Check-Inbox
Utility Command

>Check-Inbox|CR|

The command Get is a special case of Use:

```
>Get <TENEX file-name>|CR|
   = Use CMESSAGE-FILE <TENEX file-name>|CR|
Ex.: Get <SMITH>SPECIALFILE.MSG;1|CR|
```

GET-DIRECTORY SWITCH

1-ConnectDirectory: The initial CMESSAGE-FILE is MESSAGE.TXT;1 in the CONNECTED directory. (MAILSYS Standard Setting)

2-LoginDirectory: The initial CMESSAGE-FILE is MESSAGE.TXT;1 in the LOGIN directory.

For other switches that affect the CMESSAGE-FILE, type Describe Message-File or Describe SWITCHES.

----------

Compose

Draft-Creating Command; Also, Subcommand of Create Draft

```
>Compose|CR|
```

Compose gives you prompt-driven message-creation, using the subcommands To:, Cc:, Subject:, Text: and Send.

Differences from old Sndmsg:

(1) Header-fields (To:, Cc: and Subject:) terminate with |CR|. To continue the field to another line, type ,|CR| instead of |CR|. The comma is not inserted in the text string.

(2) The Text:-field terminates with a Control-Z (echoed as ^Z). The text string may be any length and may contain |CR|. TELNET USERS: Change the TELNET escape character from Control-Z to something else before you use MAILSYS.

(3) After you terminate the Text:-field, MAILSYS asks Send? Confirm with |CR| or Y|CR|, to send the message immediately. If you press |ESC|, MAILSYS prints Yes, and waits for you to confirm with |CR| or abort with |DEL|. Type N|CR| to return to Create-DRAFT subcommand level for future editing. You must then use the regular Send subcommand.

COMPOSE-SEND SWITCH (Compose subcommand overrides SWITCH.)

1-AskSend: Asks SEND? at the end of prompted message-composing sequence. Confirm with |CR| or Y|CR| to Send message immediately. If you press |ESC|, MAILSYS prints Yes, and waits for you to confirm with |CR| or abort with |DEL|. Type N|CR| to return to Create-Draft subcommand level for further editing. (MAILSYS Standard Setting)

2-Send: Immediate automatic sending.

3-NoSend: No automatic sending. Use the Send command.

-----------

Consider

Object-Handling Command

```
>Consider <sequence>|CR| = Use CSEQUENCE <sequence>|CR|
```

Consider changes the CSEQUENCE to a copy of <sequence>, where <sequence> may be either a named sequence or a literal typed-in sequence:

```
Ex.: >Consider RECENTMESSAGES|CR|
Ex.: >Consider 37:56,40-45/From: Smith|CR|
```

MAILSYS Standard Setting:

CSEQUENCE = ALLMESSAGES
-----------

Copy
Object-Handling Command

```
>Copy <object> (and name it) <name>|CR|
```

where <object> may be a sequence, filter or template, and <name> is any string of characters that does not contain a |SP|, |CR|, |ESC| or |LF|.

Copy copies the contents of a named object and saves it under a new name. The new name may be one you make up, or it may be CSEQUENCE, CFILTER, PTEMPLATE or STEMPLATE. However, you cannnot use Copy to changee the contents of a ready-made object.

If <name> is an existing object, MAILSYS tells you tht the old version will be replaced and asks you to confirm:

```
Ex.: >Copy STEMPATE (and name it) NEWTHING|CR|
     >Copy MYFILTER (and name it) CFILTER|CR| [Replace old version]
|CR|
```

Copy is also a subcommand of Edit Profle:

```
Ex.: >Edit Profile|CR|
     >>Copy MYTEMPLATE (and name it) TWOLINE|CR|
```

Both MYTEMPLATE and TWOLINE must be in the Profile. To copy filters and template into and out of the Profile, use the Import and Export commands.
-----------

Create
Object- and Draft-Creating and Editing Command

```
>Create <object type> <name>|CR|
```

Other forms of Create are:

```
>Create SEQUENCE <name>|CR|, calls the SEQUENCE-EDITOR.
>Create FILTER <name>|CR|, calls the FILTER-EDITOR.
>Create TEMPLATE <name>|CR|, calls the TEMPLATE-EDITOR.
```

If <name> is already the name of an existing object that can be edited, Create places a temporary copy of the object in the appropriate editor, and:

```
>Create <object-type> <name>|CR| = Edit <name>|CR|
```

If <name> is an object of the wrong object-type, Create refuses to accept it.

Draft is the default for Create:

```
>Create|CR| = Create Draft|CR| = Edit CDRAFT|CR|
```

calls the DRAFT-EDITOR and creates the current Draft message, named CDRAFT.

Create starts up the appropriate Object-Editor, running in a subcommand mode.

The subcommands may be used any number of times, in any order. Subcommands common to all Object-Editors are:

>>Done|CR| causes the temporary object become the current object,
and returns you to command level.

|Control-E| aborts the Create or Edit command, and destroys the
temporary object. Returns you to command level.

DELETE-key (|DEL| or |RUBOUT|) aborts a subcommand before you type
|CR|. Leaves you in the editor.

|Control-O| stops printout. Leaves you in the editor.

-----------

Current-Objects
One of Each Always Exists

If an Object is defaulted, i.e., not specified by name, in a
MAILSYS command, the system uses the current object.

| Object-Type | Current Object | MAILSYS Standard Setting |
|---|---|---|
| Draft | CDRAFT | [Empty] |
| Message-File | CMESSAGE-FILE | INBOX = <USER>MESSAGE.TXT;1 |
| Sequence | CSEQUENCE | ALLMESSAGES |
| Message | CMESSAGE "." | The message just before the first RECENT message. |
| Filter | CFILTER | [Empty]; passes any message |
| Template | PTEMPLATE | RCVD-FORM |
| | STEMPLATE | BASIC-SUMMARY |
| Destination | FDESTINATION | [Not Specified] |
| Switches | SWITCHES | First Column of SWITCHES display |

How to change a current object:

>Use (as) <current object> <object>|CR|
Ex.: >Use CFILTER UNSEEN|CR|
Changes the contents of the CFILTER to a copy of the ready-made
filter UNSEEN.
Ex.: >Use PTEMPLATE STANDARD|CR|

There are three other commands which are special cases of Use: >Get
<file-name>|CR|
= Use (as) CMESSAGE-FILE <file-name>|CR|
changes the CMESSAGE-FILE to <file-name>.
Ex.: >Get OLDMESSAGE.TXT;1
changes the CMESSAGE-FILE to the TENEX file OLDMESSAGES.TXT;1 in
the User's LOGIN directory.

>Consider <sequence>|CR|
= Use (as) CSEQUENCE <sequence>|CR|
changes the CSEQUENCE to a copy of <sequence>.
Ex.: >Consider RECENTMESSAGES|CR|
changes the contents of CSEQUENCE to a copy of RECENTMESSAGES.

>Jump-To <message-no.>|CR|
= Use (current object) CMESSAGE <message-no.>|CR|
= Use (as) . <message-no.>|CR|
changes the message-no. of the CMESSAGE to <message-no.> Ex.:
>Jump-To %|CR|
changes the CMESSAGE to the last message in your CMESSAGE-FILE.

The Copy command can be used to copy the contents of a named
sequence, filter or template into an appropriate current object:

Ex.: >Copy OLDMESSAGES (and name it) CSEQUENCE|CR| [Replace old
version.]

Ex.: >Copy STANDARD (and name it) FTEMPLATE [Replace old version.]

-----------

Date
Header-Field; Supplied by MAILSYS

When a message is sent, MAILSYS automatically supplies a Date-field in the form

Day Month Year Time-TimeZone

Ex.: 4 JUL 76 1453-EST

-----------

Daytime
Utility Command; Also TENEX Exec Command

>Daytime|CR|

prints the date and time on your terminal.

-----------

Delete
Message-Handling Command

>Delete <sequence>|CR|

marks the messages on for deletion. Deleted messages remain in the file, and may be selected with the DELETED filter.

Remove the Deleted markings with

>Undelete <sequence>|CR|

DELETE SWITCH
1-AskConfirm: MAILSYS prints a survey and asks you to confirm with a |CR| if, and only if, the message has NOT been SEEN during the current MAILSYS session. (MAILSYS Standard Setting)
2-NoAskConfirm: Does not request confirmation.

-----------

D = Single-character synonym for Delete

-----------

Describe
Utility Command

>Describe <topic> <destination>|CR|

prints out a short description of the topic named. Descriptions are available for all MAILSYS commands and subcommands, objects, control characters and other special characters, and for major features of MAILSYS.

----------

Destination
Object-type

A destination defines the place and manner in which messages are output.

In this version of MAILSYS, there is one current destination, named FDESTINATION, which is used with the File and Move commands only, and which must be a file-name.

MAILSYS Standard Initial setting:

FDESTINATION = Empty

A <destination> may be
TTY: prints messages on your terminal.
LPT: prints messages on the lineprinter.
<file-name> copies messages into a Tenex file.

How to change your FDESTINATION to a new Tenex file:

```
>Use (as current object) FDESTINATION (object) <file-name>|CR|
```

FDESTINATION may not be named or saved in your Profile.

----------

Directory
Utility Command; Also TENEX Exec Command

```
>Directory <file-list>|CR|
```

prints on your terminal the names of the files in <file-list>, where file-list is one or more TENEX file designations, separated by commas.

```
>Directory|CR|
```

prints a list of all the files in your directory.

----------

Done
Subcommand of Create and Edit

```
>>Done|CR|
```

ends subcommands of Create and Edit and returns you to the top command level. Also ends subcommands of Create and Edit within the PROFILE-EDITOR.

----------

Draft
Object-Type

In this version of MAILSYS, there is always only one current Draft message, named CDRAFT, which is the message being created for sending.

Draft is the default object-type for the Create command, and CDRAFT is the default object for the Edit command. Type Describe DRAFT-EDITOR for information about creating and sending Draft messages.

------------

DRAFT-EDITOR
Draft-Creating and Sending Commands

The Draft message currently being created for sending named CDRAFT. Call the DRAFT-EDITOR by giving one of the following commands:

```
>Create|CR|    or    >Edit CDRAFT|CR|
```

These two commands have exactly the same result; they put you in the subcommand mode, for creating and editing a Draft message.

If you want prompted composition, the command

```
>Compose|CR|
```

prompts for the same fields as old SNDMSG: To:, Cc:, Subject: and Text:.

The Text:-field terminates with |Control-Z|. After |Control-Z|, MAILSYS asks Send? If you type |CR| or Y|CR|, the message is sent immediately. If you type N|CR|, you are returned to the Create Draft subcommand level for further editing or for adding more fields.

```
>Forward <sequence>|CR|
>Forward |CR| = Forward CMESSAGE|CR|
```

copies the messages in <sequence> from your message-file into the Text:-field of the CDRAFT.

Forward fills in the Subject:-field with the authors and subjects of the forwarded messages, enclosed in square brackets, and then prompts for the To:-field and asks whether you want to add comments.

```
>Reply <message-no.>|CR|
>Reply|CR| = Reply CMESSAGE|CR|
```

helps you send replies to messages you receive.

Reply fills in the To:-field with the name of the Sender of the original message, and the Subject:-field with the subject of the original message, enclosed in parentheses, and prompts for the Text:-field.

## TYPING IN THE ADDRESS AND FCC: FIELDS

```
>><address-subcommand> <addressee-list>|CR|
```

where <addressee-list> is a series of one or more <name specs>, separated by commas, and <name spec> has one of the following forms:

<name>
    local user.

<name>(<string>)
    local user with an Attention Subfield.

<name> @ <host>
    user at some other ARPA host computer.

<name> @ <host>(<string>)
    user at some other host computer with an Attention Subfield.

@<host>, <name1>, <name2>, <name3>, ...
    several users at some other host computer.

@<host1>, <name1>, @, <name2>, @<host2>, <name3>(<string>)
    mixed list of local and remote users. Each name may be followed by an Attention Subfield. The host remains the same until another @ is inserted. An "@," means "at the local host."

In the Attention Subfield, <string> is any string of characters that does not contain a |SP|, |ESC|, |LF| or |CR|. Usually the Attention Subfield will contain names of people or groups, separated by commas.

GROUPNAMES

An <addressee-list> can be given a groupname:

```
<subcommand> <groupname>: <addressee-list>|CR|
```

where <groupname> is any string of characters that does not contain a space, |ESC|, |LF| or |CR|.

Only the groupname followed by : appears in the address-field. To show the complete <addressee-list> as typed, use the subcommand Show TypedForm. The <addressee-list> is not transmitted when the Draft message is sent, and so does not appear in, and cannot be recovered from, the received message.
----------

Draft-Fields

| Fields of CDRAFT | Subcommands of Create Draft |
|---|---|
| Rcvd-Date (Automatic) | --- |
| Date (Automatic) | --- |
| Sender (Automatic) | --- |
| Subject: | Subject: |
| From: | From: (Defaults to Sender) |
| To: | to: |
| Cc: | Cc: |
| Bcc: | Bcc: |
| Message-Id (Automatic) | --- |
| In-Reply-To: | In-Reply-To: |
| References: | References: |
| Keywords: | Keywords: |
| Precedence: | Precedence: |
| Message-Class: | Message-Class: |
| Special-Handling: | Special-Handling: |
| Fcc: | Fcc: |
| Other: (Header-fields from other message systems.) | --- |
| Text: | Text: |

----------

Edit
Object- and Draft-Editing Command

```
>Edit <name>|CR|
```

where <name> is either a current-object, such as CSEQUENCE or PTEMPLATE, or a User-created, named object. If you want to EDIT a ready-made object, first Copy it and give it a new name.

Possible Edit commands are

```
>Edit CDRAFT |CR| = Create Draft|CR| = Create|CR|
    calls the DRAFT-EDITOR
>Edit <sequence> |CR|, calls the SEQUENCE-EDITOR
>Edit <filter>|CR|, calls the FILTER-EDITOR|CR|
>Edit <template>|CR|, calls the TEMPLATE-EDITOR
>Edit SWITCHES|CR|, calls the SWITCHES-EDITOR
>Edit PROFILE|CR|, calls the PROFILE-EDITOR
```

Edit automatically places a temporary copy of the object in the appropriate editor and enters the subcommand mode.

The subcommands may be used any number of times, in any order. Type Describe SEQUENCE-EDITOR, Describe FILTER-EDITOR, etc., for details.

The subcommands common to all the editors are:

>>Done|CR| makes the modified temporary object become the current
    object.

|Control-E| aborts the Edit command at any point, and destroys
    the temporary object.  Returns you to command level.

|Control-O| stops printout at any point.  Leaves you in the
    editor.

DELETE-key (|DEL| or |RUBOUT|) aborts a subcommand at any point
    but lets you remain in the editor.
----------

Erase
Object-Handling command; Subcommand of Create and Edit

>Erase <object>|CR|

erases an entire User-created object, or the contents of a
Current Object, where a User-created object may be a named
sequence, filter or template, and where a current object may be
CDRAFT, CSEQUENCE, CFILTER, PTEMPLATE, STEMPLATE or FDESTINATION.

In the DRAFT-EDITOR:

>>Erase <field>|CR| erases the contents of a CDRAFT field.
>>Erase Header|CR| erases all fields except Text:
>>Erase ALL|CR| erases all CDRAFT fields.

----------

Exec
Utility Command

>Exec|CR|

creates an "inferior fork" of the TENEX Executive System, that
is, a copy of the TENEX Exec inside MAILSYS.

Return to MAILSYS by typing QUIT|CR| to the Exec.
----------

Exit
Utility Command

>Exit|CR|

terminates the operation of the message system and returns
control to the TENEX Executive System.

EXIT SWITCH (Quit and Exit are identical, but SWITCHES are
    independent.  Exit has subcommand that overrides SWITCH.)
1-AskExpunge:  Asks permission to expunge deleted messages and
    renumber old CMESSAGE-FILE when you Exit MAILSYS or Get a new
    CMESSAGE-FILE.  MAILSYS asks only when CMESSAGE-FILE contains
    deleted messages and you have permission to delete.  If you
    type |CR| or Y|CR|, MAILSYS expunges and renumbers.  If you
    press |ESC|, MAILSYS prints Yes and waits for you to confirm
    with |CR| or abort confirmation with |DEL|.  If you type
    N|CR|, MAILSYS does not expunge.  (MAILSYS Standard Setting)
2-Expunge:  Automatic expunging and renumbering.
3-NoExpunge:  No expunging and renumbering.  Use the Expunge

command.
----------

Expunge
Message-Handling Command

```
>Expunge|CR|
```

physically removes any deleted messages in a message file so that they cannot be Undeleted. The remaining messages are assigned new message-nos.
----------

Fcc:
Header-Field; Subcommand of Create Draft (or Edit CDRAFT)

```
>Fcc: <file-list>|CR|
```

where <file-list> is a list of TENEX file-names, separated by commas.

Fcc: sends a "file carbon copy" of your message to a list of designated TENEX files. The Fcc:-field appears only on the display and file copies.

To continue the field to another line, type ,|CR| instead of |CR|. The comma is not inserted in the field. This subcommand adds to the current contents, if any, of the Fcc:-field.
----------

FDESTINATION
Current-Object

In this version of MAILSYS, there is one current destination, named FDESTINATION, which is used with the File and Move commands only, and which must be a file-name.

MAILSYS Standard Initial setting:
FDESTINATION = None

How to Change your FDESTINATION to a new Tenex file:

```
>Use (as) FDESTINATION <TENEX file-name>|CR|
```

FDESTINATION may not be named or saved in your Profile.

----------

File
Message-Handling Command

```
>File <sequence> <destination>|CR|
```

File copies the messages listed in <sequence> into a TENEX file.

Ex.: >File 3,5,7 (on file) SMITH.MSG;1

File adds copies of messages 3, 5, and 7 to the Tenex file SMITH.MSG;1. The messages are not deleted from the CMESSAGE-FILE.

The <sequence> defaults to CSEQUENCE and <destination> defaults to the current FDESTINATION.

```
>File|CR| = File CSEQUENCE FDESTINATION|CR|
```

MAILSYS Standard Setting:
CSEQUENCE = ALLMESSAGES = *
FDESTINATION is not specified.

How to set FDESTINATION and change CSEQUENCE:

```
>Consider <sequence>|CR|
Ex.: Consider */SEEN|CR|
>Use FDESTINATION <destination>|CR|
Ex.: Use FDESTINATION OLDMESSAGES.TXT;1|CR|

Then >File|CR| = File */SEEN OLDMESSAGES.TXT;1|CR|
```

copies your SEEN messages into the file OLDMESSAGES.TXT.

```
    NOTE: File and Move are the only commands that produce files
    of MAILSYS-readable messages suitable for use as
    message-files.  If a file-name is used as the <destination>
    in a Print command, the resulting file will NOT be
    MAILSYS-readable, although the entire file may be added to
    any Draft-field with the command
    >Add-File (from file) <TENEX file-name> (to field)
    <field>|CR|
    Ex.: >Print 23,25 PTEMPLATE SMITHSTUFF.TXT;1|CR|
         >Add-File SMITHSTUFF.TXT;1 (to field) Text:|CR|
    or enterd winin a field with |Conrol-B|.
    Ex.: >Text:|CR|
          Here is the Smith file:
          |Control-B| (from file) SMITHSTUFF.TXT;1|CR|
          |Control-Z|
```

FILE SWITCH:  (File and Move are identical, but the SWITCHES are independent.  File has subcommand that overrides SWITCH.)
1-NoDelete:  Does not delete messages from CMESSAGE-FILE. (MAILSYS Standard Setting)
2-Delete:  Automatically deletes messages.
----------

Filters
Object-type

Filters are tools for selecting messages on the basis of their status or the information they contain.  A filter is always used as part of the specification of a sequence.

Ex.: Print ALLMESSAGES/From: Jones

ALLMESSAGES specifies a sequence; the slash "/" indicates that the next item is a filter.  Several sequences and filters of different types can be used in a single sequence specification. Type Describe Sequences for details.

There is always one current default filter, named CFILTER.

MAILSYS Standard Initial setting:
CFILTER = [Empty]

Possible filters are:

One-time, throwaway filters that you type in at the same time that you type in a sequence:
BEFORE <date>
AFTER <date>
ON <date> where <date> is the date on which the message was received; <date> may be entered as 4-JUL-76, 4 JUL 76, JULY 4, 1976 or 7/4/76; the month may be abbreviated or spelled out, upper- or lower-case.

<message header field> "<string>" where the string is a substring
    of the text of the header field, and is entered in quotes.
    If the string is a single word, it need not be quoted.

Ready-made, named filters:
NULLFILTER, passes any message.
SEEN, passes message marked SEEN.
UNSEEN, passes messages NOT marked SEEN.
DELETED, passes messages marked DELETED.
UNDELETED, passes messages NOT marked DELETED.
TODAY, passes messages received today.
YESTERDAY, passes messages received yesterday.

User-created, named filters:

How to Change your The Use command sets CFILTER to a copy of a named filter; CFILTER may be copied into a User-created filter with Copy. You can create your own named filters with Create FILTER <name>, and save them in your Profile.
-----------

FILTER-EDITOR

```
>Create Filter <name>|CR|
```

places a temporary copy of a new, empty filter named <name> in the FILTER-EDITOR. If a filter named <name> aleady exits, Create Filter has the same effect as

```
>Edit <name>|CR|
```

where <name> may be CFILTER or any Used-defined, named filter; places a temporary copy of <name> in the FILTER-EDITOR.

Subcommand options:

```
>>Show|CR|
```
    prints the contents of the temporary filter.

```
>>Require <attribute>|CR|
```
    the specified attribute is required to be true or present for the message to pass through the filter.

```
>>Reject <attribute>|CR|
```
    the specified attribute must be false or absent for the message to pass through the filter.

```
>>Ignore <attribute>|CR|
```
    the specified attribute is not considered for filtering purposes.

```
>>After <date>|CR|
```
    passes only messages received on or after the specified date.

```
>>Before <date>|CR|
```
    passes only messages received before the specified date.

```
>>On <date>|CR|
```
    passes only messages received on the specified date.

```
>>|Control-E|
```
    destroys the temporary filter, and does not change the contents of the pre-existing filter named <name>. Returns you to top command level.

```
>>Done|CR|
```

causes the temporary filter to become the new contents of the filter named <name>. Destroys the previous contents of name. Returns you to top command level.

Possible values for <attribute> are:

SEEN
    The message has been processed before by a template containing a Text:-field. The default is to ignore the fact that messages have been SEEN.

DELETED
    the message has been DELETED. The default is to ignore the fact that messages have been DELETED.

<address-field> <addressee-list>
    searches the addressee field for the occurrence of one of the addressees on the addressee-list.

    The <address-field> may be To:, Cc:, Bcc: or Sender. The <addressee-list> is a series of one or more <name specs>, separated by either spaces or commas. A <name Spec> has the form of <name>@<host>. Do not put a space before or after the "@". You may omit either <name> or <host>. MAILSYS does not check whether the <name spec> is a valid address.

<header field> <string list>
    searches the header field for the occurrence of one of the items on the string list.

    The <header field> may be any header except To:, Cc:, Bcc: or Sender. The <string list> is a series of one or more <string>s,separated by spaces or commas. A <string> may be any string of characters that does not contain a |CR|, |LF| or |ESC|; however, if the <string> contains a space or a comma, the entire <string> must be enclosed in quotation marks. The Show subcommand always displays <strings> enclosed in quotation marks.

    A <string> matches if it is a substring of the text of the header field, ignoring upper- and lower-case and spaces and|CR|s.

----------

Format
Subcommand of Create DRAFT

```
>>Format <field>|CR|
>>Format|CR| = Format Text:|CR|
```

Each line of the text is filled by bringing words from succeeding lines. The following conventions are used:

(1) Text must be single-spaced. A completely blank line indicates that the next line begins a paragraph. The formatter indents as many spaces (possibly none) as are indented in the text.

(2) A line containing a single |Control-I| (or tab) and no other characters, turns off the formatter for the following text. This allows user-formatted text to remain untouched. A line containing two |Control-I|'s, and no other characters, turns the formatter on again.

The formatter is idempotent, i.e., once used, the formatter can be called again with no change to the text (assuming you have not altered the text).

FORMAT SWITCH
1-NoJustify: Fills lines; spaces between words. (MAILSYS Standard Setting)
2-Justify: Automatically fills lines and evens up right-hand margins by inserting extra spaces between words.

----------

Forward
Message-Creating Command; Also Subcommand of Create DRAFT

```
>Forward <sequence>|CR|
>Forward|CR| = Forward CMESSAGE|CR|
```

copies the indicated message(s) into the Text: field, fills in the Subject:-field, and automatically calls the To: command. Type|CR| for subcommand

The Subject:-field has the form:

```
Subject: [AUTHOR1: SUBJECT1]
         [AUTHOR2: SUBJECT2], etc.
```

where AUTHOR1 is the author of the first message, SUBJECT2 is the subject of the second message, etc. One subject line per message being forwarded is shown. "..." after either the AUTHOR or SUBJECT indicates that the message has a multiple-line field.

FORWARD-SEND SWITCH (Forward subcommand overrides SWITCH.)
1-AskSend: Asks SEND? at the end of prompted message-composing sequence. Confirm with |CR| or Y|CR| to Send message immediately. If you press |ESC|, MAILSYS prints Yes, and waits for you to confirm with |CR| or abort with |DEL|. Type N|CR| to return to Create-Draft subcommand level for further editing. (MAILSYS Standard Setting)
2-Send: Immediate automatic sending.
3-NoSend: No automatic sending. Use the Send command.

----------

From:
Header-Field; Subcommand of Create DRAFT

```
>>From: <string>|CR|
```

where <string> is any string of characters that does not include |CR|.

The From:-field automatically defaults to the Sender field, unless you specifically fill in the From:-field.

To continue the field to another line, type ,|CR| instead of |CR|. The comma is not inserted in the field. This subcommand adds to the current contents, if any, of the From:-field.

FROM-NAME SWITCH
1-LoginDirectory: Fills in the From:-field with the name of the LOGIN Directory. (MAILSYS Standard Setting)
2-ConnectDirectory: Fills the From:-field with the name of the CONNECTED Directory.

----------

Get

```
>Get <TENEX file-name>|CR|
  = Use CMESSAGE-FILE <Tenex file-name>|CR|
```

During a MAILSYS session, the command Get changes the CMESSAGE-FILE to any other message-file.

MAILSYS Standard Setting:

```
CMESSAGE-FILE = MESSAGE.TXT;1 in the User's LOGIN directory
              = your INBOX
```

GET-DIRECTORY SWITCH
1-ConnectDirectory: The initial CMESSAGE-FILE is MESSAGE.TXT;1 in the CONNECTED directory. (MAILSYS Standard Setting)
2-LoginDirectory: The initial CMESSAGE-FILE is MESSAGE.TXT;1 in the LOGIN directory.

GET-EXPUNGE SWITCH
1-AskExpunge: Asks permission to expunge deleted messages and renumber old CMESSAGE-FILE when you Quit MAILSYS for Get a new CMESSAGE-FILE. MAILSYS asks only when old CMESSAGE-FILE contains deleted messages and you have permission to Delete. (MAILSYS Standard Setting)
2-Expunge: Automatic expunging and renumbering.
3-NoExpunge: No expunging and renumbering. Use the Expunge command.

----------

Groupnames
Used in Address-Fiedls To:, Cc: and Bcc:

An <addressee-list> can be given a groupname:

```
<address-subcommand> <groupname>: <addressee-list>|CR|
```

where <groupname> is any text string that does not contain a space.

Only the groupname followed by : appears in the address-field (To:, Cc: or Bcc:) of the CDRAFT. To show the complete <addressee-list> as typed, use the subcommand Show TypedForm. The <addressee-list> is not transmitted when the CDRAFT is sent, and so does not appear in the received message.
----------

Header-Fields
Subcommands of Create

Header-fields are the short fields displayed above the Text:-field of the message.

Type Describe Fields for a list of Header-Fields.

A Header-field is basically a one-line field, terminated by a |CR|. However, any Header-field may be continued to another line by typing a comma before the |CR|: that is, ",|CR|". The comma is not inserted in the field.

Any Header-subcommand adds to the previous contents, if any, of the Header-field.
----------

INBOX
Object; TENEX file

Your INBOX is the file in your TENEX file directory that receives all your incoming messages. In the ARPA Network, the INBOX is always named <USER>MESSAGE.TXT;1.

MAILSYS Standard Setting:
CMESSAGE-FILE = MESSAGE.TXT; in the User's LOGIN directory
              = your INBOX

GET-DIRECTORY SWITCH
1-ConnectDirectory: The initial CMESSAGE-FILE is MESSAGE.TXT;1 in the CONNECTED directory. (MAILSYS Standard Setting)
2-LoginDirectory: The initial CMESSAGE-FILE is MESSAGE.TXT;1 in the LOGIN directory.

For other switches that affect the INBOX, type Describe Message-File or Describe SWITCHES.
----------

Include
Subcommand of Create DRAFT; Creates a Text:-field

```
>>Include <sequence>|CR|
>>Include|CR| = Include CMESSAGE|CR|
```

adds a copy of the indicated message(s) in your CMESSAGE-FILE to the Text: field. The <sequence> defaults to the CMESSAGE.
----------

In-Reply-To:
Header-Field; Subcommand of Create DRAFT

```
>>In-Reply-To: <string>|CR|
```

where <string> is any string of characters that does not contain |CR|. This field is also automatically filled in by the Reply command.

To continue the field to another line, type ,|CR| instead of |CR|. The comma is not inserted in the field. This subcommand adds to the current contents, if any, of the In-Reply-To:-field.

----------

Jobstat
Utility Command; Also TENEX Exec Command

```
>Jobstat|CR|
```

prints on the User's terminal the TSS No. assigned, the name of the LOGN directory, and the terminal line.
----------

Jump-To
Message-Handling Command

```
>Jump-To <message-no.>|CR|
    = Use (current object) CMESSAGE <message-no.>|CR|
    = Use (current object) . <message-no.>|CR|
```

changes the message-no. of CMESSAGE to <message-no.>

MAILSYS Standard Setting:
CMESSAGE = the message just before the first RECENT message.

NOTE: CMESSAGE is also reset by the commands Print and |LPT|.
----------

Keywords:
Header-Field; Subcommand of Create DRAFT

```
>>Keywords: <string>|CR|
```

where <string> is any string of characters that does not contain |CR| (usually words separated by commas).

To continue the field to another line, type ,|CR| instead of |CR|. The comma is not inserted in the field. This subcommand adds to the current contents, if any, of the Keywords:-field.
----------

List
Message-Handling Command

```
>List <sequence> <destination>|CR|
>List|CR| = List CSEQUENCE LPT:|CR|
```

isa special command that produces a neatly packaged printout of a sequence of messages.

List prints an Index Sheet containing the name of the CMESSAGE-FILE, the date and time that the List was printed, and a set of Surveys of all the messages in <sequence>.

The complete printout of the messages follows. Each message starts a new page, and all pages have running head showing file-name, date, time and message-no.

The Surveys on the Index page are printed with the STEMPATE. The messages are printed with the PTEMPLATE.
----------

Mailer
Utility Command; Also Subcommand of Create DRAFT
Also TENEX Exec Command

```
>Mailer|CR|
```

forces immediate delivery of any messages queued by SEND, and prints out a status report of the TENEX MAILER subsystem queue.
----------

Mailstat
Utility Command; Subcommand of Create DRAFT
Also TENEX Exec Command

```
>Mailstat|CR|
```

lists queued and undeliverable messages. Allows you to manipulate undeliverable messages.
----------

Mark
Message-Handling Command

```
>Mark <sequence> <status>|CR|
>Mark <status>|CR| = Mark CMESSAGE <status>|CR|
```

changes the status of messages as recognized by MAILSYS, where status may be SEEN, UNSEEN, DELETED or UNDELETED.

```
>Mark <sequence> DELETED|CR| = Delete <sequence>|CR|
>Mark <sequence> UNDELETED|CR| = Undelete <sequence>|CR|
```

NOTE: Messages are automaticaly marked SEEN whenever they are processed with the RCVD-FORM template, the STANDARD template, or any other template that contains a TEXT:-field.

----------

Message-Class:
Header-Field; Subcommand of Create DRAFT

```
>>Message-Class <string>|CR|
```

where <string> is any string of characters that does not contain|CR|.

To continue the field to another line, type ,|CR| instead of |CR|. The comma is not inserted in the field. This subcommand adds to the current contents, if any, of the Message-Class:-field.

----------

Message File
Object-Type

A Message-File is a file which contains MAILSYS-readable messages. Each message in the Message-File is identified by a message-no.

There is always one current Message-File, named CMESSAGE-FILE, on which MAILSYS operates.

```
MAILSYS Standard Initial setting:
CMESSAGE-FILE = MESSAGE.TXT;1 in your directory.
              = your INBOX
```

```
How to Change the CMESSAGE-FILE:
>Get <TENEX file-name>|CR|
 = Use (current object) CMESSAGE-FILE <TENEX file-name>|CR|
```

```
INITIAL-SURVEY SWITCH
1-Initial Survey:  At the beginning of a MAILSYS session, and
    whenever the User Gets a new CMESSAGE-FILE, MAILSYS prints an
    automatic SURVEY or RECENT messages.  (MAILSYS Standard
    Setting)
2-No Initial Survey:  Does not print out an initial survey.
```

```
REPORTNEWMSG SWITCH
1-Periodic&AtPrompt:  Messages that arrive during a MAILSYS
    session are marked NEW and reported after a fixed interval,
    at the next MAILSYS prompt, or periodically without a prompt,
    whichever comes first.  (MAILSYS Standard Setting)
2-AtPrompt:  NEW messages are reported, after a fixed interval,
    at the next MAILSYS prompt.
3-DelayReport:  NEW messages are reported only when you leave the
    CMESSAGE-FILE or give an Expunge command.
```

```
REPORTFORM SWITCH
1-SurveyReport:  NEW messages are reported by a bulletin:  "00
    NEW MESSAGES HAVE JUST ARRIVED" and a survey of each message.
    (MAILSYS Standard Setting)
2-BulletinReport:  New messages are reported by the bulletin
    only.
3-NoReport:  New messages are not reported in MAILSYS.
```

GET-DIRECTORY SWITCH
1-ConnectDirectory: The initial CMESSAGE-FILE is MESSAGE.TXT;1
    in the CONNECTED directory. (MAILSYS Standard Setting)
2-LoginDirectory: The initial CMESSAGE-FILE is MESSAGE.TXT;1 in
    the LOGIN directory.

GET-EXPUNGE SWITCH
1-AskExpunge: Asks permission to expunge deleted messages and
    renumber old CMESSAGE-FILE when you Quit MAILSYS for Get a
    new CMESSAGE-FILE. MAILSYS asks only when old CMESSAGE-FILE
    contains deleted messages and you have permission to Delete.
    (MAILSYS Standard Setting)
2-Expunge: Automatic expunging and renumbering.
3-NoExpunge: No expunging and renumbering. Use the Expunge
    command.

----------

MESSAGE-HANDLING COMMANDS

When you enter MAILSYS, the system automatically prints out one-line SURVEYS of all RECENT messages, i.e., all messages that have arrived since you last entered the message-file. The current message, named CMESSAGE, is set to the message before the first RECENT message.

All you have to do to print out your RECENT messages, one at a time, on your terminal, is to press the LINEFEED-key |LF|. The |LF| command changes the CMESSAGE to the "next" message, and then prints it.

Note: The "next" message is the next message in the current sequence of messages, which is named CSEQUENCE. When you enter MAILSYS, CSEQUENCE is set to be ALLMESSAGES = all the messages in the CMESSAGE-FILE.

-----
COMMANDS THAT DEFAULT TO CMESSAGE

The Print command defaults to the CMESSAGE, that is,

>Print|CR| prints CMESSAGE on the User's terminal.

If you have printed a message with |LF|, the command Print|CR| will print the same message a second time.

A convenient feature is that

>Print <message-no.>|CR| also sets CMESSAGE to <message-no.>

Other commands that default to CMESSAGE are Print, Delete, File, Move, Mark, Forward and Reply. Thus you can pick out a message, print it on your terminal and react to it with the help of a group of message handling commands that default to the message you have just printed.

How to Change CMESSAGE:
>Jump-To <message.no.>|CR|
   = Use CMESSAGE <message-no.>|CR|

-----
COMMANDS THAT DEFAULT TO SEQUENCES

The message-handling commands that default to CSEQUENCE rather

than CMESSAGE are those most frequently used for handling groups
of messages.

```
>Survey|CR|   prints one-line Surveys of all messages in
                  CSEQUENCE on the User's terminal.

>List|CR|     prints all messages in CSEQUENCE on the
                  lineprinter, one message to a page,
                  preceded by a page of Surveys.
```

The Summarize command is the same as Survey, except that it
defaults to RECENTMESSAGES.

```
>Summarize|CR| prints one-line Survey of all RECENT
                  messages on the User's terminal.
```

How to change CSEQUENCE:

```
>Consider <sequence>|CR|
   = Use CSEQUENCE <sequence>|CR|
```

See next section to learn how to specify sequences.

-----
USING SEQUENCES OF MESSAGES

It is always possible to type in a <sequence> explicitly, instead
of allowing the command to default to CMESSAGE or CSEQUENCE.  A
<sequence> is always the first argument in a message-printing
command such as Print.

```
>Print <sequence>|CR|
```

Ex.: Print 6:10|CR| prints messages 6 through 10.
Ex.: Print ALLMESSAGES|CR| prints all messages in CMESSAGE-FILE.

-----
TYPING IN SEQUENCES

A Sequence is an ordered set of message-nos., defined as the
logical union of any number of individual sets. The sets may be
defined in terms of message-nos., or named Sequences (created
either by the system or by the User), or Sequences acted upon by
Filters.

A Sequence may be typed as a series of one or more <groups>,
separated by semicolons:

```
    <group>;...;<group>
```

A <group> may be typed as a series of one or more <subgroups>,
separated by commas, and followed by, and acted upon by, a series
of one or more filters. The filters are separated from each
other, and from the final <subgroup>, by slashes:

```
<subgroup>,...,<subgroup>/<filter>/.../<filter>
```

where a <subgroup> may be:
A message-no. or symbol for a message-no.:

```
    Ex.: 3, 5
         % = last message in CMESSAGE-FILE
         .  = the current message CMESSAGE
```

a range of numbers, or a range symbol.

```
    Ex.: 1:3 = 1,2,3
```

```
        9:5 = 9,8,7,6,5
        6:% 3:.  .:%
        * = 1:% = all messages in the CMESSAGE-FILE
CSEQUENCE = the current Sequence
```

A ready-made sequence, created by MAILSYS:

    ALLMESSAGES = *, all the messages in CMESSAGEFILE.
    NEWMESSAGES, arrived since you entered CMESSAGE-FILE
      (either by starting MAILSYS or by using the GET command to
      enter a message-file other than your Inbox).
    RECENTMESSAGES, arrived since the previous time you entered
    CMESSAGE-FILE.
    OLDMESSAGES, arrived before you last entered CMESSAGE-FILE.
    NULLSEQUENCE, no messages, an empty sequence.

A named sequence that you create with the SEQUENCE-EDITOR.
(Type Describe SEQUENCE-EDITOR for details.)

NOTE:

(a) DO NOT place a space |SP| before any semicolon, comma, or slash in the middle of a Sequence.

(b) If none of the groups contain Filters, the semicolon and the comma may be used interchangeably.

(c) In real life, you probably won't ever type a command more than a line long, but if you must, DO NOT put a carriage return |CR| at the end of the line. Let your terminal give you a runover line.

-----

HOW TO USE FILTERS TO SEARCH YOUR MESSAGE-FILE

MAILSYS always provides a current filter named CFILTER, but when you first enter MAILSYS, CFILTER is set to [Empty: passes any message].

How to change CFILTER:

```
>Use CFILTER (object)<filter>|CR|
```

where <filter> can be either a typed-in filter or the name of an existing filter.

You can search for information in messages by using non-empty filters.

A filter is always typed in or designated as part of a sequence.

```
>Survey ALLMESSAGES/UNSEEN|CR|
```

applies the filter "UNSEEN" to all the messges in your message-file and prints Surveys of the messages that have not been marked SEEN. The slash "/" shows that the next word or phrase is a filter.

Note: Do NOT type a space |SP| before or after a slash.

You can default to the CFILTER by pressing the ESCAPE-key |ESC| after the slash:

```
>Survey ALLMESSAGES/|ESC|CFILTER|CR|
```

If you want to search on the contents of a field, type the first two or three letters of the name of the field, press |ESC|, and allow MAILSYS to print the rest of the field name. The field name will include a colon and a space.

```
>Print */FR|ESC|om:
```

Note: Do NOT type a space |SP| after the colon; let MAILSYS do it for you.

Now type in the search string and a carriage return |CR|.

```
>Print */From: Baskerville|CR|
```

You can use two filters in a row:

```
>Survey */From: Baskerville/Subject: Hound|CR|
```

This command prints one-line surveys of all messages that have "Baskerville" in the From:-field and the word "hound" somewhere in the Subject:-field. The search ignores whether the letters are upper- or lower-case.

If you want to seach for a subject of two or more words, you must put the words in quotes:

```
>Print */To: Moriarity/Subject: "speckled band"|CR|
```

You can also use dates:

```
>Survey */UNSEEN/After: 1-Jan-76/Before: July 4, 1976|CR|
```

-----
THROWAWAY FILTERS

A one-time, throwaway filter that you type in at the same time as a sequence can have one of the following forms:

Before <date>
After <date>
On <date> where <date> is the date on which the message was
    received; <date> may be entered as 4-JUL-76, 4 JUL 76, JULY
    4, 1976 or 7/4/76; the month may be abbreviated or spelled
    out, upper- or lower-case.
<message header field> "<string>" where the string is a substring
    of the text of the header field, and is entered in quotes.
    If the string is a single word, it need not be quoted.

You can also use existing filters that have names:

CFILTER = the current filter

A ready-made filter, created by MAILSYS:
NULLFILTER, passes any message.
SEEN, passes message marked SEEN.
UNSEEN, passes messages NOT marked SEEN.
DELETED, passes messages marked DELETED.
UNDELETED, passes messages NOT marked DELETED.
TODAY, passes messages received today.
YESTERDAY, passes messages received yesterday.

A named filter that you create with the FILTER-EDITOR. Named filters can be stored in your Profile. (Type Describe FILTER-EDITOR and PROFILE-EDITOR for details.)

-----
THE FORM OF PRINTED MESSAGES

The commands that print messages in MAILSYS are Survey, Summarize, Print, |LF|, ^, and List. The exact form in which messages are printed out is controlled by objects called templates.

The templates is always the second argument of the Print command.

```
>Print <sequence><template>|CR|
```

MAILSYS provides two current, or default templates:

```
STEMPLATE   Always used by Survey, Summary and the Index page
                  produced by the List command.

PTEMPLATE   Always used by |LF|, ^ and the messages
                  printed by the List command.
```

How to change either the STEMPLATE or the PTEMPLATE:

```
>USE (current object) STEMPLATE <template>|CR|
>USE (current object) PTEMPLATE <template>|CR|
```

MAILSYS provides a set of ready-made templates, and you can also construct templates of your own, tailored to suit you individual needs and taste. Type Describe TEMPLATE-EDITOR for details.

FILES AND PRINTERS

The complete form of the print command is:

```
>Print <sequence><templates><destination>|CR|
```

Print defaults to the User's terminal so that

```
>Print |CR| = Print CMESSAGE PTEMPLATE TTY:|CR|
```

```
TTY:  =  User's terminal, always the destination for |LF| and ^,
         default destination for Print.

LPT:  Line printer, default destination for LIST
```

You can set up a current destination named FDESTINATION for the File and Move commands only.

```
>USE (current object) FDESTINATION (object)<TENEX file-name>
```

-----
OTHER FEATURES

The SEQUENCE-EDITOR allows you to name and store sequences of message-nos. for the length of a MAILSYS session. With the SEQUENCE-EDITOR, you can combine sequences according to the basic logical operations and sort them according to information in the header fields, or according to attributes such as date received or length.

The SWITCHES feature gives you a choice of options as to how many of the command and other features of MAILSYS work. You can change your SWITCHES settings with the SWITCHES-EDITOR.

The MAILSYS Profile, which can be changed with the PROFILE-EDITOR, allows you to select a set of filters, templates and SWITCHES settings that are most useful and comfortable for you. At the same time, you can override any individual setting in the Profile by an explicit command or subcommand. Thus the
[illegible]

Profile is always a set of defaults, and not of rigid
constraints.

----------

Message-ID:
Header-Field; Supplied by MAILSYS

When a message is sent, MAILSYS automatically supplies a
Message-ID field in the form

<[Host] Date.Name>

```
 where
  Host = Offical ARPA Host Name
  Date = Day-Month-Year
  Hour:Minute:Second-Timezone
  Name = LOGIN Directory of User.
```

Ex.: <[BBNA]4-Jul-76 116:10:08-EST.SOMEBODY>

----------

Message-No.

When a message is added to a message-file, MAILSYS automaically
supplies a Message-No.-field and assigns the message the next
serial number in the message-file.

The Delete command marks a message for deletion but does not
change the Message-No. However, when the message-file is
expunged, either by the use of the Expunge command, or when the
User leaves the message by the use of the Quit, Exit or Get
command, the DELETED messages are phycially removed from the file
and the remaining messages are given new Message-Nos.

An ordered set of message-nos. forms a Sequence.

There is always one current message-no., named CMESSAGE, and
symbolized by the period or dot "."

MAILSYS Standard Settings:
CSEQUENCE = ALLMESSAGES
CMESSAGE is set just before the first RECENT message.

This convention is adopted so that the command LINEFEED |LF| can
be used to print out the first recent message.

>|LF| LINEFEED changes the CMESSAGE to the next message on
    CSEQUENCE, and then prints the new CMESSAGE on the your
    terminal, using the PTEMPLATE.

>^ UP-ARROW is the "reverse" of |LF|; changes the CMESSAGE to the
    previous message on CSEQUENCE, and then prints it.

>Print <sequence>|CR|
    changes the CMESSAGE to the last message-no. in <sequence>,
    provided that the message-no. is within the CSEQUENCE.
    Since CSEQUENCE is initially set to ALLMESSAGES, the
    message-no. is always a part of the CSEQUENCE unless the
    User has given a Consider command.

The CMESSAGE can be entered symbolically as "."

>Show CMESSAGE|CR|
    = Show . |CR|
    prints the message-no. of the CMESSAGE on your terminal.

```
>Use CMESSAGE <message-no.>|CR|
    = Use . <message-no.>|CR|
    = Jump-To <message-no.>|CR|
    changes the message-no. of the CMESSAGE to <message-no.>.
```
----------

Move
Message-Handling Command

```
>Move <sequence> <destination>|CR|
```

Move copies the messages listed in <sequence> into a TENEX file.

```
Ex.: >Move 3,5,7 (on file) SMITH.MSG;1
```

Move adds copies of messages 3, 5, and 7 to the Tenex file SMITH.MSG;1. The messages are not deleted from the CMESSAGE-FILE.

The <sequence> defaults to CSEQUENCE and <destination> defaults to the current FDESTINATION.

```
>Move|CR| = Move CSEQUENCE FDESTINATION|CR|
```

MAILSYS Standard Setting:
CSEQUENCE = ALLMESSAGES = *
FDESTINATION is not specified.

How to set FDESTINATION and change CSEQUENCE:

```
>Consider <sequence>|CR|
>Use FDESTINATION <destination>|CR|

Ex.: >Consider */SEEN|CR|
     >Use FDESTINATION OLDMESSAGES.TXT;1|CR|
Then >Move|CR| = Move */SEEN OLDMESSAGES.TXT;1|CR|
```
copies your SEEN messages into the file OLDMESSAGES.TXT.

NOTE: File and Move are the only commands that produce files of MAILSYS-readable messages suitable for use as message-files. If a file-name is used as the <destination> in a Print command, the resulting file will NOT be MAILSYS-readable, although the entire file may be added to any Draft-field with the command

```
>Add-File (from file) <TENEX file-name> (to field)
<field>|CR|
Ex.: >Print 23,25 PTEMPLATE SMITHSTUFF.TXT;1|CR|
     >Add-File SMITHSTUFF.TXT;1 (to field) Text:|CR|
```
or entered wihin a field with |Control-B|.

```
Ex.: >Text:|CR|
      Here is the Smith file:
      |Control-B| (from file) SMITHSTUFF.TXT;1|CR|
      |Control-Z|
```

MOVE SWITCH: (File and Move are identical, but the SWITCHES are independent. Move has subcommand that overrides SWITCH.)
1-NoDelete: Does not delete messages from CMESSAGE-FILE. (MAILSYS Standard Setting)
2-Delete: Automatically deletes messages.

----------

Neted
Text Editor; Subcommand of Create Draft

Teco, Neted, and Xed are the three text editors available for editing message fields. To call Neted, type

```
>>Neted <field>|CR|
>>Neted|CR| = Neted Text:|CR|
```

Neted creates an "inferior fork" of TENEX Neted, that is, a copy of Neted inside MAILSYS, with the specified field of the message in the Neted buffer. You may then use all of the available Neted commands to edit the field.

After editing, return to MAILSYS by typing Quit|CR| to the Neted program. MAILSYS prints out OK?

If you respond to OK? by typing |CR| or Y |CR|, MAILSYS inserts the edited version of the buffer as the new message field. If you type N |CR|, the edited version of the buffer disappears, and MAILSYS does not change the contents of the message field. In either case, you are returned to the MAILSYS Create-Draft subcommand level.

You can also return to MAILSYS by typing SAVE<file-name>|CR|. This command creates a TENEX file containing copy of the buffer, then executes Quit.

----------

News
Utility Command

```
>News|CR|
```

News prints out a list of changes made to MAILSYS since the last version of the system.

----------

Next

```
>Next|CR| = |LF|
```

The command Next|CR| has the same effect as the single-character command LINEFEED-key |LF|. Next (or |LF|) and UP-ARROW ^ allow you to "browse" forward and back through CSEQUENCE, using PTEMPLATE and printing messages one by one on your terminal. These commands cannot be directly mimicked by PRINT, because they allow you to step through the CSEQUENCE automatically, printing one message at a time.

>Next|CR| changes the CMESSAGE to the next message on the
    CSEQUENCE and then prints the new CMESSAGE on your terminal,
    using the PTEMPLATE.

>^ is the "reverse" of Next|CR|, changes the CMESSAGE to the
    preceding message on CSEQUENCE, and then prints it.

----------

Precedence:
Header-Field; Subcommand of Create DRAFT

```
>>Precedence: <string>|CR|
```

where <string> is any string of characters that does not contain|CR|.

To continue the field to another line, type ,|CR| instead of |CR|. The comma is not inserted in the field. This subcommand adds to the current contents, if any, of the Precedence:-field.

----------

Print
Message-Handling Command

```
>Print <sequence> <template> <destination>|CR|
 Print |CR| = Print CMESSAGE PTEMPLATE TTY: |CR|
```

where <destination> defaults to TTY: NOSEPARATE and prints the messages on your terminal without pausing between messages.

----------

PROFILE-EDITOR

```
>Edit Profile|CR|
```

places a temporary copy of the Profile in the PROFILE-EDITOR, and enters the subcommand mode.

Subcommand Options:

```
>>Show|CR|
    prints temporary contents of PROFILE.

>>IMPORT <object> (and name it) <name>
    copies a named object from the ACTIVE ENVIRONMENT and stores
    it in the Profile under <name>.

>>EXPORT <object> (and name it) <name>|CR|
    copies a named object from the Profile and places it in the
    ACTIVE ENVIRONMENT under <name>.

>>Copy <object> (and name it) <name>|CR|
    copies an object that is already in the Profile and stores it
    in the Profile under <name>.

>>Erase <name>|CR|
    erases a User-defined named object from the Profile.

>>Use <current object> <name>|CR|
    changes the contents of any current object in your PROFILE to
    a copy of the named object.

where <current object> may be
    CMESSAGE-FILE, takes a TENEX file-name only
    CFILTER, takes named filter only.
    PTEMPLATE, takes named template only.
    STEMPLATE, takes named template only.

>>Create <object> (and name it) <name>|CR|

>>Edit <object>|CR|
    where <object> may be a named filter or template, the
    SWITCHES, or the MESSAGE-FILE-RECORD.

>>|Control-E|
    Destroys the temporary Profile, and does not change the
    contents of the pre-existing Profile.  Returns you to command
    level.

>>Done|CR|
    Causes the temporary Profile to become the new contents of
    your Profle.  Destroys the previous contents of your Profile.
    Returns you to top command level.
```

----------

PTEMPLATE
Current-Object

The PTEMPLATE is the default template for the Print command, and the template always used by |LF| and ^. Only one PTEMPLATE exists at a time.

MAILSYS Standard Setting:
PTEMPLATE = RCVD-FORM

How to Change PTEMPLATE:
```
>Use PTEMPLATE <template>|CR|
```

where <template> is any named template; Use changes the contents of the PTEMPLATE to a copy of <template>.

```
Ex.: >Use PTEMPATE STANDARD|CR|
     >Use PTEMPLATE MYTEMPLATE|CR|
```

```
>Copy PTEMPLATE (and name it) <name>|CR|
```

copies the contents of PTEMPLATE into a User-defined templatei named <name>.
```
Ex.: >Copy PTEMPLATE (and name it) MYTEMPLATE|CR|
```

```
>Copy <template> (and name it) PTEMPLATE|CR|
```

copies the contents of any named template into PTEMPLATE.
```
Ex.: >Copy MYTEMPLATE (and name it) PTEMPLATE|CR|
```

How to save your choice of PTEMPLATE in your Profile:
```
>Edit Profile|CR|
>>Import MYTEMPLATE|CR|
>>Use PTEMPLATE MYTEMPLATE|CR|
>>Done|CR|
```
----------

QUESTION-MARK
Shows you the choices you have at any point in MAILSYS.

```
>?         gives a list of top-level commands.

>SU?       gives a list of commands that begin with SU.

>SURVEY ? gives a list of objects that can be use at the
           <sequence> position in a SURVEY command.  These include
           ready-made sequences, User-created sequences, if any,
           and CSEQUENCE.
```

After listing your options, MAILSYS reprints the command and waits for you to type. You can either complete the command with one of the words or characters listed, or you can abort the command with |DEL|.

In a text string, ? is an ordinary character. Do NOT use ? in a TENEX file-name; it will abort the type-in.
-----------

QFD
Utility Command; Also TENEX Exec Command

```
>QFD <file-list>|CR|
```

prints on your terminal all TENEX directory information about

<file list>, where <file list> is one or more TENEX file
dessignations, separated by commas.

>QFD|CR|

prints all TENEX directory information about all the files in
your directory.

This command prints the equivalent of the TENEX command Directory
with subcommands EVERYTHING, CRAM, and NO (HEADING).

----------

Quit
Utility Command

>Quit <CR>

terminates the operation of the message system and returns
control to the TENEX Executive System.

```
QUIT SWITCH (Quit and Exit are identical, but SWITCHES are
    independent.  Quit has subcommand that overrides SWITCH).
1-AskExpunge:  Asks permission to expunge deleted messages and
    renumber old CMESSAGE-FILE when you Quit MAILSYS or Get a new
    CMESSAGE-FILE.  MAILSYS asks only when CMESSAGE-FILE contains
    deleted messages and you have permission to delete.  (MAILSYS
    Standard Setting)
2-Expunge:  Automatic expunging and renumbering.  If you type
    |CR| or Y|CR|, MAILSYS expunges and renumbers.  If you press
    |ESC|, MAILSYS prints Yes and waits for you to confirm with
    |CR| or abort confirmation with |DEL|.  If you type N|CR|,
    MAILSYS does not expunge.
3-NoExpunge:  No expunging and renumbering.  Use the Expunge
    command.
```

----------

Q = Single-character synonym for Quit

----------

Rcvd-Date
Header-Field; Supplied by MAILSYS

When a message is received, MAILSYS automaticaly supplies a
Rcvd-Date-Field that shows name of Sender's host computer, and
date and time that the message was received by the addressee.

Ex.: Mail from BBN-TENEXA rcvd at 4-Jul-76 1201-EST.

----------

Reference:
Header-Field; Subcommand of Create DRAFT

>>Reference: <string>|CR|

where <string> is any string of characters that does not
contain|CR|.

To continue the field to another line, type ,|CR| instead of
|CR|.  The comma is not inserted in the field.  This subcommand
adds to the current contents, if any, of the References:-field.

----------

Reply
Draft-Creating Command; Also Subcommand of Create DRAFT

>Reply <Message-No.>|CR|
>Reply |CR| = Reply CMESSAGE|CR|

fills in the To:, Subject:, and In-Reply-To:-fields with information from the indicated message in your CMESSAGE-FILE, and calls the Text: field.

The To:-field contains the name of the SENDER of the original message. The Subject:-field contains the original Subject: enclosed in parentheses ( ). The In-Reply-To:-field contains the original MESSAGE-ID.

REPLY-SEND SWITCH (Reply subcommand overrides SWITCH.)
1-AskSend: Asks SEND? at the end of prompted message-composing sequence. Confirm with |CR| or Y|CR| to Send message immediately. If you press |ESC|. MAILSYS prints Yes, and waits for you to confirm with |CR| or abort with |DEL|. Type N|CR| to return to Create-Draft subcommand level for further editing. (MAILSYS Standard Setting)
2-Send: Immediate automatic sending.
3-NoSend: No automatic sending. Use the Send command.

REPLY-COPIES SWITCH (Reply subcommand overrides SWITCH.)
1-NoCopies: Sends reply only to Sender of original message. (MAILSYS Standard Setting)
2-AskCopies: If you type |CR| or Y|CR|, MAILSYS send copies of reply to all names in the To: and Cc: fields. If you press |ESC|, MAILSYS prints Yes, and waits for you to confirm with |CR| or abort with |DEL|. If you type N|CR|, MAILSYS sends reply only to original sender.
3-Copies: Sends copies of reply to all names in To: and Cc: fields of original message.

----------

Restore-Draft
Subcommand of CREATE Draft

>Restore-Draft (from file) <TENEX file-name>|CR|

can be used only with a TENEX file created by Save-Field ALL, Save-Field Headers, or Store-Draft. This command appends the contents of each field of the stored draft to the corresponding field of CDRAFT.

----------

Retrieve
Message-Handling Command

>Retrieve <Message-ID field of archived message>|CR|

causes a message to be sent to the DATACOMPUTER requesting the retrieval of a message previously archived.

----------

Run
Utility Command; Also TENEX Exec Command

>Run <TENEX file name>|CR|

Run allows you to load and run a TENEX "Save Core" File without leaving MAILSYS or using the Exec command to start up a lower fork of the TENEX Exec program.

The MAILSYS command Run works like the TENEX Exec command Run, with the following exceptions*

(1) The TENEX command Run is the default command for the TENEX Executive System. MAILSYS has no default command.

(2) The TENEX Exectutive System as a set of priorities for looking in different directories in the system for any file-name that is typed without a directory name. MAILSYS looks only in the User's CONNECTED directory.

Ex.: >Run MYPROGRAM.SAV;3|CR|
where MYPROGRAM is a program in the User's directory. MAILSYS executes the program. However, the command
    >Run TECO.SAV |ESC| fails because the TECO.SAV file in the the TENEX <SUBSYS> directory. To execute the Teco program using the Run command, it is necessary to type in the directory name:
    >Run <SUBSYS>TECO.SAV|ESC|

----------

Save-Field
Subcommand of Create DRAFT; Files a CDRAFT or Draft-field

```
>>Save-Field <All, Headers or field> (on file)
    <TENEX file-name>|CR|
```

Save-Field makes it possible to save the contents of the entire CDRAFT, the Header-fields only, or a single field of the CDRAFT in a standard TENEX file.

To bring back an entire Draft, or a set of Header-fields, that have been stored with Store-Draft into the CDRAFT, use the command

```
>Restore-Draft (from file) <TENEX file-name>|CR|
```

Restore-Draft appends the fields stored in the TENEX file to the corresponding fields of the CDRAFT.

To append a single field to a field of the CDRAFT, use the command

```
>Add-File <TENEX file-name> (to field) <field>|CR|
```

----------

Send
Subcommand of Create DRAFT

```
>>Send|CR| [CONFIRM]|CR|
```

sends out the message, and prints out "SENDING...DONE" when the message is placed in a queue for the TENEX MAILER subsystem to pick up and try to deliver.

If the message cannot be delivered, MAILER sends you a message and tells you why.

SEND-ERASE SWITCH (Send has subcommand that overrides SWITCH.)
1-Erase: Automatically erases DRAFT Fields after sending.
    (MAILSYS Standard Setting)
2-NoErase: DRAFT Fields are not erased.

SEND-ARCHIVE SWITCH -- ARPA Network Archiving Facility
    (Send has subcommand that overrides SWITCH.)
1-NoArchive: Outgoing messages are not archived in the ARPA
    Network DATACOMPUTER. (MAILSYS Standard Setting)
2-Archive: Outgoing messages are automatically archived in the

ARPA Network DATACOMPUTER for permanent storage.

The DATACOMPUTER is an experimental, public-access information facility whose entire contents is accessible to all users of the ARPANET. Messages Archived in the DATACOMPUTER can be retrieved with the command Retrieve.

WARNING!!! The Archive subcommand has no relationship to the Archive facility that may be implemented as part of your local TENEX system.

----------

Sender
Header-field; Supplied by MAILSYS

The Sender Field contains the ARPA Network Mail Address of the User's LOGIN Directory in the form

<Name> @ <Host>

The From:-field automatically defaults to the Sender field unless you specifically fill in the From:-field.

----------

Sequences
Object-Type

Sequences tell which messages are to be processed.

A sequence is a list of message numbers which specify the messges to be processed. A sequence can contain one or more filers which select messages according to (a) information in the Header fields, or (b) the status of the message, e.g., SEEN or RECENT. (See below, TYPING IN SEQUENCES.) Sequences cannot be saved in your Profile. In Hermes Version 2.4, sequences disappear as soon as the Use Gets another message-file or Quits MAILSYS.

There is always one current default sequence, named CSEQUENCE.

MAILSYS Standard Initial setting:
CSEQUENCE = ALLMESSAGES

Possible sequences are:

One-time, throwaway sequences that you type into a command.

Ready-made, named sequences:
ALLMESSAGES = *, all the messages in CMESSAGEFILE.
NEWMESSAGES, arrived since you entered CMESSAGE-FILE (either by starting MAILSYS or by using the Get command to enter a message-file other than your Inbox).
RECENTMESSAGES, arrived since the last time you entered CMESSAGE-FILE
OLDMESSAGES, arrived before the last time you entered CMESSAGE-FILE
NULLSEQUENCE, no messages, an empty sequence

User-created, named sequences.

-----
TYPING IN SEQUENCES

A Sequence is an ordered set of message-nos., defined as the logical union of any number of individual sets. The sets may be defined in terms of message-nos., or named Sequences (created either by the system or by the User), or Sequences acted upon by Filters.

A Sequence may be typed as a series of one or more <group>s, separated by semicolons:

```
    <group>;...;<group>
```

A <group> may be typed as a series of one or more <subgroups>, separated by commas, and followed by, and acted upon by, a series of one or more filters. The filters are separated from each other, and from the final <subgroup>, by slashes:

```
<subgroup>,...,<subgroup>/<filter>/.../<filter>
```

```
where a <subgroup> may be:
a message-no.  or symbol for a message-no.:
    Ex.: 3, 5
         % = last message in CMESSAGE-FILE
         .  = the current message CMESSAGE
a range of numbers, or a range symbol.
    Ex.: 1:3 = 1,2,3
         9:5 = 9,8,7,6,5
         6:% 3:.  .:%
         * = 1:% = all messages in the CMESSAGE-FILE
a named Sequence
    Ex.: ALLMESSAGES = *
         RECENTMESSAGES
         CSEQUENCE = the current Sequence
         any User-created Sequence
```

NOTE:
(a) DO NOT place a space |SP| before any semicolon, comma, or slash n the middle of a Sequence.
(b) If none of the groups contain Filters, the semicolon and the comma may be used interchangeably.
(c) In real life, you probably won't ever type a command more than a line long, but if you must, DO NOT put a carriage return |CR| at the end of the line. Let your terminal give you a runover line.

----------

SEQUENCE-EDITOR

```
>Create Sequence <name>|CR|
```

places a temporary copy of a new and empty sequence named <name > in the SEQUENCE-EDITOR. If a sequence named <name> aready exits, Create SEQUENCE has the same effect as

```
>Edit <name>|CR|
```

where <name> may be CSEQUENCE or any named sequence; places a temporary copy of <name> in the SEQUENCE-EDITOR.

Subcommand options:

```
>>Show|CR|
```
prints the contents of the temporary sequence on your terminal; can be used at any time to check the temporary sequence as it is being edited.

```
>>Survey|CR|
```
Prints Surveys of the entire temporary sequence.

```
>>Sort (by) <attribute>|CR|
    rearranges the messages on the temporary sequence into
    alphabetic or numerical order where <attribute may be
    Message-No., Char-Count (Length), Date, Rcvd-Date,
    From-Field, Message-ID, or Subject-Field

>>Add <sequence:2>|CR|
    creates a new temporary sequence consisting of the messages
    that appear on the temporary sequence plus the messages on
    <sequence:2>, with duplicate messages removed.  This is the
    union of the temporary sequence and <sequence:2>.
>>Add|CR| gives an error report.

>>Erase <sequence:2>|CR|
    creates a new temporary sequence by removing the messages on
    <sequence:2> from the temporary sequence.
>>Erase|CR| erases the entire temporary sequence.

>>Intersect-with <sequence:2>|CR|
    creates a new temporary sequence which contains only the
    messages that appear on both the temporary sequence and
    <sequence:2>.  This is the intersection of the temporary
    sequence and <sequence:2>.

>>|Control-E|
    Destroys the temporary sequence, and does not change the
    contents of the pre-exisitng sequence named <name>.  Returns
    you to top command level.

>>Done|CR|
    Causes the temporary sequence to become the neew conents of
    the sequence named <name>.  Destroys the previous contents of
    <name>.  Returns you to top command level.
----------

Show
Object-Handling Command

>Show <object>|CR|

where <object> may be

All, shows contents of all objects in the ACTIVE ENVIRONMENT.
    = Show|CR|
Names, shows names only of all objects.
Current-Objects, shows contents of CDRAFT, CMESSAGE-FILE,
    CMESSAGE, CSEQUENCE, CFILTER, PTEMPLATE, STEMPLATE,
    FDESTINATION, and SWITCHES
Any object in the ACTIVE ENVIRONMENT,
    shows contents of object

>Show All <object-type>|CR|

where <object-type> may be
Objects, shows contents of all objects in the ACTIVE ENVIRONMENT
    = Show All|CR|
Fields, shows contents of all fields of the CDRAFT.
    = Show CDRAFT|CR|
Sequences, shows contents of all Sequences
Filters, shows contents of all Filters
Templates, shows contents of all Templates
Destinations, shows contents of FDESTINATION
    = Show FDESTINATION|CR|
----------
```

Show TypedForm
Subcommand of Create DRAFT

>>Show TypedForm <address-field>|CR|

shows the contents of an address-field (To:, Cc: or Bcc:) AS YOU HAVE TYPED IT. Where distribution lists with groupnames are used, Show shows only the groupname, while Show TypedForm allows you to see the members of the list.
----------

Special-Handling:
Header-Field; Subcommand of Create DRAFT

>>Special-Handling: <string>|CR|

where <string> is any string of characters that does not contain |CR|.

To continue the field to another line, type ,|CR| instead of |CR|. The comma is not inserted in the field. This subcommand adds to the current contents, if any, of the Special-Handling:-field.
----------

Status
Object-Handling Command

>Status|CR| = Show CMESSAGE-FILE|CR|

shows the name of the CMESSAGE-FILE, the number of messages it contains, the number UNSEEN and the number DELETED.
----------

STEMPLATE
Object

The STEMPLATE is the template always used by the Survey and Summarize commands.

MAILSYS Standard Initial setting:
STEMPLATE = BASIC-SUMMARY

How to change STEMPLATE:
>Use STEMPLATE <template>|CR|

where <template> is any named template; copies the contents of <template> into the STEMPLATE.

>Copy STEMPLATE (and name it) <name>|CR|

copies the contents of STEMPLATE into a User-defined templatei named <name>.

>Copy <template> (and name it) STEMPLATE|CR|

copies the contents of any named template into STEMPLATE.

Only one STEMPLATE can exist at a time. Your choice of STEMPLATE may be saved in your PROFILE.
-----------

Store-Draft
Subcommand of Create Draft; Creates or Appends to CDRAFT

>Store-Draft (on file) <TENEX file-name>|CR|

```
- Save-Field All (on file) <TENEX file-name>|CR|
```

saves the entire CDRAFT in a TENEX file.

To bring the Draft stored with Store-Draft back into the CDRAFT, use the command

```
>Restore-Draft (from file) TENEX file-name)|CR|
```

appends the fields stored in the <TENEX file to the corresponding fields of the CDRAFT.

----------

Subject:
Header-Field; Subcommand of Create DRAFT

```
>Subject: <string>|CR|
```

where <string> is any string of characters that does not contain|CR|.

To continue the field to another line, type ,|CR| instead of |CR|. The comma is not inserted in the field. This subcommand adds to the current contents, if any, of the Subject:-field.

----------

Suggestion
Utility Command

```
>Suggestion |CR|
```

Suggestion prompts for Subject: and Text:-fields. When the final |Control-Z| of the Text:-field is typed, Suggestion creates and immediately send a message to the designers of MAILSYS.

Note: You cannot delay sending for editing and second thoughts. Suggestion does not interfere with CDRAFT and so may be used to fire off a quickie suggestion whle you are creating another message.

----------

Summarize

```
>Summarize <sequence> <destination>|CR|
>Summarize|CR| = Summarize RECENTMESSAGES TTY:|CR|
```

Summarize prints messages according to STEMPLATE on the <destination>.

Note: the <sequence> defaults to RECENTMESSAGES for convenience in reviewing each day's messages. Otherwise, Summarize is identical to Survey.

----------

Survey
Message-Handling Command

```
>Survey <sequence> <destination>|CR|
>Survey|CR| = Survey CSEQUENCE TTY:|CR|
```

prints messages according to STEMPLATE on the <destination>.

-----------

S = Single-character synonym for Survey

----------

SWITCHES
Object

SWITCHES give you a choice of different ways in which some MAILSYS commands and other features operate. The command Edit SWITCHES calls the SWITCHES-EDITOR and lets you set SWITCHES for the current MAILSYS session. You cannot Copy or Create SWITCHES.

Each SWITCH can have only one setting at a time. You can set SWITCH settings in your Profile with

```
>Edit Profile|CR|
>>Edit SWITCHES|CR|
>>>Set <switch-name> (to) <setting>|CR|
>>>Done|CR|
>>Done|CR|
```

INDIVIDUAL SWITCHES
(MAILSYS Standard Setting is No. 1 for each Switch)

Message-File:

INITIAL-SURVEY SWITCH
1-InitialSurvey: At the beginning of a MAILSYS session, and whenever the User Gets a new CMESSAGE-FILE, MAILSYS prints an automatic SURVEY or RECENT messages.
2-NoInitialSurvey: Does not print out an initial survey.

REPORTNEWMSG SWITCH
1-Periodic&AtPrompt: Messages that arrive during a MAILSYS session are marked NEW and reported after a fixed interval, at the next MAILSYS prompt, or periodically without a prompt, whichever comes first.
2-AtPrompt: NEW messages are reported, after a fixed interval, at the next MAILSYS prompt.
3-DelayReport: NEW messages are reported only when you leave the CMESSAGE-FILE or give an Expunge command.

REPORTFORM SWITCH
1-SurveyReport: NEW messages are reported by a bulletin: "00 NEW MESSAGES HAVE JUST ARRIVED" and a survey of each message.
2-BulletinReport: New messages are reported by the bulletin only.
3-NoReport: New messages are not reported in MAILSYS.

GET-DIRECTORY SWITCH
1-ConnectDirectory: The initial CMESSAGE-FILE is MESSAGE.TXT;1 in the CONNECTED directory.
2-LoginDirectory: The initial CMESSAGE-FILE is MESSAGE.TXT;1 in the LOGIN directory.

GET-EXPUNGE SWITCH
1-AskExpunge: Asks permission to expunge deleted messages and renumber old CMESSAGE-FILE when you Quit MAILSYS for Get a new CMESSAGE-FILE. MAILSYS asks only when old CMESSAGE-FILE contains deleted messages and you have permission to Delete.
2-Expunge: Automatic expunging and renumbering.
3-NoExpunge: No expunging and renumbering. Use the Expunge command.

Message-Handling:

DELETE SWITCH

1-AskConfirm: MAILSYS prints a survey and asks you to confirm
with a |CR| if, and only if, the message has NOT been SEEN
during the current MAILSYS session.
2-NoAskConfirm: Does not request confirmation.

FILE SWITCH: (File and Move are identical, but the SWITCHES are
independent. File has subcommand that overrides SWITCH.)
1-NoDelete: Does not delete messages from CMESSAGE-FILE.
2-Delete: Automatically deletes messages.

MOVE SWITCH: (File and Move are identical, but the SWITCHES are
independent. Move has subcommand that overrides SWITCH.)
1-NoDelete: Does not delete messages from CMESSAGE-FILE.
2-Delete: Automatically deletes messages.

Draft-Creating:

FROM-NAME SWITCH
1-Login-Directory: Fills in the From:-field with the name of the
LOGIN Directory.
Connect-Directory: Fills the From:-field with the name of the
CONNECTED Directory.

COMPOSE-SEND SWITCH (Compose subcommand overrides SWITCH.)
1-AskSend: Asks SEND? at the end of prompted message-composing
sequence. Confirm with |CR| or Y|CR| to Send message
immediately. If you press |ESC|, MAILSYS prints Yes, and
waits for you to confirm with |CR| or abort with |DEL|. Type
N|CR| to return to Create-Draft subcommand level for further
editing.
2-Send: Immediate automatic sending.
3-NoSend: No automatic sending. Use the Send command.

FORWARD-SEND SWITCH (Forward subcommand overrides SWITCH.)
1-AskSend: Asks SEND? at the end of prompted message-composing
sequence. Confirm with |CR| or Y|CR| to Send message
immediately. If you press |ESC|, MAILSYS prints Yes, and
waits for you to confirm with |CR| or abort with |DEL|. Type
N|CR| to return to Create-Draft subcommand level for further
editing.
2-Send: Immediate automatic sending.
3-NoSend: No automatic sending. Use the Send command.

REPLY-SEND SWITCH (Reply subcommand overrides SWITCH.)
1-AskSend: Asks SEND? at the end of prompted message-composing
sequence. Confirm with |CR| or Y|CR| to Send message
immediately. If you press |ESC|. MAILSYS prints Yes, and
waits for you to confirm with |CR| or abort with |DEL|. Type
N|CR| to return to Create-Draft subcommand level for further
editing.
2-Send: Immediate automatic sending.
3-NoSend: No automatic sending. Use the Send command.

REPLY-COPIES SWITCH (Reply subcommand overrides SWITCH.)
1-NoCopies: Sends reply only to Sender of original message.
2-AskCopies: If you type |CR| or Y|CR|, MAILSYS send copies of
reply to all names in the To: and Cc: fields. If you press
|ESC|, MAILSYS prints Yes, and waits for you to confirm with
|CR| or abort with |DEL|. If you type N|CR|, MAILSYS sends
reply only to original sender.
3-Copies: Sends copies of reply to all names in To: and
Cc: fields of original message.

FORMAT SWITCH
1-NoJustify: Fills lines; spaces between words.

2-Justify: Automatically fills lines and evens up right-hand
    margins by inserting extra spaces between words.

SEND-ERASE SWITCH (Send has subcommand that overrides SWITCH.)
1-Erase: Automatically erases DRAFT Fields after sending.
2-NoErase: DRAFT Fields are not erased.

SEND-ARCHIVE SWITCH -- ARPA Network Archiving Facility
(Send has subcommand that overrides SWITCH.)
1-NoArchive: Outgoing messages are not archived in the ARPA
    Network DATACOMPUTER.
2-Archive: Outgoing messages are automatically archived in the
    ARPA Network DATACOMPUTER for permanent storage.

    The DATACOMPUTER is an experimental, public-access
    information facility whose entire contents is accessible to
    all users of the ARPANET. Messages Archived in the
    DATACOMPUTER can be retrieved with the command Retrieve.

    WARNING!!! The Archive subcommand has no relationship to the
    Archive facility that may be implemented as part of your
    local TENEX system.

Utility Commands:

QUIT SWITCH (Quit and Exit are identical, but SWITCHES are
    independent. Quit has subcommand that overrides SWITCH).
1-AskExpunge: Asks permission to expunge deleted messages and
    renumber old CMESSAGE-FILE when you Quit MAILSYS or Get a new
    CMESSAGE-FILE. MAILSYS asks only when CMESSAGE-FILE contains
    deleted messages and you have permission to delete.
2-Expunge: Automatic expunging and renumbering. If you type
    |CR| or Y|CR|, MAILSYS expunges and renumbers. If you press
    |ESC|, MAILSYS prints Yes and waits for you to confirm with
    |CR| or abort confirmation with |DEL|. If you type N|CR|,
    MAILSYS does not expunge.
3-NoExpunge: No expunging and renumbering. Use the Expunge
    command.

EXIT SWITCH (Quit and Exit are identical, but SWITCHES are
    independent. Exit has subcommand that overrides SWITCH.)
1-AskExpunge: Asks permission to expunge deleted messages and
    renumber old CMESSAGE-FILE when you Exit MAILSYS or Get a new
    CMESSAGE-FILE. MAILSYS asks only when CMESSAGE-FILE contains
    deleted messages and you have permission to delete. If you
    type |CR| or Y|CR|, MAILSYS expunges and renumbers. If you
    press |ESC|, MAILSYS prints Yes and waits for you to confirm
    with |CR| or abort confirmation with |DEL|. If you type
    N|CR|, MAILSYS does not expunge.
2-Expunge: Automatic expunging and renumbering.
3-NoExpunge: No expunging and renumbering. Use the Expunge
    command.
----------

SWITCHES-EDITOR

>Edit SWITCHES|CR|

places a temporary copy of the SWITCHES settings in the SWITCHES-EDITOR.

Subcommand options:

>>Show <switch-name>|CR| shows the temporary setting of an
    individual switch.
>>Show|CR|

    shows a table of temporary settings of the entire set of
    SWITCHES.

```
>>Set <switch-name> (to be) <setting>|CR|
```
    changes the temporary setting of the individual switch.

```
>>|Control-E|
```
    destroys the temporary settings of SWITCHES, and does not
    change the previous settings.  Returns you to top command
    level.

```
>>Done|CR|
```
    causes the temporary settings of SWITCHES to become the new
    SWITCHES; ends subcommands, and returns you to top command
    level.

-----------

Talk
Utility Command

```
>Talk (Characters to ^Z ignored) <string>|CR|
```

allows you to type in a string of characters that has no effect on MAILSYS. You can terminate the <string> only by typing |Control-Z|.

The most common reason for using Talk is to "talk" to someone who has linked to you through the TENEX Exec subsystem "Link".

The single-character command ";" allows you to type a one-line string of characters that has no effect on MAILSYS. The ";" line is terminated by a |CR|.

----------

Teco
Text Editor; Subcommand of Create DRAFT

Teco, Neted, and Xed are the text editors available for editing message fields. To call Teco, type

```
>>Teco <field>|CR|
>>Teco|CR| = Teco Text:|CR|
```

Teco creates an "inferior fork" of TENEX Teco, that is, a copy of Teco inside MAILSYS, with the specified field of the message in the Teco buffer. You may then use all of the available Teco commands to edit the field.

After editing, return to MAILSYS by typing ;H|ESC| to the Teco program. MAILSYS prints out OK?

If you respond to OK? by typing Y, MAILSYS inserts the edited version of the buffer as the new message field. If you type N, the edited version of the buffer disappears, and MAILSYS does not change the contents of the field. In either case, you are returned to MAILSYS subcommand level.

----------

Template
Object-type

A template specifies which parts of the message are output on the <destination>.

There are always two current default templates, named PTEMPLATE and STEMPLATE.

MAILSYS Standard Setting:
PTEMPLATE = RCVD-FORM.
STEMPLATE = BASIC-SUMMARY

Possible templates are:

Ready-made, named templates:

RCVD-FORM
    includes the message-no., the character count, the UNSEEN-RECENT status, and a VERBATIM item which moves a copy of the message exactly as it was received. When you edit a copy of RCVD-FORM, you can erase the VERBATIM item, but the VERBATIM item cannot be split up.

STANDARD
    moves all the message, including all fields, the message-no., the character count, and the UNSEEN-RECENT status. Rearranges the order of the fields if they are not in standard form. A copy can be Edited to create a new template.

BASIC-SUMMARY
    (used by SURVEY) moves a one-line summary of the message that includes the UNSEEN-RECENT status, the message-no., the character count, the date received, the From:-field, and as much of the Subject:-field as will fit on the line. A copy can be Edited to create a new template.

NULLTEMPLATE
    moves nothing. A copy can be Edited to create a new template.

User-created, named templates.

The Use command sets either the PTEMPLATE or the STEMPLATE to a copy of any other template (e.g., STANDARD or BASIC-SUMMARY). PTEMPLATE and STEMPLATE can be set and User-created templates can be saved in your PROFILE.

----------

TEMPLATE-EDITOR

```
>Create Template <name>|CR|
```

places a temporary copy of a new, empty template named <name> in the TEMPLATE-EDITOR. IF a template named <name> aleady exists, Create Template has the same effect as

```
>Edit <name>|CR|
```

where <name> may be PTEMPLATE, STEMPLATE or any User-created template; places a temporary copy of <name> in the TEMPLATE-EDITOR.

Subcommand options:

```
>>Show <line-number>|CR|
  Show|CR|
          Shows entire temporary template

>>Erase (line number) <number>|CR|
  Erase|CR|
          Erases entire temporary template.

>>Item-Insert (on) <line-number> (before) <item-number>
  (items) <item-group> |CR|
where <item-group> is a series of one or more <items>,
          separated by spaces, commas, or plus signs.

>>Line-Insert (before) <line-number>|CR|
(Type template lines, ending with ^Z)
<item-text>|Control-Z|
where <item-text> is a series of zero, one or more <items>,
          separated by either spaces or |CR|s.

<item> may be one of the following:

Status
          prints -+

Status+
          prints UNSEEN RECENT

Verbatim
          prints entire message exactly as received.

<field-name>
          prints contents of field.

<field-name>+
          prints name of field followed by contents of field.

Source
          Prints FROM:-field, except that if the message is from
          the User (i.e., if the FROM:-field contains the name of
          the LOGIN directory), Source prints "To:" followed by
          the first name in the To: field.

"<quoted string>"
          Prints <quoted string> exactly as typed.  <Quoted
          string> may include spaces, but may not include |CR|s.

Separate
          Inserts a formfeed |Control-L| that starts a new page
          with the next <item> (or the next message if Separate
          is at the end of the template).

and <field-name> may be:
 Message-No.
 Char-Count
 Rcvd-Date
 Date:
 Sender:
 Subject:
 From:
 To:
 Cc:
 Info:
 Bcc:
 Message-ID:
 In-Reply-To:
 References:
 Keywords:
 Precedence:
 Message-Class:
 Classification:
 Special-Handling:
 Fcc:
 Other: Any other Header-field.
 Text:

>>|Control-E|
```

Destroys temporary template, and does not change
contents of <name>. Returns you to top command level.

>>Done|CR|

Causes temporary template to become to become new
contents of template named <name>. Destroys previous
contents of <name>. Returns you to top command level.

----------

Text:
Text:-Field; Subcommand of Create DRAFT

>>Text: <text-string> |Control-Z|

where <text-string> is any string of characters that does not
contain |Control-Z| (echoed as; ^Z) and that may contain|CR|.

Unlike the Header-fields, the Text:-field is terminated by
|Control-Z|. This allows you to use |CR| normally within the
Text: of the message.

The <text-string> is added to the current contents (if any) of
the Text:-field.

TELNET USERS: Change your escape character from |Control-Z| to
something else before you use MAILSYS. Otherwise, |Control-Z| at
the end of the Text:-field returns you to TELNET.

----------

Text-Editing Characters

|Control-A|

deletes a single character; does not delete back across unit
parts of an addressee name in the To:, Cc:, or Bcc: fields,
or a TENEX file designator in the Fcc: -field. Echoes as \
followed by the deleted character.

|Control-W|

deletes a single word in a text string, and echoes as <
followed by the first letter of the deleted word. Also
deletes an entire addressee <name spec> or TENEX file
designator.

|Control-Q|

deletes a line of text in a text string, or used at the
beginning of a line, deletes the previous line. Echoes as a
new line; works the same way in To:, Cc:, Bcc:, and Fcc:
fields. In a command string, deletes the current part of a
commands.

|Control-B| <TENEX file designator>

inputs text from a designated file while you are creating a
message field; i.e., after you have given the command and
before you type the final|CR| (for a Header) or |Control-Z|
(for the Text:-field). Does not echo. You can abort
|Control-B| with the DELETE-key |RUBOUT| while you are typing
the <TENEX file-name>.

----------

To:
Header-Field; Subcommand of Create DRAFT

>>To: <addressee-list>|CR|

The command To: allows you to specify a list of addressees to
whom "action" copies of the message are to be sent.

Type Describe Address-Fields|CR| for an explanation of <addressee-list>.

To continue the field to another line, type ,|CR| instead of |CR|. The comma is not inserted in the field. This subcommand adds to the current contents, if any, of the To:-field.

----------

Undelete
Message-Handling Command

```
>Undelete <sequence>|CR|
>Undelete|CR| = Undelete CSEQUENCE|CR|
```

Undelete removes the DELETE markings from messages that have been marked for deletion. There is no difference between messages that have been Deleted and then Undeleted and messages that have never been Deleted. Messages that do not have DELETE markings may be selected with the UNDELETED filter.

----------

Use
Object-Handling Command

```
>Use <current object> <literal value or name>|CR|
```

Use creates a current object which is a copy of the literal value or named object. Type Describe Current-Object for details.

Possible Current-Objects:

CMESSAGE-FILE, takes a Tenex file-name only
CSEQUENCE, takes literal value, named sequence, or any
    combination of sequences with filters. Type Describe
    Sequences for details.
CFILTER, takes literal value or named filter
PTEMPLATE, takes named template only
STEMPLATE, takes named template only.
FDESTINATION, takes TENEX file-name only.

There are three other commands which are special cases of Use:

```
>Get <file-name>|CR|
   = Use (current object) CMESSAGE-FILE <file-name>|CR|
   changes the CMESSAGE-FILE to <file-name>.
```

MAILSYS Standard Setting:
CMESSAGE-LIST = MESSAGE.TXT;1 in the User's LOGIN directory.

```
>Consider <sequence>|CR|
   = Use CSEQUENCE <sequence>|CR|
   changes the CSEQUENCE to a copy of <sequence>.
```

MAILSYS Standard Setting:
CSEQUENCE = ALLMESSAGES

```
>Jump-To <message-no.>|CR|
   = Use (current object) CMESSAGE <message-no.>|CR|     = Use
(as) .  <message-no.>|CR|
   changes the message-no.  of the CMESSAGE to <message-no.>
```

MAILSYS Standard Setting:
CMESSAGE = the message just before the first RECENT message.

NOTE: CMESSAGE is also reset by the commands Print and |LF|.
----------

Version
Utility Command

```
>>Version|CR|
```

Version prints out the MAILSYS herald line and other information about the system in use. If you have trouble, the information provided by VERSION will be of great help in debugging the system.

----------

Xed
Text Editor; Subcommand of Create DRAFT

Xed, Neted and Teco are the three text editors available for editing message fields. To call Xed, type

```
>>Xed <field>|CR|
>>Xed|CR| = Xed Text:|CR|
```

Xed creates an "inferior fork" of TENEX Xed, that is, a copy of Xed inside MAILSYS, with the specified field of the message in the Xed buffer. You may then use all of the available Xed commands to edit the field.

After editing, if you type E|CR|, Xed prints out "Returning updated text to MAILSYS," and MAILSYS inserts the edited version of the buffer as the new message field. If you type Q|CR|, Xed prints out "Returning to MAILSYS without updating," the edited version of the buffer disappears, and MAILSYS does not change the contents of the message field. In either case, you are returned to MAILSYS command level.
----------