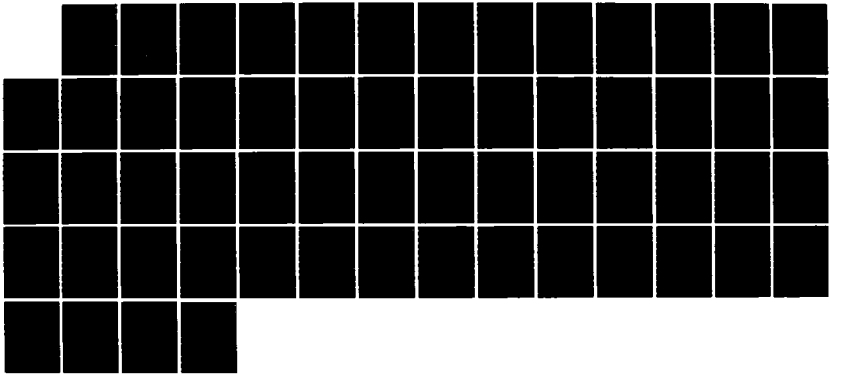
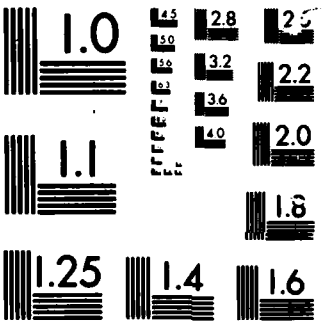
AD-A167 128 REQUIREMENTS FOR THE MILITARY MESSAGE SYSTEM (MMS) 1/1
FAMILY: DATA TYPES AND USER COMMANDS(U) NAVAL RESEARCH
LAB WASHINGTON DC C L HEITMEYER 11 APR 86 NRL-MR-5760
UNCLASSIFIED F/G 9/3 NL





MICROCOPY CHART

NRL Memorandum Report 5760

Requirements for the Military Message System (MMS) Family: Data Types and User Commands

CONSTANCE L. HEITMEYER

Computer Science and Systems Branch
Information Technology Division

April 11, 1986

NAVAL RESEARCH LABORATORY
Washington, D.C.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NRL Memorandum Report 5760	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Research Laboratory 6b. OFFICE SYMBOL (If applicable) Code 7596	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Washington, DC 20375-5000	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Space and Naval Warfare Sys. Com. 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Washington, DC 20363-5001	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 35167G; PROJECT NO.; TASK NO. 68003; WORK UNIT ACCESSION NO DN880-204
11. TITLE (Include Security Classification) Requirements for the Military Message System (MMS) Family: Data Types and User Commands	
12. PERSONAL AUTHOR(S) Heitmeyer, Constance L.	
13a. TYPE OF REPORT Interim 13b. TIME COVERED FROM 1981 TO 1985	14. DATE OF REPORT (Year, Month, Day) 1986 April 11 15. PAGE COUNT 59
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB-GROUP	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Requirements; Security model; Functional specifications; Specifications; Program families; Message systems

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

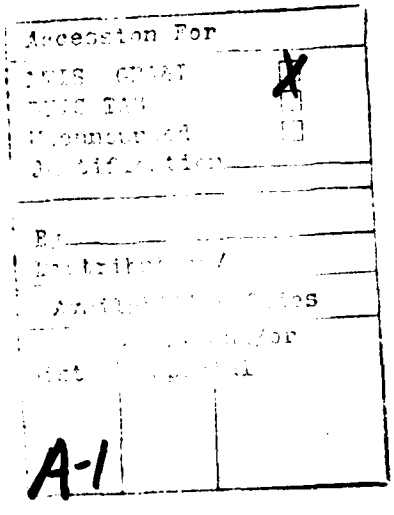
This report, an application of Parnas' family methodology, describes the required user-visible behavior of a family of Military Message Systems (MMS). These systems have two significant areas of commonality: all enforce the rules described by the MMS security model; and the user-visible data types and user commands required by each system can be extracted from a large set of data types and user commands. Based on a study of several existing message systems, this report defines the set of data types (e.g., message, message file, directory) and user commands associated with the MMS family. We use an Intermediate Command Language (ICL) to describe the user commands abstractly, i.e., to specify each command's effects without imposing unnecessary restrictions on either the command syntax or the physical characteristics of the user's terminal. To simplify and shorten the specifications, we define a hierarchy of data types. Also included is a glossary of terms.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Constance L. Heitmeyer	22b. TELEPHONE (Include Area Code) (202) 767-2774 22c. OFFICE SYMBOL Code 7590

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

CONTENTS



REQUIREMENTS FOR THE MILITARY MESSAGE SYSTEM (MMS) FAMILY: DATA TYPES AND USER COMMANDS

Introduction

A major goal of the Military Message System (MMS) project is to specify the requirements and design of a family of military message systems. Together with the MMS security model [5], this report defines the functional requirements of the MMS family. It describes the set of data types associated with the family and provides informal specifications for the MMS user commands. The report is a revision and extension of earlier work [2-4].

A previous report [2] specifies three members of the MMS family. This report, which is an extension of [2], has the same organization and includes most of the content of [2] as a subset. This material was included to make the report self-contained. For readers familiar with the previous report, we list the differences between the two in Appendix A.

The report contains four sections: Section 1 provides background information and descriptions of the MMS data types; Section 2 lists the MMS user commands; Section 3 contains specifications for the user commands; and Section 4 provides a glossary.

1. Overview of the MMS Family

This section reviews the family methodology and its application in the MMS project, identifies and describes the MMS data types, summarizes the role of the Intermediate Command Language (ICL) in the specifications, and discusses the relationship between the specifications and the MMS security model [5].

In this report, two data items are of the same **data type** if a common set of operations may be applied to them. Examples of data types are 'entity', 'message', and 'message file'. A data item may belong to more than one data type. For example, a message *m* belongs to both the data type 'message' and the data type 'entity', since all operations defined for entity and all operations defined for message may be applied to *m*.

1.1. Background: Family Methodology

Review of the requirements of future military message systems suggests that a single message system will not suffice for all environments. While these systems will have much in common, supporting many of the same message processing functions and enforcing the same security rules, they will also have important differences, e.g., in the special functions that they perform, in their user interfaces, and in the hardware on which they are built.

In the past, similar systems have been constructed in two different ways. In many cases, such systems have been developed independently. In other cases, the first system in a series of similar systems is built to satisfy a given set of requirements; subsequent systems are built by changing the code of the first system to satisfy new requirements. Both approaches have serious disadvantages: they result in high costs for development and maintenance; with the second approach, systems built subsequent to the first are often difficult to change and exhibit performance problems [3,8].

To help overcome these problems, we advocate a different approach, called the *family methodology* [8], which requires developers to consider the entire family before building a single member. Using this approach, developers maximize what family members have in common and, to the extent feasible, minimize member differences. During the design phase, a modular structure suitable for all family members is formulated. The different features of family members are assigned to separate modules, so that producing a different family member can be done by replacing modules, not by changing the overall program structure. For example, a family member with a different user command language can be generated by replacing the user command language module.

Manuscript approved January 6, 1986.

Applying the family methodology to military message systems requires that we identify and exploit features common to all members [3]. To date, we have identified two significant areas of commonality among military message systems:

1. *Security.* Family members enforce the same security rules.
2. *Functions.* The user-visible data types and user commands required by individual members can be extracted from a large set of data types and user commands.

The MMS security model [5] defines the set of security rules that all MMS family members enforce. This report defines the set of user-visible data types and user commands associated with the MMS family.

While it is the intent of this document to describe the user commands required by a wide range of military message systems, we do not rule out future additions to the set of user commands. Some family members may require commands that we cannot anticipate. Other commands, such as commands invoked by the system operator for auditing and archiving and UNDO* commands, though compatible with this document, are beyond its scope.

1.2. MMS Data Types

This section and Section 3 describe the set of data types and user commands associated with the MMS family. In developing this set, we have reviewed, and incorporated features of, a number of message systems. Among the most important of these are SIGMA [9], an experimental military message system which includes several special functions for processing formal military messages; HERMES [6], a system with a broad range of functions that is widely used on the ARPA network; NMIC-SS [7], a military system developed for intelligence analysts; and Laurel [1], a display-oriented system (with a mouse) used in the Xerox Research Internet.

An important feature of the MMS family is that some members support fewer functions than others. For example, M0 is a receive-only family member that supports the receipt and display of messages and their storage in message files; it does not allow users to compose and transmit messages. Because the operations it provides are very limited, M0 only offers a few data types (informal sent messages, message files, directories, and users) and 12 user commands [2]. In contrast, another family member, M2, includes many more commands, e.g., commands to compose and transmit formal messages, to create and edit text files, to define discretionary access controls, and to reclassify information. M2 offers many more data types than M0 (including formal messages, draft messages, text files, and terminals) and 76 user commands [2].

Associated with every MMS member is some subset of the data types described below. We expect nearly all MMS members to support a few of these data types, namely, message, message file, directory, and user.

1.2.1. Entities and Users

A MMS user issues commands to log into and out of the system and to perform various operations on MMS data items. There are two major classes of MMS data types: users and entities.** A **user** is a person authorized to use the MMS. Associated with each user are three attributes: the user's **clearance**, a set of **authorized roles** (e.g., downgrader, releaser, system security officer), and a set of **current roles.** Examples of MMS **entities** include:

*A command followed by an UNDO restores a system to its state prior to the execution of the command.

The meaning of the term **data type is broader in this report than in the MMS security model formalization, since the report identifies users and entities as data types, whereas in the formalization data type is an attribute of an entity.

messages	**filters**	**directories**
message entries	**forms**	**devices**
message files	**comments**	**paragraphs**
text files	**user profiles**	**message fields**
keywords/keyword sets		

Associated with each entity are four attributes: the entity **classification** (e.g., SECRET), its **value**, its **data type**, and its **access set** (which describes the operations that specified users may perform on the entity). Some entities, called **containers**, may have an additional attribute, called **Container Clearance Required (CCR)**, that requires any user who wishes to view information in the container to have a clearance greater than or equal to the classification of the container. In a MMS, a **userID** is a name for a user and a **reference** is a name for an entity.

1.2.2. Message Files and Message Entries

Each MMS user is assigned a special message file, called an **inbox**, in which the MMS places an **entry** for each message sent to the user. An entry contains the message and message status information (e.g., whether the message is "new", whether the message was received "for-action", "for-info", etc.). When the user displays his inbox or any other message file, only part of each message entry in the file is displayed. The part consists of selected message fields (e.g., the Subject and To fields) and the message status information. Some MMS members allow users to include **keywords** in message entries. These keywords can be included in filters (see below) and hence are useful in searching for messages that satisfy certain criteria. In most MMSs, users may create named message files for organizing and storing messages.

Every MMS that supports formal messages has two system files, OUTGOING and INCOMING. The MMS appends an entry to OUTGOING for each formal message that it transmits and an entry to INCOMING for each formal message that it receives. Both files are non-CCR, and no entries in them may be deleted or removed.

1.2.3. Messages

Every message is either a draft message or a sent message. A **draft message** is a message in draft form; users create and edit draft messages. A **sent message** is a message that has been released, i.e., sent to one or more other users. By issuing the command SEND_MSG, the user converts a draft message into a sent message. A user may send messages to local users, i.e., users on the same MMS, or to remote users, i.e., users on different message systems. For each remote user to whom a message is sent, the MMS may transmit a copy of the message over the appropriate network. For each local user, the MMS creates an entry for the message and inserts the entry into the user's inbox.

Every message has a **message type.** Many family members support two message types: **formal** messages and **informal** messages. The message type is determined at the time the message is created. When a user issues the SEND_MSG command, thus converting a message from a draft to a sent message, the message type does not change. Messages of two different message types can differ (1) in the set of fields that they contain and (2) in the set of user commands that can be applied to them.

1.2.4. Text Files

A **text file** is a named sequence of paragraphs. In some MMS members, users can create and edit named text files. The purpose of text files is to store address lists or other textual information. Users can transfer such information to and from messages.

1.2.5. Filters

A **filter** is a set of criteria that can be applied to a sequence of message entries. A message entry "matches" a filter if it satisfies the filter's criteria. Some filters, such as ALL and NEW, are provided by the system. Every entry satisfies ALL, whereas only message entries received by the MMS after the user's last MMS session satisfy NEW. Some MMS members allow users to define their own special filters. Such filters may be **literal**; for example, the user command "DISPLAY_MF INBOX SUBJ=IRAN" displays all entries in the user's inbox that contain the string "IRAN" in the Subject field. Users may also create **named** filters; for example, a user might define a filter named IRAN with value "SUBJ=IRAN". Then, the user command "DISPLAY_MF INBOX IRAN" has the same effect as the command in the previous example.

1.2.6. Forms

A **form** defines the output format for a given entity type. Some MMS members allow users to define special output forms and use them in place of the standard forms provided by the MMS. Such forms may be used in various ways, e.g., to vary the order in which message fields are displayed/printed, to suppress the display/printing of certain fields or certain parts of message entries, etc.

1.2.7. Comments

A **comment** is text that is attached to another entity. A MMS that supports comments may allow a user to attach more than one comment to a single entity. Many family members limit the data types to which comments may be attached (e.g., to message fields, paragraphs of the Text field of messages, and message entries). Displaying or printing an entity results in the output of the entity's value and the values of any comments that are attached to the entity.* Since comments do not have explicit names, the user gains access to a comment via the entity to which it is attached. A user who is editing an entity may modify any comments attached to the entity to which he has 'modify' access.

1.2.8. Directories

A **directory** is a set whose elements are entities and other directories. In some family members, all entities in a directory must belong to the same type. For example, in M2, each user is assigned a message file directory, which contains the user's inbox and each message file that the user creates, and a text file directory, which contains each text file that the user creates.

1.2.9. User Profiles

A **user profile** is a set that modifies MMS behavior for the user who owns the profile. In MMS members that support user profiles, each new user is assigned a standard profile that he may later change to satisfy his individual requirements. A user profile contains default definitions and command scripts. A **default definition** allows a user to substitute his own definition for a given system parameter. For example, a user may define a special form for printed messages; by including this form in his profile, the user causes the message copies that he prints to have the specially defined format rather than a standard MMS format. A **command script** is a named sequence of user commands. By defining scripts, the user can invent new, more powerful commands. For example, a user may define a script named ROUTE that forwards a set of messages to one group of users for action and to another group for information, copies the set to a specified file, and then deletes the set from his inbox. Thus a single ROUTE command replaces four individual user commands.

*By specifying a form suppressing comments as an input parameter to a DISPLAY PRINT command, the user can suppress the output of comments.

1.2.10. Devices

A MMS receives input from and transmits output to **devices.** Examples of MMS devices are user terminals, printers, and communication ports. Users enter commands via a terminal and most user output is displayed at the terminal. The PRINT commands and the SEND command result in output to printers and communication ports, respectively.

1.3. Intermediate Command Language

An important feature of this report is use of an Intermediate Command Language (ICL) to describe the set of user commands that MMS family members support. The ICL is designed so that the commands supported by any single family member can be described by some ICL subset. For examples of the ICL subsets associated with three family members (M0, M1, and M2), see [2].

1.3.1. ICL Overview

Each ICL command is an *abstract* description of a user command in that it specifies the command's user-visible effects without imposing unnecessary restrictions on either the command syntax or the physical characteristics of the user's terminal. The specific form a user command takes can vary from one family member to another. For example, a user command to display a message may take any one of the following forms:

- the user types the string "display message"
- the user types the string "show message"
- the user selects the menu item "DISPLAY MESSAGE"
- the user depresses a function key labeled "DISPLAY MESSAGE"
- the user clicks a mouse to display the message contained in the entry to which the cursor points

Such syntactic differences are not reflected in the ICL. Two user commands in different family members that have identical effects are associated with the same ICL command. Thus, if the result of each of the above user commands is identical, i.e., each causes the user terminal to display the specified message, each user command is associated with the same ICL command, namely, DISPLAY_MSG.

1.3.2. Special ICL Commands

The ICL treats the editing of some MMS entities and all access sets in either of the following ways. To initiate editing, the user may issue the command EDIT, which returns a copy of the item to be edited. The user may then apply various editing commands to modify the item as he desires. Once the user is satisfied with the edited version, he issues an UPDATE command, which changes the value of the item to that of the edited version. If the user is not satisfied with the edited version, he can omit issuing the UPDATE command, thus retaining the item's original value.

Alternatively, the user can omit explicit invocation of the EDIT command. He simply modifies a displayed data item and then issues the UPDATE command with one parameter pointing to the displayed data item and the other to the reference of the entity whose value is to be modified. With this approach, the displayed data item and the entity to be modified must both belong to the same data type.

The ICL excludes commands normally associated with a text editor, e.g., commands that insert text, delete text, search for text strings, and copy text from one area of the display to another. Although such commands may be security-relevant, e.g., copying data from a SECRET window to an UNCLASSIFIED window, they are beyond the current scope of the ICL.

The ICL provides two alternative approaches for removing message entries from message files. The first approach is based on the commands DELETEME, UNDELETEME, EXPUNGE, and MOVEME. With this approach, the user issues the DELETEME command to mark an

entry 'deleted'. He may subsequently UNDELETE the entry, which removes the 'deleted' mark, or EXPUNGE the message file, which removes all entries that are marked 'deleted' from the file. MOVEME creates a copy of the indicated entry, appends the copy to the specified file, and marks the original entry 'deleted'. The second approach uses REMOVEME and MOVE2ME. REMOVEME removes the indicated entry from the message file. The effect of MOVE2ME is identical to that of MOVEME, except the original entry is removed from the file rather than marked with a 'deleted' mark. We anticipate that a single MMS member will support one of these approaches but not both.

1.3.3. Implementing Simpler Versions of ICL Commands

A given family member may implement a simpler version of some of the ICL commands that it implements. Consider, for example, the ICL command, DISPLAY_MF, which takes three input parameters:

(1) a message file reference,

(2) a filter reference or a literal filter, and

(3) a form reference.

A family member that does not implement filters and forms can only support a simplified version of this command, e.g., a version with a single input parameter, a message file reference. Alternatively, a family member that implements filters but not forms may only support the first two input parameters of DISPLAY_MF, thus forcing every displayed message to have a standard format.

1.4. MMS Security Model

A MMS is required to enforce the rules of the MMS security model [5]. Thus a MMS must begin operation in a **secure state**, where secure state is as defined in the model. Each time a user issues an ICL command, one or more changes in system state can occur. However, these state changes can occur, i.e., the ICL command can be completed, only if the constraints of the security model are satisfied. One immediate consequence of enforcing the MMS security model is that each ICL command that displays the value of a MMS entity must also display the entity's classification. Thus, in the specifications below, all ICL commands that display an entity value have at least two output parameters: a value and a classification.

The MMS security model requires that all entities of a given type be treated as either objects or containers. An **object** is the smallest unit in the MMS that has a classification; it contains no other objects and cannot be multilevel. In contrast, a **container** has a classification but may contain objects and/or other containers each with its own classification. In a MMS, filters, forms, profiles, and some message fields (e.g., the To and Subject fields) are usually objects, while messages, the Text field of messages, message files, directories, and text files are usually containers.

2. Two Lists of the ICL Commands

This section contains two lists of the ICL commands associated with the MMS family. The first list is organized by data type, the second by generic command name. Appendix B contains a third list of the ICL commands organized alphabetically.

2.1. Organization by Data Type

Listed below are the ICL commands, grouped according to the following eleven data types: message, message file, text file, filter, form, terminal, printer, comment, directory, user profile, and user.* Also listed is the location of each command's specification.

*Because there are relatively few ICL commands that operate on message entries, e.g., DELETEME, this list as well as the list in Table 1 includes commands on message entries under the data type 'message file'.

Commands on Messages

Command Name	*Where Spec.*	*Command Name*	*Where Spec.*
CREATE_MSG	3.12.1	FORINFO_MSG	3.13.2
DISPLAY_MSG	3.2.2	FORACTION_MSG	3.15.1
PRINT_MSG	3.2.3	FORCOORD_MSG	3.14.1
EDIT_MSG	3.2.4	FORRELEASE_MSG	3.14.2
UPDATE_MSG	3.2.5	INSERT_MSG	3.11.2
DUP_MSG	3.11.1	DISPLAYAS_MSG	3.2.8
RECLASSIFY_MSG	3.2.7	PRINTAS_MSG	3.2.9
SEND_MSG	3.12.2	EDITAS_MSG	3.2.10
READDRESS_MSG	3.15.2	UPDATEAS_MSG	3.15.2
REPLY_MSG	3.13.1		

Commands on Message Files

CREATE_MF	3.3.1	EDITKEYME_MF	3.3.12
DESTROY_MF	3.2.1	UPDATEKEYME_MF	3.3.13
DISPLAY_MF	3.3.2	SORT_MF	3.3.14
PRINT_MF	3.3.3	DUP_MF	3.3.11
DELETEME_MF	3.3.4	RENAME_MF	3.2.12
UNDELETEME_MF	3.3.5	ADDNAME_MF	3.2.13
EXPUNGE_MF	3.3.6	RECLASSIFY_MF	3.2.7
COPYME_MF	3.3.7	DISPLAYAS_MF	3.2.8
MOVEME_MF	3.3.8	PRINTAS_MF	3.2.9
REMOVEME_MF	3.3.9	EDITAS_MF	3.2.10
MOVE2ME_MF	3.3.10	UPDATEAS_MF	3.2.11

Commands on Text Files

CREATE_TF	3.4.1	DUP_TF	3.2.6
DESTROY_TF	3.2.1	RENAME_TF	3.2.12
DISPLAY_TF	3.2.2	ADDNAME_TF	3.2.13
PRINT_TF	3.2.3	RECLASSIFY_TF	3.2.7
EDIT_TF	3.2.4	DISPLAYAS_TF	3.2.8
UPDATE_TF	3.2.5	PRINTAS_TF	3.2.9
COPYFRENT_TF	3.4.2	EDITAS_TF	3.2.10
COPYTOENT_TF	3.4.3	UPDATEAS_TF	3.2.11

Commands on Filters

CREATE_FILT	3.5.1	RENAME_FILT	3.2.12
DESTROY_FILT	3.2.1	ADDNAME_FILT	3.2.13
DISPLAY_FILT	3.2.2	RECLASSIFY_FILT	3.2.7
PRINT_FILT	3.2.3	DISPLAYAS_FILT	3.2.8
EDIT_FILT	3.2.4	PRINTAS_FILT	3.2.9
UPDATE_FILT	3.2.5	EDITAS_FILT	3.2.10
DUP_FILT	3.2.6	UPDATEAS_FILT	3.2.11

Commands on Forms

Command Name	*Where Spec.*	*Command Name*	*Where Spec.*
CREATE_FORM	3.6.1	RENAME_FORM	3.2.12
DESTROY_FORM	3.2.1	ADDNAME_FORM	3.2.13
DISPLAY_FORM	3.2.2	RECLASSIFY_FORM	3.2.7
PRINT_FORM	3.2.3	DISPLAYAS_FORM	3.2.8
EDIT_FORM	3.2.4	PRINTAS_FORM	3.2.9
UPDATE_FORM	3.2.5	EDITAS_FORM	3.2.10
DUP_FORM	3.2.6	UPDATEAS_FORM	3.2.11

Commands on Terminals

CREATE_TERM	3.10.1	MAXCLAS_TERM	3.10.4
DESTROY_TERM	3.2.1	DISPLAYAS_TERM	3.2.8
DISPLAY_TERM	3.10.2	PRINTAS_TERM	3.2.9
PRINT_TERM	3.10.3	EDITAS_TERM	3.2.10
RECLASSIFY_TERM	3.2.7	UPDATEAS_TERM	3.2.11

Commands on Printers

CREATE_PRNT	3.10.1	MAXCLAS_PRNT	3.10.4
DESTROY_PRNT	3.2.1	DISPLAYAS_PRNT	3.2.8
DISPLAY_PRNT	3.10.2	PRINTAS_PRNT	3.2.9
PRINT_PRNT	3.10.3	EDITAS_PRNT	3.2.10
RECLASSIFY_PRNT	3.2.7	UPDATEAS_PRNT	3.2.11

Commands on Comments

CREATE_CMT	3.7.1	DISPLAYAS_CMT	3.2.8
DESTROY_CMT	3.2.1	PRINTAS_CMT	3.2.9
EDIT_CMT	3.2.4	EDITAS_CMT	3.2.10
UPDATE_CMT	3.2.5	UPDATEAS_CMT	3.2.11
RECLASSIFY_CMT	3.2.7		

Commands on Directories

CREATE_DIR	3.9.1	DISPLAYAS_DIR	3.2.8
DESTROY_DIR	3.2.1	PRINTAS_DIR	3.2.9
DISPLAY_DIR	3.2.2	EDITAS_DIR	3.2.10
PRINT_DIR	3.2.3	UPDATEAS_DIR	3.2.11
RECLASSIFY_DIR	3.2.7		

Commands on User Profiles

Command Name	*Where Spec.*	*Command Name*	*Where Spec.*
CREATE_PROF	3.8.1	RECLASSIFY_PROF	3.2.7
DESTROY_PROF	3.2.1	DISPLAYAS_PROF	3.2.8
DISPLAY_PROF	3.2.2	PRINTAS_PROF	3.2.9
PRINT_PROF	3.2.3	EDITAS_PROF	3.2.10
EDIT_PROF	3.2.4	UPDATEAS_PROF	3.2.11
UPDATE_PROF	3.2.5		

Commands on Users

CREATE_USER	3.16.1	ADDAROLE_USER	3.16.7
DESTROY_USER	3.16.2	RMVAROLE_USER	3.16.8
DISPLAY_USER	3.16.3	ADDCROLE_USER	3.16.9
PRINT_USER	3.16.4	RMVCROLE_USER	3.16.10
CHGCLEAR_USER	3.16.5	LOGIN_USER	3.16.11
CHGPW_USER	3.16.6	LOGOUT_USER	3.16.12

2.2. Organization by Generic Command Name

ICL commands with the same or similar meanings share the same generic command name. In some cases, two commands with the same generic name have identical semantics (e.g., RECLASSIFY_MF and RECLASSIFY_TERM). In other cases, two commands with the same generic name have somewhat different semantics (e.g., DISPLAY_MF and DISPLAY_MFD). Table 1 lists the generic command names, indicates the data types for which each generic command is defined, and provides the location of the specification of the ICL command with the given generic command name and data type. A blank table entry indicates that the generic command is not defined on the given data type. The data types shown in Table 1 are identical to those shown above with one exception; commands on terminals and on printers are shown under the data type *device*.

In Table 1, related generic commands are listed together. Commands may be related because they are inverses (e.g., CREATE and DESTROY), because their semantics are similar (e.g., DISPLAY and PRINT), or because they are usually invoked sequentially (e.g., EDIT and UPDATE).

Table 1: ICL Commands Organized by Generic Command Name

COMMAND	*DATA TYPE*									
	MSG	*MF*	*TF*	*FILT*	*FORM*	*dev.*	*CMT*	*DIR*	*PROF*	*USER*
CREATE_	3.12.1	3.3.1	3.4.1	3.5.1	3.6.1	3.10.1	3.7.1	3.9.1	3.8.1	3.16.1
DESTROY_		3.2.1	3.2.1	3.2.1	3.2.1	3.2.1	3.2.1	3.2.1	3.2.1	3.16.2
DISPLAY_	3.2.2	3.3.2	3.2.2	3.2.2	3.2.2	3.10.2		3.9.2	3.2.2	3.16.3
PRINT_	3.2.3	3.3.3	3.2.3	3.2.3	3.2.3	3.10.3		3.9.3	3.2.3	3.16.4
EDIT_	3.2.4		3.2.4	3.2.4	3.2.4		3.2.4		3.2.4	
UPDATE_	3.2.5		3.2.5	3.2.5	3.2.5		3.2.5		3.2.5	
DELETEME_		3.3.4								
UNDELETEME_		3.3.5								
EXPUNGE_		3.3.6								
COPYME_		3.3.7								
MOVEME_		3.3.8								
REMOVEME_		3.3.9								
MOVE2ME_		3.3.10								
EDITKEYME_		3.3.12								
UPDATEKEYME_		3.3.13								
SORT_		3.3.14								
COPYFRENT_			3.4.2							
COPYTOENT_			3.4.3							
DUP_	3.11.1	3.3.11	3.2.6	3.2.6	3.2.6					
RECLASSIFY_	3.2.7	3.2.7	3.2.7	3.2.7	3.2.7	3.2.7	3.2.7	3.2.7	3.2.7	
MAXCLAS_						3.10.4				
SEND_	3.12.2									
READDRESS_	3.15.2									
REPLY_	3.13.1									
FORINFO_	3.13.2									
FORACTION_	3.15.1									
FORCOORD_	3.14.1									
FORRELEASE_	3.14.2									
INSERT_	3.11.2									
DISPLAYAS_	3.2.8	3.2.8	3.2.8	3.2.8	3.2.8	3.2.8	3.2.8	3.2.8	3.2.8	
PRINTAS_	3.2.9	3.2.9	3.2.9	3.2.9	3.2.9	3.2.9	3.2.9	3.2.9	3.2.9	
EDITAS_	3.2.10	3.2.10	3.2.10	3.2.10	3.2.10	3.2.10	3.2.10	3.2.10	3.2.10	
UPDATEAS_	3.2.11	3.2.11	3.2.11	3.2.11	3.2.11	3.2.11	3.2.11	3.2.11	3.2.11	
RENAME_		3.2.12	3.2.12	3.2.12	3.2.12					
ADDNAME_		3.2.13	3.2.13	3.2.13	3.2.13					
CHGCLEAR_										3.16.5
CHGPW_										3.16.6
ADDAROLE_										3.16.7
RMVAROLE_										3.16.8
ADDCROLE_										3.16.9
RMVCROLE_										3.16.10
LOGIN_										3.16.11
LOGOUT_										3.16.12

3. Command Specifications

3.1. Introduction

This section contains specifications for the ICL commands associated with the MMS family. Appendix B provides an alphabetical listing of the ICL commands and the location of each command's specifications.

3.1.1. Data Type Hierarchy

The set of ICL commands can be partitioned into two smaller sets: the first contains commands associated with entities (e.g., UPDATE_MSG), the second commands associated with users (e.g., CREATE_USER). The specifications of many commands in the first set differ only in the data types of their parameters. For example, the ICL commands, RECLASSIFY_MF and RECLASSIFY_TERM, have the same syntax and semantics, except the former operates on a message file, whereas the latter operates on a terminal. To simplify and shorten the specifications, we use a single form to define such commands.

To accomplish this, we have constructed the hierarchy of data types shown in Figure 1.* In the figure, an arrow pointing from data type *a* to data type *b* indicates that *b* inherits the properties of *a*. We refer to *a* as a **donor** of *b*. The most abstract data type in Figure 1, i.e., the data type with the fewest properties, is the highest in the hierarchy, namely, "entity". Every other data type shown possesses both its own associated properties and properties inherited from its donors. Thus the type "draft message" has the properties of "draft message", of "message", and of "entity". This means, for example, that a draft message has a Text field (because every message has a Text field) and that a draft message has a classification (because every entity has a classification).

Among the properties associated with each data type in Figure 1 is a set of ICL commands. In addition, each data type *potentially* inherits each ICL command associated with its donors. We say *potentially* because each data type below "entity" may only inherit *some* of the ICL commands associated with "entity". Consider, for example, the ICL commands, UPDATE and RENAME, both of which are associated with the data type "entity". In a given MMS, "draft message" may inherit UPDATE but cannot inherit RENAME, since a draft message may be modified but cannot be renamed. We note, however, that a data type always inherits *all* ICL commands associated with donors *other* than "entity".

Table 2 lists the data type "user" as well as each data type shown in Figure 1, the ICL commands with which the data type is associated, and the number of the subsection in which the ICL commands are specified. Within each subsection, the ICL commands are specified in the order presented in Table 1.

A few of the entries in Table 2, e.g., "informal draft message", have no associated ICL commands. Such data types inherit all of their ICL commands from donor data types. Thus, for example, "informal draft message" inherits the ICL commands associated with "draft message" and "message" along with the subset of ICL commands associated with "entity" that are inherited by "message" and "draft message".

The data type hierarchy in Figure 1 shows only two message types: informal and formal. The MMS family does not exclude message systems that support additional message types; a future family member may, like SIGMA [9], support three message types: informal messages, formal AUTODIN messages, and formal memoranda. Because we cannot anticipate all the additional message types that will be required, we have omitted other message types from Figure 1. For each family member that supports additional message types, the data hierarchy must be modified to include the types and an entry for each additional type must be added to Table 2. We assume that the set of commands associated with the new message types is a subset of the set of commands shown in Table 2 with slightly modified parameters (e.g., "formal memo ref" instead of "formal message ref", etc.).

*Because the hierarchy shown in Figure 1 includes only those MMS data types that are needed to specify the ICL commands, several MMS data types, e.g., paragraph and TO-field, do not appear in the figure.

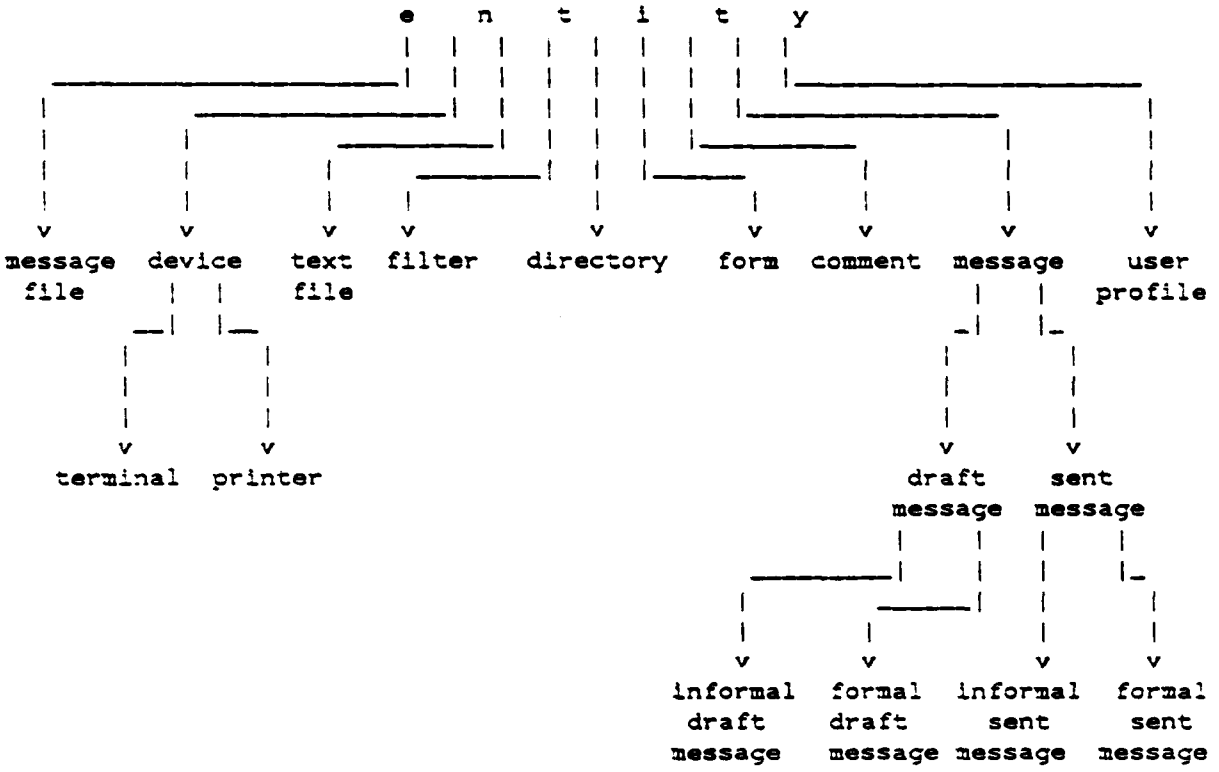


Figure 1: Hierarchy of MMS Entities

Table 2: MMS Data Types and Their Associated ICL Commands

Section	*Data Type*	*ICL Commands*
3.2	entity	DESTROY,DISPLAY,PRINT,EDIT,UPDATE, RECLASSIFY,DISPLAYAS,PRINTAS,EDITAS, UPDATEAS,DUP,RENAME,ADDNAME
3.3	message file	CREATE,DISPLAY,PRINT,DELETEME,UNDELETEME, EXPUNGE,SORT,COPYME,MOVEME,REMOVEME, MOVE2ME,DUP,EDITKEYME,UPDATEKEYME
3.4	text file	CREATE,COPYFRENT,COPYTOENT
3.5	filter	CREATE
3.6	form	CREATE
3.7	comment	CREATE
3.8	user profile	CREATE
3.9	directory	CREATE,DISPLAY,PRINT
3.10	device	CREATE,DISPLAY,PRINT,MAXCLAS
-	terminal	None
-	printer	None
3.11	message	DUP,INSERT
3.12	draft message	CREATE,SEND
3.13	sent message	REPLY,FORINFO
3.14	formal draft message	FORCOORD,FORRELEASE
3.15	formal sent message	FORACTION,READDRESS
-	informal draft message	None
-	informal sent message	None
3.16	user	CREATE,DESTROY,DISPLAY,PRINT,CHGCLEAR, CHGPW,ADDAROLE,RMVAROLE,ADDCROLE, RMVCROLE,LOGIN,LOGOUT

3.1.2. Form Used to Specify the ICL Commands

Each form lists the one or more ICL commands to which it applies and provides a generic name for the commands. The sections of the form named "Input Pars" and "Output Pars" indicate the command parameters. The user provides the input parameters along with the command name (the user interface may provide defaults for some of the input parameters). The output parameters identify the items that are displayed at the user terminal or output by a printer. Each parameter is expressed as "x:y", where x is the parameter and y is the data type or attribute type (e.g., classification, access set) of the parameter. The "Description" section gives a prose description of the command semantics. The "Constraints" section defines any constraints on the command's input parameters. In particular, this section identifies the specific data types that inherit the ICL commands associated with the data type "entity".

The successful completion of each ICL command requires that the preconditions described in the MMS security model and various message system preconditions are satisfied. This document does not include the security model preconditions nor a complete statement of the message system preconditions. A complete statement of the preconditions for the three family members defined in [2] will be included in a future report. However, we do list below three preconditions that *all* ICL commands must enforce:

(a) Each input parameter must have the type defined in the "Input Pars" section and must satisfy the conditions listed in the "Constraints" section.

(b) Each reference/userID supplied as an input parameter must refer to an existing entity/user.

(c) To invoke any command other than the LOGIN command, the user must be logged in to some terminal.

Any ICL command that does not satisfy the required preconditions cannot be completed.

The following abbreviations are used in the specifications:

Abbreviation	Meaning
ref	reference
char	character
setof	set of
val	value
seqof	sequence of

3.1.3. Names, References, and UserIDs

Although the specifications do not constrain the form of entity names and user names, we do assume that names (i.e., references and userIDs) satisfy the definitions in the MMS security model. Further, we treat names of entities and users that are to be created (i.e., do not yet exist) differently from names of existing entities and users. In the specifications below, the data type of an entity name is "ref" if the entity exists and "char string" if the entity does not exist. Similarly, the data type of a user name is "userID" if the user exists and "char string" if the user does not exist. Thus a name does not become a reference/userID until the MMS checks that the preconditions of the CREATE command are satisfied (e.g., that an entity/user with the specified name does not already exist) and until the new entity/user has been created.

As in the MMS security model, each entity name is either a direct reference or an indirect reference. An example of a direct reference is "imagen1", which names a particular printer. An example of an indirect reference is "heitmeyer:rumors", where "rumors" names a message file in the directory "heitmeyer".* In the specifications below, any entity name with data type "ref" or "char string" represents either a direct reference or an indirect reference. Whether an entity name can be a direct reference or an indirect reference depends on the data type of the entity. In most MMSs, for example, formal messages will have both direct and indirect references, whereas message files will have only indirect references.

To illustrate the above, we consider two sample user commands. "display printer imagen1" and "create mf [heitmeyer]rumors"*, which correspond to the ICL commands "DISPLAY_PRNT pname:printer ref" and "CREATE_MF mfname:char string".** The first command displays various information about a printer with the direct reference "imagen1". The second creates a new message file with indirect reference "[heitmeyer]rumors" and inserts the new file in the directory with reference "[heitmeyer]".

3.2. Commands on entity

3.2.1.

Generic Name	DESTROY	*Specific ICL Cmds*	DESTROY_MF DESTROY_TF DESTROY_TERM DESTROY_PRNT DESTROY_FILT DESTROY_FORM DESTROY_PROF DESTROY_CMT DESTROY_DIR
Input Pars	ename:entity ref		
Output Pars	-		
Constraints	The type of the entity **ename** is message file, text file, device, directory, filter, form, profile, or comment.		
Description	Destroys the entity **ename.** Removes the entity from any containers that contain it.		
Comment	The DESTROY command may not be applied to messages. Informal messages and draft formal messages disappear from the MMS when all references to them are removed. Formal messages never disappear from the MMS, since they are permanently recorded in INCOMING and OUTGOING.		

*The form of the example references and user commands used in this section is only one of many alternative forms.

**To simplify this example, we show only one parameter for each command.

3.2.2.

Generic Name	DISPLAY	*Specific ICL Cmds*	DISPLAY_MSG DISPLAY_TF DISPLAY_FILT DISPLAY_FORM DISPLAY_PROF
Input Pars	ename:entity ref g:form		
Output Pars	ent:entity val c:classification seqof(cmt:comment val,c1:classification,loc:location)		
Constraints	The type of the entity **ename** is message, text file, filter, form, or profile.		
Description	Displays the value **ent** and the classification **c** of the entity **ename.** For each comment attached to the entity **ename,** displays the comment's value **cmt** and its classification **c1** in the location **loc.** The user is not permitted to edit the displayed entity. The form **g** defines the format of the displayed information.		
Comments	The location **loc** may be associated with the entity to which the comment is attached. This document makes no assumptions about how **loc** is implemented.		

3.2.3.

Generic Name	PRINT	*Specific ICL Cmds*	PRINT_MSG PRINT_TF PRINT_FILT PRINT_FORM PRINT_PROF
Input Pars	A:seqof(ename:entity ref) g:form		
Output Pars	ent:entity val c:classification seqof(cmt:comment val,c1:classification,loc:location)		
Constraints	The type of the entity **ename** is message, text file, filter, form, or profile.		
Description	Prints the value **ent** and the classification **c** of the entity **ename.** For each comment attached to the entity **ename,** prints the comment's value **cmt** and its classification **c1** in the location **loc.** The form **g** defines the format of the printed information.		

3.2.4.

Generic Name	EDIT	*Specific ICL Cmds*	EDIT_MSG EDIT_TF EDIT_FILT EDIT_FORM EDIT_PROF EDIT_CMT
Input Pars	ename:entity ref		
Output Pars	ent:entity val c:classification		
Constraints	The type of the entity **ename** is draft message, text file, filter, form, comment, or profile.		
Description	Displays the value **ent** and the classification **c** of the entity **ename.** The user is permitted to edit the displayed entity.		

3.2.5.

Generic Name	UPDATE	*Specific ICL Cmds*	UPDATE_MSG UPDATE_TF UPDATE_FILT UPDATE_FORM UPDATE_PROF UPDATE_CMT
Input Pars	ename:entity ref ent:entity val		
Output Pars	-		
Constraints	The type of the entity **ename** is draft message, text file, filter, form, profile, or comment.		
Description	Sets the value of the entity **ename** to **ent.**		

3.2.6.

Generic Name	DUP	*Specific ICL Cmds*	DUP_TF DUP_FILT DUP_FORM
Input Pars	ename1:entity ref ename2:char string		
Output Pars	-		
Constraints	The type of the entity named **ename1** is text file, filter, or form. The new entity cannot be created if an entity with reference **ename2** and the same data type as **ename1** already exists.		
Description	Creates a new entity that is a duplicate of the entity **ename1.** The new entity has the reference **ename2** and the same type, value, classification, and access set as **ename1.** Inserts the new entity in the appropriate directory.		
Comments	Can be used to "copy" an entity from one directory to another.		

3.2.7.

Generic Name	RECLASSIFY	*Specific ICL Cmds*	RECLASSIFY_MF RECLASSIFY_MSG RECLASSIFY_TF RECLASSIFY_TERM RECLASSIFY_PRNT RECLASSIFY_FILT RECLASSIFY_FORM RECLASSIFY_PROF RECLASSIFY_CMT RECLASSIFY_DIR
Input Pars	ename:entity ref c:classification		
Output Pars	-		
Constraints	The type of the entity **ename** is message, message file, text file, device, directory, filter, form, profile, or comment.		
Description	Assigns the classification c to the entity **ename.**		
Comments	An issue is whether sent messages can be reclassified. In some MMSs, it may be desirable to allow this, since the original classification may be wrong OR after a certain amount of time, the message may be eligible for downgrading.		

3.2.8.

Generic Name	DISPLAYAS	*Specific ICL Cmds*	DISPLAYAS_MF DISPLAYAS_TF DISPLAYAS_MSG DISPLAYAS_FILT DISPLAYAS_FORM DISPLAYAS_TERM DISPLAYAS_PRNT DISPLAYAS_CMT DISPLAYAS_DIR DISPLAYAS_PROF
Input Pars	ename:entity ref g:form		
Output Pars	accset:access set		
Constraints	The type of the entity **ename** is message file, text file, message, directory, filter, form, device, profile, or comment.		
Description	Displays the access set **accset** of the entity **ename.** The user is not allowed to modify the access set. The form **g** defines the format of the displayed information.		

3.2.9.

Generic Name	PRINTAS	*Specific ICL Cmds*	PRINTAS_MF PRINTAS_TF PRINTAS_MSG PRINTAS_FILT PRINTAS_FORM PRINTAS_TERM PRINTAS_PRNT PRINTAS_CMT PRINTAS_DIR PRINTAS_PROF
Input Pars	A:seqof(entity ref) g:form		
Output Pars	accset:access set		
Constraints	Same as DISPLAYAS.		
Description	Prints the access set **accset** of the entity **ename.** The form **g** defines the format of the printed information.		

3.2.10.

Generic Name	EDITAS	*Specific ICL Cmds*	EDITAS_MF EDITAS_TF EDITAS_MSG EDITAS_FILT EDITAS_FORM EDITAS_TERM EDITAS_PRNT EDITAS_CMT EDITAS_DIR EDITAS_PROF
Input Pars	ename:entity ref		
Output Pars	accset:access set		
Constraints	Same as DISPLAYAS.		
Description	Displays the access set **accset** of the entity **ename.** The user is permitted to edit the displayed access set.		

3.2.11.

Generic Name	UPDATEAS	*Specific ICL Cmds*	UPDATEAS_MF UPDATEAS_TF UPDATEAS_MSG UPDATEAS_FILT UPDATEAS_FORM UPDATEAS_TERM UPDATEAS_PRNT UPDATEAS_CMT UPDATEAS_DIR UPDATEAS_PROF
Input Pars	ename:entity ref accset:access set		
Output Pars	-		
Constraints	Same as DISPLAYAS.		
Description	Modifies the access set of the entity **ename** to have the value **accset.**		

3.2.12.

Generic Name	RENAME	*Specific ICL Cmds*	RENAME_MF RENAME_TF RENAME_FILT RENAME_FORM
Input Pars	ename1:entity ref ename2:char string		
Output Pars	-		
Constraints	The type of the entity named **ename1** is message file, text file, filter, or form. The entity cannot be renamed if an entity with reference **ename2** and the same data type as **ename1** already exists.		
Description	Changes the name of the entity from **ename1** to **ename2.**		
Comments	Can be used to "move" an entity from one directory to another.		

3.2.13.

Generic Name	ADDNAME	*Specific ICL Cmds*	ADDNAME_MF ADDNAME_TF ADDNAME_FILT ADDNAME_FORM
Input Pars	ename1:entity ref ename2:char string		
Output Pars	-		
Constraints	The type of the entity named **ename1** is message file, text file, filter, or form. The new name for the entity cannot be created if an entity with reference **ename2** and the same data type as **ename1** already exists.		
Description	Creates an additional name **ename2** for the entity **ename1.**		
Comments	None.		

3.3. Commands on message file

3.3.1.

Generic Name	CREATE	*Specific ICL Cmds*	CREATE_MF
Input Pars	mfname:char string c:classification ccr:boolean as:access set		
Output Pars	-		
Constraints	The new message file cannot be created if a message file with reference **mfname** already exists. The string **mfname** identifies the directory in which the new message file is to be inserted.		
Description	Creates a message file with reference **mfname**, classification **c**, CCR mark **ccr**, and access set **as**. The new message file contains no entries. Inserts message file **mfname** in the specified directory.		

3.3.2.

Generic Name	DISPLAY	*Specific ICL Cmds*	DISPLAY_MF
Input Pars	mfname:message file ref f:filter g:form		
Output Pars	seqof(v:message entry val.c:classification) seqof(cmt:comment val.c1:classification.loc:location)		
Constraints	None.		
Description	Displays the value **v** and the classification **c** of all message entries in **mfname** that satisfy the filter **f**. Also displays the value **cmt** and the classification **c1** (in the location **loc**) of each comment attached to one of these entries. The form **g** defines the format of the displayed information.		

3.3.3.

Generic Name	PRINT	*Specific ICL Cmds*	PRINT_MF
Input Pars	mfname:message file ref f:filter g:form		
Output Pars	seqof(v:message entry val,c:classification) seqof(cmt:comment val,cl:classification,loc:location)		
Constraints	None.		
Description	Prints the value **v** and the classification **c** of all message entries in **mfname** that satisfy the filter **f.** For each entry in **mfname** that satisfies the filter, also prints the value **cmt** and the classification **cl** (in the location **loc**) of each attached comment. The format of the printed information is defined by the form **g.**		

3.3.4.

Generic Name	DELETEME	*Specific ICL Cmds*	DELETEME_MF
Input Pars	A:seqof(message entry ref) f:filter		
Output Pars	-		
Constraints	None.		
Description	Marks 'deleted' each message entry in **A** that satisfies the filter **f.**		

3.3.5.

Generic Name	UNDELETEME	*Specific ICL Cmds*	UNDELETEME_MF
Input Pars	A:seqof(message entry ref) f:filter		
Output Pars	-		
Constraints	None.		
Description	Removes the 'deleted' mark from each message entry in **A** that satisfies the filter **f.**		

3.3.6.

Generic Name	EXPUNGE	*Specific ICL Cmds*	EXPUNGE_MF
Input Pars	mfname:message file ref		
Output Pars	-		
Constraints	None.		
Description	Removes all message entries marked 'deleted' from the message file **mfname.** Maintains the order of the remaining message entries.		

3.3.7.

Generic Name	COPYME	*Specific ICL Cmds*	COPYME_MF
Input Pars	A:seqof(message entry ref) f:filter mfname:message file ref		
Output Pars	-		
Constraints	None.		
Description	Appends a copy of each message entry in **A** that satisfies the filter **f** to the message file **mfname.** Each new entry in **mfname** is marked 'new'.		

3.3.8.

Generic Name	MOVEME	*Specific ICL Cmds*	MOVEME_MF
Input Pars	A:seqof(message entry ref) f:filter mfname:message file ref		
Output Pars	-		
Constraints	None.		
Description	Appends a copy of each message entry in **A** that satisfies the filter **f** to the message file **mfname.** Marks 'deleted' each message entry in **A** that is copied to **mfname.** Each new entry in **mfname** is marked 'new'.		

3.3.9.

Generic Name	REMOVEME	*Specific ICL Cmds*	REMOVEME_MF
Input Pars	A:seqof(message entry ref) f:filter		
Output Pars	-		
Constraints	None.		
Description	Removes each message entry in **A** that satisfies the filter **f** from the file in which it is contained. Maintains the order of the remaining message entries.		

3.3.10.

Generic Name	MOVE2ME	*Specific ICL Cmds*	MOVE2ME_MF
Input Pars	A:seqof(message entry ref) f:filter mfname:message file ref		
Output Pars	-		
Constraints	None.		
Description	Appends a copy of each message entry in **A** that satisfies the filter **f** to the message file **mfname.** Removes each message entry that is copied from the file in which it is contained.		

3.3.11.

Generic Name	DUP	*Specific ICL Cmds*	DUP_MF
Input Pars	mfname1:message file ref mfname2:char string		
Output Pars	-		
Constraints	The new message file cannot be created if a message file with reference **mfname2** already exists in the user's message file directory.		
Description	Creates a new message file that is a duplicate of the message file **mfname1.** The new message file has the reference **mfname2** and the same classification, CCR-mark, and access set as **mfname1.** Creates a copy of each entry in message file **mfname1** and inserts the entry in message file **mfname2.** New copies of the messages associated with the entries are NOT created. The new entries point to the same message copies as the existing entries. Inserts the new message file in the user's message file directory.		

3.3.12.

Generic Name	EDITKEYME	*Specific ICL Cmds*	EDITKEYME_MF
Input Pars	mename:message entry ref		
Output Pars	setof(k:keyword val,c:classification)		
Constraints	None.		
Description	Displays the value **k** and the classification **c** of each keyword associated with the message entry **mename.** The user is permitted to edit the set of keywords.		
Comments	1. Some family members may treat the set of keywords as an object rather than a container (not consistent with the definition above).		

3.3.13.

Generic Name	UPDATEKEYME	*Specific ICL Cmds*	UPDATEKEYME_MF
Input Pars	mename:message entry ref A:setof(keyword val,classification)		
Output Pars	-		
Constraints	None.		
Description	Assigns the set of keywords **A** and their associated classifications to the message entry named **mename.**		

3.3.14.

Generic Name	SORT	*Specific ICL Cmds*	SORT_MF
Input Pars	mfname:message file ref c:criterion		
Output Pars	-		
Constraints	None.		
Description	Reorders the message entries in **mfname** according to the criterion **c**. For example, the user may reorder the entries using a criterion that puts the entries in DTG order, most recent DTG last.		

3.4. Commands on text file

3.4.1.

Generic Name	CREATE	*Specific ICL Cmds*	CREATE_TF
Input Pars	tname:char string c:classification as:access set		
Output Pars	tf:text file val c:classification		
Constraints	The new text file cannot be created if a text file with reference **tname** already exists. The string **tname** identifies the directory in which the new text file is to be inserted.		
Description	Creates a text file with reference **tname**, classification **c**, and access set **as.** Inserts text file **tname** in the specified directory. Displays the value **tf** of the new text file and its classification **c**. The user is permitted to edit the displayed text file.		

3.4.2.

Generic Name	COPYFRENT	*Specific ICL Cmds*	COPYFRENT_TF
Input Pars	ename:entity ref tname:text file ref		
Output Pars	-		
Constraints	None.		
Description	Appends a copy of the entity **ename** to the text file **tname.** If the entity **ename** is an object, the new entity has the same value and classification as **ename** but has 'paragraph' as its data type. If the entity **ename** is a sequence of objects, each new object has the same value and classification as its counterpart object but has 'paragraph' as its data type.		

3.4.3.

Generic Name	COPYTOENT	*Specific ICL Cmds*	COPYTOENT_TF
Input Pars	tname:text file ref ename:entity ref		
Output Pars	-		
Constraints	None.		
Description	Appends a copy of each object in **tname** to the entity **ename.** Each new object retains the value and classification of the corresponding object in **tname.** The data type of each new object may have a different data type than the corresponding object in **tname,** since the object type must be consistent with the entity into which the new object is being inserted.		

3.5. Commands on filter

3.5.1.

Generic Name	CREATE	*Specific ICL Cmds*	CREATE_FILT
Input Pars	fname:char string c:classification as:access set		
Output Pars	filt:filter val c:classification		
Constraints	The new filter cannot be created if a filter with reference **fname** already exists. The string **fname** identifies the directory in which the new filter is to be inserted.		
Description	Creates a filter with reference **fname**, classification **c**, and access set **as.** Inserts the filter **fname** in the specified directory. Displays the value **filt** and its classification **c.** The user is permitted to edit the displayed filter. The initial value of the filter is null.		

3.6. Commands on form

3.6.1.

Generic Name	CREATE	*Specific ICL Cmds*	CREATE_FORM
Input Pars	fname:char string c:classification ftype:form type as:access set		
Output Pars	fval:form val c:classification		
Constraints	The new form cannot be created if a form with reference **fname** already exists. The string **fname** identifies the directory in which the new form is to be inserted.		
Description	Creates a form with reference **fname**, classification **c**, form type **ftype**, and access set **as.** Inserts the form **fname** in the specified directory. Displays the value **fval** and its classification c. The user is permitted to edit the displayed form. (The MMS may assign a default, nonempty value to the newly created form.)		

3.7. Commands on comment

3.7.1.

Generic Name	CREATE	*Specific ICL Cmds*	CREATE_CMT
Input Pars	ename:entity reference c:classification as:access set loc:location		
Output Pars	cval:comment val c:classification		
Constraints	None.		
Description	Attaches a comment to the entity **ename.** The comment has the classification **c** and access set **as.** Displays the comment's value **cval** and its classification **c.** The user is permitted to edit the displayed comment (initially empty). Some family members may allow the user to indicate the location **loc** of the comment within the entity.		

3.8. Commands on user profile

3.8.1.

Generic Name	CREATE	*Specific ICL Cmds*	CREATE_PROF
Input Pars	pname:profile ref c:classification		
Output Pars	pval:profile val c:classification		
Constraints	Only the system security officer can create profiles.		
Description	Creates a profile named **pname** with classification **c**. Displays the value **pval** of the profile and its classification **c**. The user is permitted to edit the displayed profile.		
Comment	1. The purpose of this command is to allow the security officer to define more than a single user profile. 2. Since there are no directories for user profiles, there must be a way that the security officer can display the names of all existing user profiles. I assume that this can be handled by the user interface; e.g., when the security officer provides a "?" as an argument for DISPLAY_PROF, the user interface displays the names of all existing profiles and then prompts the user for the appropriate one.		

3.9. Commands on directory

3.9.1.

Generic Name	CREATE	*Specific ICL Cmds*	CREATE_DIR
Input Pars	dname:char string c:classification as:access set		
Output Pars	-		
Constraints	The new directory cannot be created if a directory with reference **dname** already exists. The string **dname** specifies the directory in which the new directory is to be inserted. A new directory must always be inserted in an existing directory.		
Description	Creates a new directory named **dname** with classification **c** and access set **as.** Inserts the new directory in the directory specified in **dname.**		

3.9.2.

Generic Name	DISPLAY	*Specific ICL Cmds*	DISPLAY_DIR
Input Pars	dname:directory ref g:form		
Output Pars	c:classification seqof (r:ref,cl:classification,t:type[,ccr:boolean]) seqof(cmt:comment val,c1:classification,loc:location)		
Constraints	None.		
Description	Displays the name, classification, and data type of each entity in the directory named **dname.** If the entity in the directory is of type "message file", displays the CCR mark of the entity. Displays the classification **c** of the directory. Displays each comment **cmt** attached to the directory and its classification **c1** in the location **loc.** The form **g** defines the format of the displayed information.		

3.9.3.

Generic Name	PRINT	*Specific ICL Cmds*	PRINT_DIR
Input Pars	dname:directory ref g:form		
Output Pars	c:classification seqof (r:ref.cl:classification,t:type,ccr:boolean}) seqof(cmt:comment val,cl:classification,loc:location)		
Constraints	None.		
Description	Prints the name, classification, and data type of each entry in the directory named **dname.** If the entry is of type "message file", prints the CCR mark of the entry. Prints the classification **c** of the directory. Prints the value **cmt** and the classification **cl** (in the location **loc**) of each comment attached to the directory. The form **g** defines the format of the printed information.		

3.10. Commands on device

3.10.1.

Generic Name	CREATE	*Specific ICL Cmds*	CREATE_TERM CREATE_PRNT
Input Pars	dname:char string c:classification as:access set		
Output Pars	-		
Constraints	The new device cannot be created if a device with reference **dname** already exists.		
Description	Creates a device with reference **dname**, maximum classification **c**, and access set **as.**		

3.10.2.

Generic Name	DISPLAY	*Specific ICL Cmds*	DISPLAY_TERM DISPLAY_PRNT
Input Pars	A:seqof(device ref) g:form		
Output Pars	seqof(dname:device ref,c1:classification,c2:classification)		
Constraints	None.		
Description	For each device in **A**, displays the device reference **dname**, the maximum classification **c1**, and the current classification **c2**. The form **g** defines the format of the displayed information.		
Comment	There should be a version of this command that allows the user to display this information for all devices of the given device type, e.g., DISPLAY_TERM ALL.		

3.10.3.

Generic Name	PRINT	*Specific ICL Cmds*	PRINT_TERM PRINT_PRNT
Input Pars	A:seqof(device ref) g:form		
Output Pars	seqof(dname:device ref,c1:classification,c2:classification)		
Constraints	None.		
Description	For each device in **A**, prints the device reference **dname**, the maximum classification **c1**, and the current classification **c2**. The form **g** defines the format of the printed information.		
Comment	Same comment as in 3.10.2.		

3.10.4.

Generic Name	MAXCLAS	*Specific ICL Cmds*	MAXCLAS_TERM MAXCLAS_PRNT
Input Pars	dname:device ref c:classification		
Output Pars	-		
Constraints	None.		
Description	Sets the maximum classification of the device **dname** to **c.**		

3.11. Commands on message

3.11.1.

Generic Name	DUP	*Specific ICL Cmds*	DUP_MSG
Input Pars	msgid:message ref mfname:message file ref		
Output Pars	dmsg:draft message val c:classification		
Constraints	None.		
Description	Creates a copy of the message **msgid.** The new message has a new reference in its ID field and the same classification, the same message type (e.g., formal or informal), and, if the original message is a draft message, the same access set as the original message; the userID or role of the user who created the message is in the From field; and the creating user's site is in the Originator field. Creates a message entry for the new message and appends it to the message file **mfname.** The new entry is marked 'new'. Displays both the value **dmsg** and the classification **c** of the new message. The user is permitted to modify the displayed message.		

3.11.2.

Generic Name	INSERT	*Specific ICL Cmds*	INSERT_MSG
Input Pars	msgid:direct message ref mfname:message file ref		
Output Pars	-		
Constraints	None.		
Description	Creates a message entry for the message with direct reference **msgid** and appends the entry in the message file named **mfname.** The new entry is marked 'new'.		
Comments	A user may have only a direct reference for a message. This commands permits the user to append an entry for the message to the specified message file.		

3.12. Commands on draft message

3.12.1.

Generic Name	CREATE	*Specific ICL Cmds*	CREATE_MSG
Input Pars	t:message type c:classification as:access set mfname:message file ref		
Output Pars	dmsg:draft message val c:classification		
Constraints	None.		
Description	Creates a draft message of message type **t,** classification **c,** and access set **as.** Creates a message entry for the new message and appends it to the message file **mfname.** The new entry is marked 'new'. Displays the value **dmsg** and the classification **c** of the new message. The new message has a reference assigned by the MMS in its ID field, the userID or role of the user who created the message in its From field, and the user's site in its Originator field. The user is permitted to edit the displayed message.		

3.13. Commands on sent message

3.13.1.

Generic Name	REPLY	*Specific ICL Cmds*	REPLY_MSG
Input Pars	msgid:sent message ref t:message type c:classification as:access set mfname:message file ref		
Output Pars	dmsg:draft message val c:classification		
Constraints	None.		
Description	Creates a message of message type **t**, classification **c**, and access set **as**. Creates a message entry for the message and appends it to the message file **mfname**. The new entry is marked 'new'. Displays the value **dmsg** of the new message. The new message has a reference assigned by the MMS in its ID field, the userID or role of the user who created the message in its From field, the contents of the From field of the message **msgid** in the To field, and the same Subject field as **msgid**. Also displays the message classification **c**. The user is allowed to edit the new message.		

3.13.2.

Generic Name	FORINFO	*Specific ICL Cmds*	FORINFO_MSG
Input Pars	B:setof(sent message ref) A:setof(addressee)		
Output Pars	-		
Constraints	None.		
Description	For formal messages, forwards for "info" each message in **B** to each addressee in **A**. For informal messages, forwards each message in **B** to each addressee in **A**. For each message in **B** and each user in **A**, creates a message entry for the message and appends the entry to the user's inbox. Each entry has the 'for info' mark and a 'new' mark.		

3.14. Commands on formal draft message

3.14.1.

Generic Name	FORCOORD	*Specific ICL Cmds*	FORCOORD_MSG
Input Pars	B:setof(formal draft message ref) A:setof(addressee)		
Output Pars	-		
Constraints	None.		
Description	Forwards each formal draft message in **B** to each addressee in **A**. The data type of the forwarded messages does not change. For each message in **B** and each user in **A**, creates an entry for the message and appends it to the user's inbox. Each entry has the 'for coord' mark and a 'new' mark. Each addressee may either send comments on the message via another message or, if the addressee has "update" permission, he may modify the message.		

3.14.2.

Generic Name	FORRELEASE	*Specific ICL Cmds*	FORRELEASE_MSG
Input Pars	msgid:formal draft message ref a:addressee		
Output Pars	-		
Constraints	None.		
Description	The formal draft message **msgid** is forwarded to the addressee **a** for release. The user who receives the message "for release" has a new entry for the message in his inbox; the entry has the 'for release' mark and a 'new' mark.		

3.15. Commands on formal sent message

3.15.1.

Generic Name	FORACTION	*Specific ICL Cmds*	FORACTION_MSG
Input Pars	B:setof(formal sent message ref) A:setof(addressee)		
Output Pars	-		
Constraints	None.		
Description	Forwards for "action" each formal sent message in **B** to each addressee in **A.** For each message in **B** and each user in **A,** creates an entry for the message and appends it to the user's inbox. Each entry has the 'for action' mark and a 'new' mark.		

3.15.2.

Generic Name	READDRESS	*Specific ICL Cmds*	READDRESS_MSG
Input Pars	msgid:formal sent message ref mfname:message file ref		
Output Pars	dmsg:draft message val c:classification		
Constraints	None.		
Description	Creates a copy of the sent formal message **msgid.** The new formal draft message has the same value as **msgid,** namely, **dmsg;** the same classification as **msgid,** namely, **c;** a reference assigned by the MMS in its ID field; and the same message type as **msgid.** Creates a message entry for the new message and appends it to the message file **mfname.** The user is only permitted to fill in the address fields of the new message.		

3.16. Commands on user

3.16.1.

Generic Name	CREATE	*Specific ICL Cmds*	CREATE_USER
Input Pars	uid:char string pw:password cl:clearance A:setof(role) pname:profile ref		
Output Pars	-		
Constraints	The new user cannot be created if there already exists a user with userID **uid.**		
Description	Creates a user with userID **uid,** password **pw,** clearance **cl,** profile **pname,** and the authorized roles in **A.** Also creates a message file directory, a text file directory, a filter directory, a form directory, an inbox, and initial access sets for the directories and the inbox. The new user cannot be created if there already exists a user with userID **uid.**		

3.16.2.

Generic Name	DESTROY	*Specific ICL Cmds*	DESTROY_USER
Input Pars	uid:userID		
Output Pars	-		
Constraints	None.		
Description	Destroys the user **uid.** Also destroys the user's message file directory, text file directory, filter directory, form directory, and inbox. Any message files, text files, filters, or forms that are solely in this user's directories are also destroyed.		

3.16.3.

Generic Name	DISPLAY	*Specific ICL Cmds*	DISPLAY_USER
Input Pars	A:seqof(userID) g:form		
Output Pars	seqof(uid:userID,cl:clearance,A:setof(role), B:setof(role),tname:terminal ref)		
Constraints	None.		
Description	For each user in **A**, displays the userID **uid**, clearance **cl**, set of authorized roles **A**, and, if **uid** is logged on, his set of current roles **B** and the terminal **tname** that he is logged onto. Only the system security officer is permitted to execute this command. The form **g** defines the format of the displayed information.		

3.16.4.

Generic Name	PRINT	*Specific ICL Cmds*	PRINT_USER
Input Pars	A:seqof(userID) g:form		
Output Pars	seqof(uid:userID,cl:clearance,A:setof(role), B:setof(role),tname:terminal ref)		
Constraints	None.		
Description	For each user in **A**, prints the userID **uid**, clearance **cl**, the set of authorized roles **A**, and, if **uid** is logged on, his set of current roles **B** and the terminal **tname** that he is logged onto. Only the system security officer is permitted to execute this command. The form **g** defines the format of the printed information.		

3.16.5.

Generic Name	CHGCLR	*Specific ICL Cmds*	CHGCLR_USER
Input Pars	uid:userID cl:clearance		
Output Pars	-		
Constraints	None.		
Description	Changes the clearance of the user **uid** to **cl.**		

3.16.6.

Generic Name	CHGPW	*Specific ICL Cmds*	CHGPW_USER
Input Pars	uid:userID oldpw:password newpw:password		
Output Pars	-		
Constraints	A precondition for this command is that the old value of the password is **oldpw.** (The system security officer need not provide the old password.)		
Description	Changes the password of the user **uid** to **newpw.**		

3.16.7.

Generic Name	ADDAROLE	*Specific ICL Cmds*	ADDAROLE_USER
Input Pars	uid:userID A:setof(role)		
Output Pars	-		
Constraints	None.		
Description	Adds the roles in **A** to the set of roles authorized for the user **uid.**		

3.16.8.

Generic Name	RMVAROLE	*Specific ICL Cmds*	RMVAROLE_USER
Input Pars	uid:userID A:setof(role)		
Output Pars	-		
Constraints	None.		
Description	Removes the roles in **A** from the set of roles authorized for the user **uid.**		

3.16.9.

Generic Name	ADDCROLE	*Specific ICL Cmds*	ADDCROLE_USER
Input Pars	uid:userID A:setof(role)		
Output Pars	-		
Constraints	None.		
Description	Adds the roles in **A** to the set of current roles authorized for the user **uid.**		

3.16.10.

Generic Name	RMVCROLE	*Specific ICL Cmds*	RMVCROLE_USER
Input Pars	uid:userID A:setof(role)		
Output Pars	-		
Constraints	None.		
Description	Removes the roles in **A** from the set of current roles authorized for the user **uid.**		

3.16.11.

Generic Name	LOGIN	*Specific ICL Cmds*	LOGIN_USER
Input Pars	tname:terminal ref uid:userID pw:password c:classification A:setof(role)		
Output Pars	-		
Constraints	None.		
Description	Logs the user with userID **uid** and password **pw** onto the terminal **tname.** Sets the classification of the terminal to **c.** Assigns the roles in **A** as the current roles assigned to the user. Fixes the access set of **tname** so that the user **uid** can RECLASSIFY the terminal and LOGOUT of the terminal.		

3.16.12.

Generic Name	LOGOUT	*Specific ICL Cmds*	LOGOUT_USER
Input Pars	uid:userID tname:terminal ref		
Output Pars	-		
Constraints	None		
Description	Logs the user with userID **uid** off of the terminal **tname.** Fixes the access set of the terminal so that the user **uid** can no longer RECLASSIFY or LOGOUT of the terminal. Removes all message entries marked 'deleted' from each of the user's message files. For each entry that is marked 'new', removes the 'new' mark. Sets the user's current role set to the empty set.		

4. Glossary

This section provides definitions for terms used in this report. In particular, terms that appear as parameters to ICL commands are defined. Many of the definitions have been extracted from the MMS security model [5] and references [2-4].

access set: a set of triples associated with an entity. Each triple consists of a userID or role, an operation, and an operand index. If a given operation requires more than one operand, the operand index specifies the position in which a reference to this entity may appear as an operand. The existence of a particular triple in the access set implies that a user corresponding to the userID or role is authorized to invoke the specified operation on the entity with which the access set is associated.

address field: a field of a message that contains addresses. The To field, the CC field, and the From field are examples of address fields.

addressee: a userID, a role, or an organization to whom a message can be sent.

authorized role: a role that the user is authorized for. The system security officer assigns the user's set of authorized roles.

classification: a designation reflecting the damage that could be caused by unauthorized disclosure of information. Includes a sensitivity level (UNCLASSIFIED, CONFIDENTIAL, SECRET, TOP SECRET) and a set of zero or more compartments (CRYPTO, NUCLEAR, etc.).

clearance: the degree of trust associated with a person, expressed in the same way as a classification. In a secure MMS, each user will have a clearance.

command script: a named sequence of user commands. To tailor the MMS to his individual requirements, a user may define a script that comprises a frequently invoked sequence of user commands. The name and definition of a script are included in the user's profile.

comment: text that is attached to an entity. More than one comment may be attached to a single entity. A user obtains access to a comment by means of the entity to which the comment is attached.

container: a unit of information that has a classification and that may contain objects and/or other containers each with their own classification. Unlike an object, a container can be multilevel.

CCR: an attribute of a container. CCR is an abbreviation for Container Clearance Required. CCR containers require that a user who wishes to view information in a container have a clearance that exceeds or equals the classification of the container.

current role: a role that the user has at a given time. The user's set of current roles is a subset of his set of authorized roles. The user defines his set of current roles.

device: a place where a MMS reads or writes information. Examples of MMS devices are user terminals, printers, and communication ports.

directory: a set each of whose elements is a named entity or another directory.

draft message: a message in draft form. Draft messages are messages that users create and edit. The SEND_MSG command converts a draft message into a sent message, distributing the latter to its addressees.

entity: an object or a container.

filter: a set of criteria that can be applied to message entries. To satisfy a filter, a message entry must satisfy the filter's criteria. Filters are used to retrieve messages from message files or message file subsets. An example of a *named* filter is ALL: every message entry satisfies the filter ALL. A filter may also be expressed as a literal, e.g., "ORIG=CINCPAC and SUBJ=IRAN".

form: a description of output format for entities of a given type. A form has either a system-provided or a user-provided name. By defining a form of a given type, a user can change the output formats of entities of that type from the standard MMS formats to a specially defined format.

formal message: a class of messages that are exchanged by military organizations rather than by individuals. Such messages are always "on the record" and are stored for lengthy periods after their transmission. They contain special fields, such as Info and Precedence. Special operations, e.g., FORACTION_MSG, FORINFO_MSG, and READDRESS_MSG, may be applied to formal messages. Only users authorized for the role "releaser" can release a formal message.

inbox: a message file in which the MMS inserts each message that is sent to a given user. Every user has an inbox.

informal message: a class of messages that are exchanged by individuals. Such messages are "off the record" and there is no official requirement for retaining copies. They have fewer fields than formal messages and fewer operations can be applied to them.

keyword: one or more words that can be inserted in a message entry. MMS members that support keywords allow users to associate a set of keywords with each message entry. Keywords are used to define search criteria in filters.

message: a set of fields, including To, From, Subject, and Text. A message is either a draft message or a sent message. Moreover, every message has a message type, e.g., formal.

message entry: consists of a message and status information about the message, e.g., whether the message is 'for action', 'for release', etc. In some MMS members, a message entry may also include a set of keywords.

message file: a sequence of message entries.

message type: an attribute of a message that determines the fields that the message contains, the set of operations that can be applied to the message, and whether the message is "on the record". In many MMSs, there are two message types: formal and informal.

object: the smallest unit of information in the MMS that has a classification. An object contains no other objects and cannot be multilevel.

password: a character string that is used to restrict usage of a userID.

printer: a device that provides MMS users with hardcopy output.

reference: a name for an entity. The reference may be direct, e.g., a date-time-group coupled with an originator. A reference may also be indirect, e.g., the "current message's Text field's third paragraph."

role: the job that the user is performing, such as downgrader, releaser, etc. A user is always associated with at least one role at any instant, and the user can change roles during a session. The system security officer assigns the user's set of authorized roles. Whenever the user is logged in, his set of current roles specifies the roles that are valid at the time.

sent message: a message that has been released. The SEND_MSG command converts a draft message into a sent message.

sort criterion: a criterion that is used to order the message entries in a file. For example, a sort criterion named DTG may be used to order the entries according to Date-Time-Group, most recent DTG last.

terminal: the device onto which the user logs in to use the MMS. Most MMS output is displayed on the user's terminal.

text file: a sequence of paragraphs. A user may save text, address lists, and other miscellaneous information in text files.

user: a person authorized to use the MMS.

userID: a character string used to denote a user of the system. To use the MMS, a person presents a userID to the system, and the system authenticates that the person is the user corresponding to that userID. Each user has a unique userID.

user profile: a set that determines MMS operation for the user who owns the profile. Each new user is assigned a standard user profile that he can later modify to suit his individual requirements. A profile defines the values of default parameters and names and defines command scripts.

APPENDIX A

This appendix compares this report and reference [2]. Each section of the report is listed below. A section is labeled "OLD," if it is extracted without change from [2]; "REVISED," if it is a revision of material in [2]; and "NEW," if the section is not taken from [2].

SECTION	*DESCRIPTION*
Introduction	NEW
1.	NEW
1.1-1.2	NEW
1.2.1	OLD except for REVISED list of data types
1.2.2	OLD except for 3 last sentences which are NEW
1.2.4-1.2.10	NEW
1.3	OLD
1.3.1	OLD
1.3.2	NEW
1.3.3	NEW
1.4	OLD except for examples in last sentence
2.	OLD
2.1	REVISED to include NEW data types and user cmds
2.2	intro.: OLD Table 1: REVISED to include NEW cmds and data types
3.1.1	first 5 paragraphs: OLD 6th paragraph: NEW
Fig. 1	REVISED to include NEW data types
Table 2	REVISED to include NEW data types and user commands
3.1.2	OLD, except for addition of "seqof" to the abbreviations
3.2.1-3.2.11	REVISED to include more user cmds
3.2.12-3.2.13	NEW
3.3.1	OLD
3.3.2-3.3.5	REVISED
3.3.6	OLD
3.3.7-3.3.8	REVISED
3.3.9	OLD
3.3.10-3.3.14	NEW
3.4	OLD
3.5-3.8	NEW
3.9.1	NEW
3.9.2-3.9.3	REVISED
3.10	OLD except for addition of NEW cmds on printer
3.11.1	OLD
3.11.2	NEW
3.12	OLD
3.13.1	OLD

SECTION	DESCRIPTION
3.13.2	REVISED
3.14.1	REVISED
3.14.2	OLD
3.15.1	REVISED
3.15.2	OLD
3.16.1-3.16.4	REVISED
3.16.5-3.16.12	OLD
4.	OLD, except for the following definitions: comment:NEW device:NEW directory:REVISED filter:NEW form:NEW keyword:NEW message entry:REVISED printer:NEW sort criterion:NEW user profile:NEW and the deletion of the definitions for message file directory and text file directory

APPENDIX B

This section provides an alphabetical index to the ICL command specifications.

Acknowledgments

The author is grateful to Mark Cornwell, Rob Jacob, and Carl Landwehr for their valuable comments on earlier drafts of this report.

REFERENCES

[1] D. Brotz, "Laurel Manual," Rep. CSL-81-6, XEROX Palo Alto Research Center, Palo Alto, Calif., May 1981.

[2] C. Heitmeyer and M. Cornwell, "Specifications for Three Members of the Military Message System (MMS) Family," Memo Rep. 5645, Naval Res. Lab., Wash., DC, Sep. 1985. ADA158455

[3] C. Heitmeyer and S. Wilson, "Military Message Systems: Current Status and Future Directions," *IEEE Trans. on Comm.*, Vol., COM-28, No. 9, Sept. 1980, pp. 1645-1654.

[4] C. Heitmeyer, "An Intermediate Command Language for Military Message Systems," NRL tech memo 7590-450:CH:ch, Nov. 1981.

[5] C. Landwehr, C. Heitmeyer, and J. McLean, "A Security Model for Military Message Systems," *ACM Trans. on Computer Systems*, Vol. 2, No. 3, Aug. 1984. p. 198-222.

[6] C. Mooers, "The HERMES Guide," BBN Rep. 4995, Bolt Beranek and Newman, Cambridge, Mass., Aug. 1982.

[7] "NMIC Support System User Manual," Planning Research Corp./Inform. Sci. Co., McLean, VA, Rep. PRC-R-2028, Apr. 1980.

[8] D. Parnas, "On the Design and Development of Program Families," *IEEE Trans. on Software Eng.*, Vol. SE-2, pp. 1-9, Mar. 1976.

[9] J. Rothenberg, "SIGMA Message Service: Reference Manual," Version 2.3, Rep. ISI/TM-78-11.2, USC/Inform. Sci. Inst., Marina del Rey, Calif., June 1979. ADA072840

END

FILMED

6-86

DTIC