GCOS-IV SOFTWARE DESIGN SPECIFICATION

File System
Cataloging Services

Issued by: _____
E.Roddy
L.V.Allen, Manager File System Software

Issue Date: August 9,1978
Revision Date: February 1,1979

CONTENTS

CONTENTS (cont)

SECTION 1

FMS SERVICES

1.0 Introduction

1.1 Purpose_and_Scope

This document describes and defines the functional capabilities and external interfaces of the File Management Supervisor (FMS) Services to be implemented for SR5v.

The services provided by FMS apply only to files to be cataloged by the FMS. In addition to such files, there are uncataloged files or files that may be cataloged by means other than FMS. None of the services described herein apply to such uncataloged files.

The services that FMS provides depend on supporting services performed by other operating system subsystems such as Resource Manager. Although occasionally referenced in passing, the operation of these other programs is not described herein.

The version of the EDS does not include the additional functions and options necessary to support:

- Mass Data File subsystem

1.2 Background_Assumed_for_the_Reader

The reader should be familiar with the concepts of the GIII File Management System, GCOS IV ECL, and Resource Manager.

2.0 External_Functional_Description

2.1 Definition_of_Terms

Access_Control_List

An Access Control List is a logical structure used within GCOS 6 to limit access to certain resources including files, directories, peripherals, and application software. The limiting of access includes both those who can use the resource and how the resource may be used.

Access_Control_List,_Initial

The initial access control list is a list associated with a directory that specifies what the access control list of a newly created directory or file subordinate to it will be.

Access_Rights

ACCESS RIGHTS are the privileges or permissions given to a user or users which allow them to use a service or access a resource. An example of a resource from a Media Management System (MMS) point of view is a tape or a disk or a set thereof.

Account

An ACCOUNT is mechanism provided to charge users for services and resources. Typically, an account is assigned to a project or department of which the user is part. An ACCOUNT is also the top directory node for a substructure on a disk volume set. An account node may reside on more than one disk volume set. It differs slightly from a regular directory in that there are special records (or files) associated with an ACCOUNT that contains information such as valid users for this ACCOUNT, accounting information, etc.

One ACCOUNT is always designated as the SYSTEM ACCOUNT and catalogs system information such as mail boxes, work stations, and users who can log on to the system.

Account_Administrator

An ACCOUNT ADMINISTRATOR is the person given the responsibility and authority to allow other persons to

operate under the account and to set some maximum dollar value which the user may charge to the account. The ACCOUNT ADMINISTRATOR represents the OWNER of the ACCOUNT.

Account_Id

An Account_Id is a unique name assigned to each account in the system.

Activity

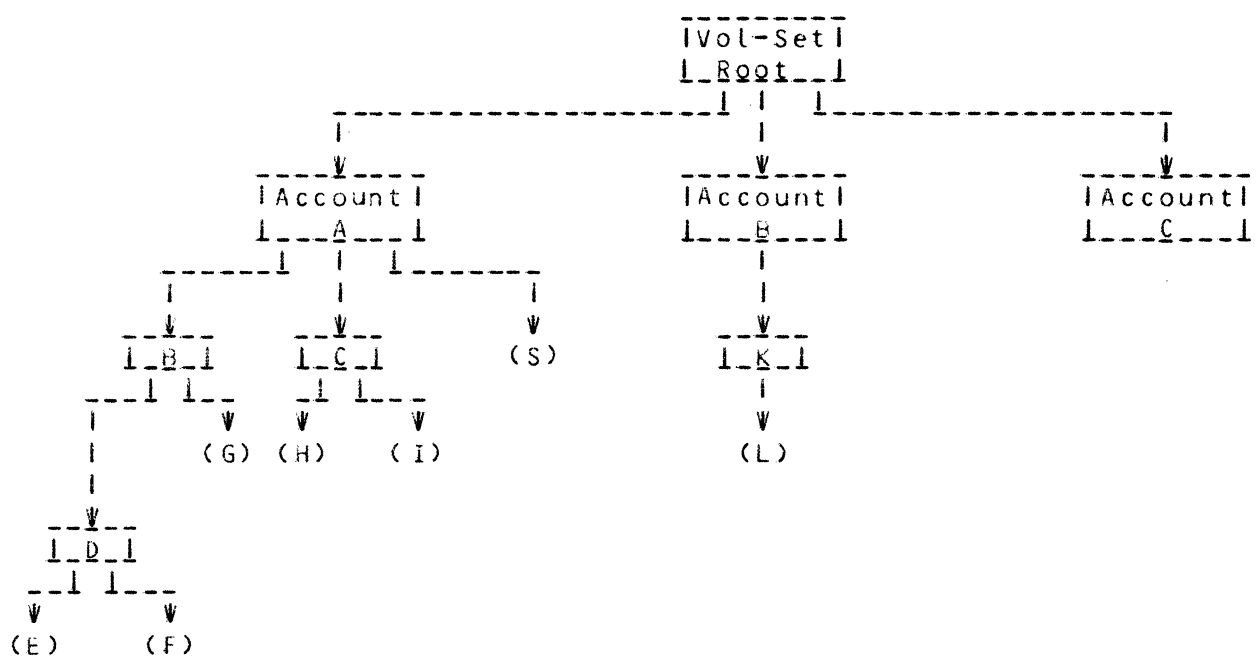
An activity is a subdivision of a job that corresponds to the execution of one or more processes. It is synonymous with the term "job step".

Attribute_Set

The attribute set consists of the information about a file that may be needed when that file is accessed. This information is specified when the file is created. Some of these attributes can be permanently modified or temporarily overridden for the duration of a job.

Catalog

The FMS Catalog is a structure of mass storage records built and maintained by FMS to retain information about files and their authorized users. A separate substructure is created for each account to record information about files cataloged under that account name. The records, directories and file descriptions within each account's substructure are organized to allow hierarchical grouping of files for the account. This organization may be illustrated as follows:



where rectangles represent directories and circles represent descriptions.

Creator

The CREATOR of an entity is the person who first establishes the entity. The right to create an entity subordinate to a catalog node is a right that must be granted by the OWNER of the node. The CREATOR of an entity has several implied rights with respect to that entity; he may read, write, or delete the entity and he may assign the rights of other users (including the OWNER of the superior nodes) with respect to the entity. The CREATOR of an entity is an OWNER of that entity and all subordinate entities.

Directory

A DIRECTORY is a entry in the structure that serves as an index to subordinate directories, file descriptions, and links to other parts of the structure. It is used to establish the hierarchical or "tree" organization of the structure. It also contains access control information for the directory node and default access control and attributes for subordinate files and directories.

Directory_Initial_Working

The directory that is established as the working directory for a user when he first logs into the system is the initial working directory. This directory is specified in the user profile and is based on the userid and account under which he enters the system.

Directory_Root

The root directory is that directory that is superior to all other directories or files on the containing volume set. The root directory is often referred to simply as the "root".

Directory_Working

The working directory identifies the user's current location within the catalog hierarchy with regard to relative pathnames. Any pathname specified by the user that does not begin with a greater than(>) character is considered to be relative to the working directory. By default, this directory is used by the search rules (See section 3.3.4).

Directory_Hierarchy

The directory hierarchy is the tree structure organization of the logical contents of the catalog.

Entry_Name

An entry name is a name given to an item contained in the catalog data structure. It may contain one or more components separated by periods. All names given to entries immediately subordinate to a directory must be unique.

Equal_Convention

The equal convention is a method used by some commands to specify one or more characters in a group of subordinate entry names.

File

A FILE is a collection of information consisting of records pertaining to a single subject. In the context of business data, a payroll file is an example. A file may reside on a disk or tape volume set and may be recorded on more or part of a volume or on more than one volume. It may be cataloged or uncataloged. If it resides on disk and is cataloged, it must reside on the same volume set as it's file description.

File_Generation

A File Generation, or simply a Generation, is a file which is part of a Generation Set.

File_Generation_Set

A Generation Set is a cataloged set of successive, functionally related files which are controlled by the system.

File_Name

Filename is a generic term for the identifier of a file. More specific terms are absolute pathname, relative pathname, entryname, external file name and internal file name. Each of these is defined in this section.

File_Name,_External_(efn)

An external file name is fully qualified pathname which identifies a file unambiguously to the file system. An external filename is often referred to as a pathname or a filename.

File_Name,_Internal_(ifn)

The internal file name relates the program's declaration of a file to an external file name. The file so referenced must be present in the search path. The internal file name corresponds to the "file code" of GCOS III.

File_Set

A collection of one or more related files recorded consecutively on a tape volume set. A file set may be cataloged or uncataloged.

Link

A Link is an entry in the catalog structure that specifies the pathname of an entry in another directory. It allows references to items in other directories as if the were actually contained in the working directory.

Logical_Unit_Designator_(LUD)

The Logical Unit Designator (LUD) allows a single file to be referenced by different ifn's within subsequent activities without establishing an equivalence between each new ifn and the efn. The LUD serves to equate the ifn in the subsequent activity with the file assigned to the same LUD in a previous activity. The allocation for the file must be continued across each activity in question. Thus, when an efn changes, only a single command change is required; each activity does not have to be updated.

Owner

The OWNER of a node is the person to whom the charges for that node will be applied; e.g., the ACCOUNT ADMINISTRATOR is the OWNER of the account. The OWNER of a node has the implicit right to delete that node or any subordinate node, however all other rights to the node or any subordinate node must be explicitly granted. An example of a node is a directory, volume set, etc..

Parameter

A parameter is controlling information provided by the user to the command processor. Parameters customarily appear in argument lists as self-identifying, positional, or "keyword value" elements. Parameter may appear as single elements or as lists of elements.

Pathname

A pathname is a character string that specifies a directory, file, or link by its position in the directory hierarchy. The pathname may be either relative or absolute.

Pathname,_Absolute

An absolute pathname is a concatenation of file name with all superordinate directories leading back to the volume set root.

Pathname,_Relative

A relative pathname is a pathname that uniquely identifies a cataloged object (directory, file, or link) relative to the working directory.

Person_Id

A Person_Id is a unique name assigned to each individual authorized to use resources of the system. The individual may act in several roles and operate under one or more accounts. Associated with each Person_Id is a password, used during Log_In, to insure the identity of the individual.

Star_Convention

The star convention is a method used by some commands to specify a common component of several entry names by the use of the asterisk character.

System_Administrator

The SYSTEM ADMINISTRATOR is the person concerned with the overall integrity, security, and maintenance of the computer hardware, software, and database throughout

the enterprise. His organization includes database, communications, and storage administrators as well as a staff of system programmers. The SYSTEM ADMINISTRATOR represents the OWNER of the system resources.

User

A USER is an entity which utilizes the resources of the system. He has a defined set of access rights to system services, files, work stations and mailboxes.

User_Session

A User Session is defined for a user to be the ordered set of commands and resources invoked from the "log on" action initiated by the user to the termination of processing on behalf of the user. A session spans the time from the "Log on" to the "Bye", regardless of input media or mode of system connection.

Volume

A VOLUME, from the MMS point of view, is either a disk or tape volume. Any given volume known to MMS belongs to one and only one volume set or it is unassigned (must be cleaned, etc.).

Volume_Set

Volumes are managed in groups of related volumes called VOLUME SETS. One or more disk or tape volumes (of the same type) may constitute a VOLUME SET. A VOLUME SET may either be in scratch, hold or assigned status.

2.2 FMS_Services

The file management supervisor is the administrator of a data structure used to maintain information about each file's space, attributes of the file and authorized users of that space. Information is kept about accounts, users, files and directories and their implicit and explicit relationships.

The information is built and maintained by a set of services that are invoked by ECL statements and/or direct calls from programs. These services provide the capability of

creating, deleting, allocating, retrieving, modifying and listing entries in the data structure.

2.2.1 Cataloging

Cataloging is the method by which information concerning the file is built and maintained and is the basis, thus, for all other services. Cataloging services allow files to be created and deleted and their descriptions to be grouped and modified.

2.2.1.1 Catalog_Creation_and_Maintenance

The following operations to build and maintain directories, links, and files must be supported:

- o Creation - the construction of a directory, link or file description, and particularly in the case of file creation, means the assignment of space for the file.

- o Modification - the changing of a catalog link or file description, but no changing of file content.

- o Deletion - the removal of a catalog and any subordinate catalogs, links, or file descriptions and usually the release of space assigned to files. Deletions are achieved in two forms: in one, the file space is overwritten with zeroes before release and in the other, this relatively time-consuming operation is skipped.

- o Listing - the display of directories, links, and file descriptions but no display of file content.

- o Inquiries - provide program access to information in directories, links, or file descriptions, but not to file content.

2.2.1.2 File_Allocation_and_Deallocation

- o File Allocation - Validate the requester's right to access a cataloged file. If access is authorized, grant read or read and write access to the file content and build the mapping information that this requires. This function also establishes an association of the allocated file and an internal name used by any programmatic reference to the file content.

- o File Deallocation - Remove the association between the file and the internal file name and delete the mapping information used to access the file content.

2.2.1.3 Miscellaneous Services

- o Add file space
- o Release file space
- o Read file attributes
- o Write file attributes

2.3 Catalog Structure

The catalog is a structure of mass storage records used to maintain information about files and their authorized users. The catalog is based on a tree structured directory hierarchy which contains directory, file and link entries.

2.3.1 Directory

A directory is an entry in the tree structure which serves as an index to files links and other directories. The directory also contains access control information for itself and default access control lists for subordinate directories and files.

The directory is located by specifying a name concatenated to the name of the superior directory. That directory is located in the same manner by the name of its superior directory. The first directory, considered as the base of the tree is called the root directory. All other directories are subordinate to it.

2.3.2 File

Each cataloged file is described by a file description entry in the catalog structure which contains a full set of attributes describing properties of the named file. This information includes the file's physical location, size protection required, date last accessed, access control information, etc. It may be located by specifying the file's name concatenated to the name of its superior directory.

2.3.3 Link

Each link contains the pathname of a target entry. It may reference directly to the entry of interest or name another link. The entry being referenced may be a directory, file

subordinate to its own directory, in another directory, on another volume set or a file at a remote site.

2.4 Catalog_Structure_Placement

Although one can visualize the catalog as a tree with multiple directory levels all emanating from the root, there may be in fact multiple trees on-line at the same time. Where each tree is uniquely identified by its root directory.

Each tree is laid out on one disk volume set and the name of the host volume set is also the name of the tree's root directory. Therefore, the pathname indicates the disk volume set on which the directives, links and file descriptions are to be placed.

2.5 Referencing_Cataloged_Directories,_Files,_and_Links

Reference to any entry begins at the first entry (or root directory) in the tree and consists of a string of directory names ending with the name of the target file, link or directory.

2.5.1 Entry_Name

An entry name is the name of a file, directory or link. An entry name consists of at least one and no more than 32 characters. This name may consist of upper or lower case alphabetic characters (A-Z, a-z), digits (0-9), hyphens (-), periods (.), and underscores (_). However, the first character of the name must not be a hyphen or an underscore.

The period is used to separate components of an entry name, where each component is a logical part of the name. In some cases, system software such as the PL/1 and COBOL compilers will consider the last component of the file name as a qualifier that denotes the file's content. Several system conventions (e.g., the star convention and equal convention) may be used with the FMS services to operate on component parts of the name.

If the entry name represents a generation set, a generation identifier may be append to the name. This generation identifier is used to indicate a specific member of the generation set when creating, deleting or allocating a generation file. The syntax provides for both absolute and relative specification of the desired generation and is indicated by:

*Gnnnn	absolute generation
*0 (or LAST)	last generation created
*+1 (or NEW)	new generation
*-nn	generation relative to the last generation.

If no specific generation is specified the last generation is assumed for read allocations and a new generation is assumed for write allocations.

2.5.2 Pathname

A pathname is a sequence of entry names used to reference a file or directory. Each entry name except the last in a pathname is the name of a directory entry in the directory hierarchy. The last entry name in a pathname is the name of a file, directory or link entry. Names which are concatenated to form the pathname are separated by greater-than characters (>). The total pathname must not exceed 168 characters in length. The number of levels that may be specified in a pathname will depend on the accumulative size of the entry names, however it may not exceed 15.

There are two basic types of pathnames - absolute and relative. An absolute pathname traces an entry from the root directory; a relative pathname traces an entry from the current working directory. A special case of a relative pathname is a "simple" pathname.

2.5.2.1 Absolute Pathname

>name1	Start at the system root and locate name1.
>name1>name2	Start at the system root and locate name1. In directory name1, locate name2.

If the first element of a pathname begins with a circumflex (^) character position 5/14 in 150 R646 IRV), the element is a volume set name and specifies that the root of the directory hierarchy is located on the identified volume. For example:

^AB1234>name1	Go to the root on the volume set "AB1234", and locate name1
^AB789>name1>name2	Go to the root on the volume set "AB789", and locate name1. Then locate name2.

A null volume set name in an absolute pathname of this form references the Working Volume Set. For example:

^>name1	- Go to the root of the Working Volume Set and locate name1.
^>name1>name2	- Go to the root of the Working Volume Set and locate directory name1. In directory name1, locate name2.

2.5.2.2 Relative_Pathname

A relative pathname appears much like an absolute pathname except that it may not contain a leading greater-than (>) or circumflex (^) character. It is interpreted as a pathname which is qualified by using the current working directory as a prefix. The simplest form of a relative pathname, known as a simple pathname, is the single name of an entry. For example:

name1	Locate name1 in the current working directory.

A more complex form of the relative pathname uses a directory located by adding the current working directory as the starting point to search through one or more levels for the desired file or directory. See the following example.

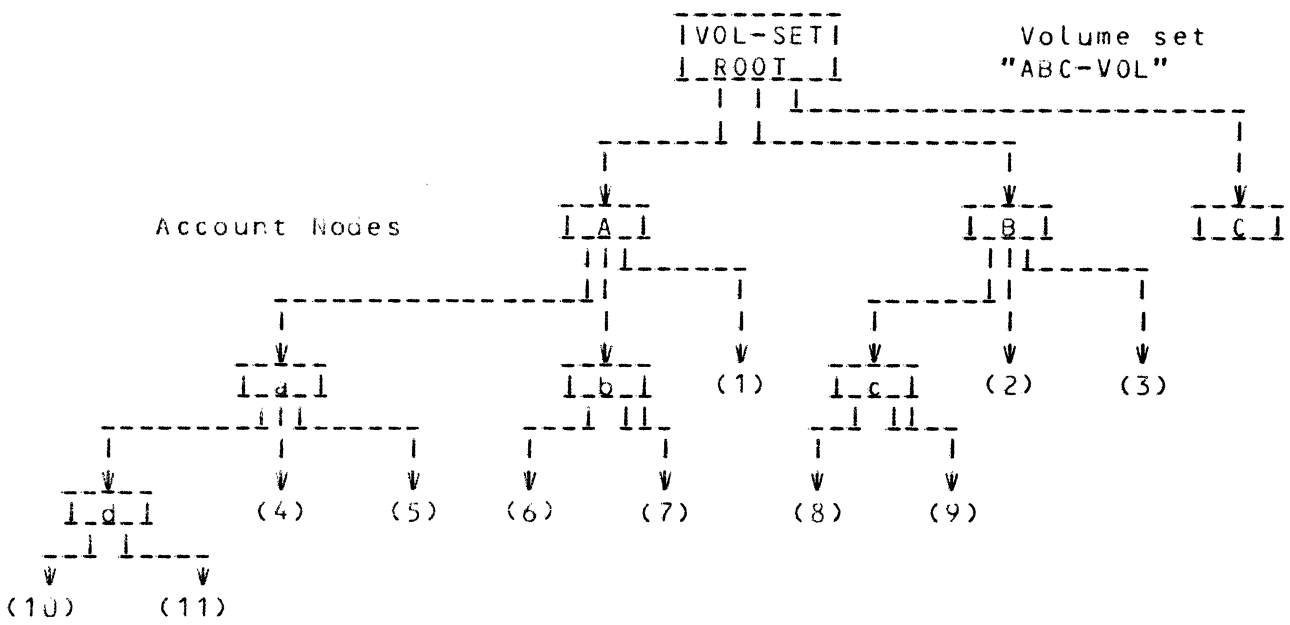
name1>name2	Locate name1 in the current working directory and then locate name2. Note that the search rules are not applied since it is a multiple entry pathname.

A less-than (<) character position 3/12 in ISO R646 IRV) can be used at the beginning of a relative pathname to indicate that the directory immediately superior to the current working directory is where the entry name may be found. The less-than character can be used to denote levels in the directory hierarchy similar to the use of the greater-than character. Each less-than character represents one level in the hierarchy (toward the root), starting at the current working directory. In this way, a directory several levels superior to the working directory can be searched.

<<<name1	Go back 3 directories and then locate name1.
<<name1>name2	Go back 2 directories and locate name1. Then find name2.

2.5.2.3 Examples

Assume a volume set with the following structure:



Where:

1. Volume set "ABC-VOL" contains Accounts "A","B", and "C".

2. Account "A" has two immediately subordinate directories, "a" and "b", and one immediately subordinate file, "1".

3. Account "B" has one immediately subordinate directory, "c", and two immediately subordinate files, "2" and "3".

4. Account "C" is undefined for the purpose of this example.

5. Subordinate directory "a" of Account "A" has one immediately subordinate directory, "d", and two immediately subordinate files, "4" and "5".

6. Subordinate directory "d" of directory "a" has two immediately subordinate files, "10" and "11".

7. Subordinate directory "b" of Account "A" has two immediately subordinate files, "6" and "7".

8. Subordinate directory "c" of Account "B" has two immediately subordinate files, "8" and "9".

Then:

The absolute pathname of file "11" is - ^ABC-VOL>A>a>d>11

The absolute pathname of file "4" is - ^ABC-VOL>A>a>4

The absolute pathname of file "6" is - ^ABC-VOL>A>b>6

The absolute pathname of file "1" is - ^ABC-VOL>A>1

The absolute pathname of file "8" is - ^ABC-VOL>B>c>8.

The absolute pathname of file "2" is - ^ABC-VOL>B>2.

If the Working Directory is at Account "A" (^ABC-VOL>A), then:

The relative pathname of file "11" is - a>d>11,

The relative pathname of file "4" is - a>4,

The relative pathname of file "6" is - b>6,

The relative pathname of file "1" is - 1,

The relative pathname of file "8" is - <B>c>8, and

The relative pathname of file "2" is - <B>2.

If the Working Directory is at subordinate directory "a" of Account "A" (^ABC-VOL>A>a), then:

The relative pathname of file "11" is - d>11,

The relative pathname of file "4" is - 4,

The relative pathname of file "6" is - <b>6,

The relative pathname of file "1" is - <1,

The relative pathname of file "8" is - <<B>c>8, and

The relative pathname of file "2" is - <<B>2.

2.5.2.4 Star Names

Many commands that accept pathnames as input allow the final entryname in the pathname to be a star name. A star name is

an entryname. The star convention matches a star name with entrynames in a single directory to identify a group of entries. The entrynames matching a star name have components in common and are matched according to specific rules. Commands that accept star name perform their function on each entry identified by the star name.

o Constructing Star Names

Star names are constructed according to the following rules:

1. A star name is an entryname. Therefore, it is composed of a string of 32 or fewer characters as described in section 2.5.1.

2. A star name is composed of one or more nonnull components. This means that a star name cannot begin or end with a period (.) and cannot contain two or more consecutive periods.

3. Each question mark (?) character appearing in a star name component is treated as a special character.

4. Each asterisk or star character appearing in a star name component is treated as a special character.

5. A star name component consisting of only a double star (**) is treated as a special component.

o Interpreting Star Names

A star name is matched to entrynames in a single directory according to the following rules:

1. Each question mark (?) in a star name component matches any one character that appears in the corresponding component and character position of an entryname.

2. Each asterisks in a star name component matches any number of characters (including none) that appear in the corresponding component and character position of an entryname. If the asterisk is the only character of the star name component, it matches any corresponding component of an entryname. Only one asterisk can appear in each star name component, except for the double star component as noted in the next rule.

3. The double star component (**) in a star name matches any number of components (including none) in the corresponding position of an entryname. Only one double star component can appear in a star name.

2.5.2.5 Equal_Names

Some commands that accept pairs of pathnames (e.g., the Rename command) allow the final entryname of the first pathname to be a star name, and the final entryname of the second pathname to be an equal name. An equal name is an entryname containing special characters that represent one or more characters from the entrynames identified by the star name. This provides a powerful mechanism for mapping certain character strings from the first pathname into the second pathname of the pair.

o Constructing Equal Names

An equal name is constructed according to the following rules:

1. An equal name is an entry name. Therefore, it is composed of a string of 32 or fewer characters as described in section 2.5.1.

2. An equal name is composed of one or more nonnull components. This means that an equal name cannot begin or end with a period (.) and cannot contain two or more consecutive periods.

3. Each percent sign appearing in an equal name component is treated as a special character.

4. Each equal sign (=) in an equal name component is treated as a special character.

5. An equal name component consisting only of a double equal sign (==) is treated as a special component.

o Interpreting Equal Names

An equal name maps characters from the entrynames that match the star name (first entryname) into the second entryname of a pair according to the following rules:

1. Each percent sign (%) in an equal name component represents the single character in the corresponding component and character position of an entryname

identified by the star name. An error occurs if the corresponding character does not exist.

2. An equal sign (=) in an equal name component represents the corresponding component of an entryname identified by the star name. An error occurs if the corresponding component does not exist. An error also exists if an equal sign appears in a component that contains a percent character. Only one equal sign can appear in each equal name component, except for a double equal sign component, as noted in the next rule.

3. The double equal sign (==) component of an equal name represents all components of an entryname identified by the star name that have no other corresponding components in the equal name. Often the double equal sign component represents more than one component of an entryname identified by the star name. If so, the number of components represented by the entire equal name is the same as the number of components in the entryname. When the equal name contains the same number or more components than the entryname, a double equal sign is meaningless and, therefore, ignored. Only one double equal sign component can appear in an equal name.

2.5.3 Remote Files

In GCOS IV, all pathnames are relative to root directories (Volume Sets) located at the host site. Therefore, in order to reference a file located at another site, a link entry must be created in the local site's catalog structure which defines the location of the file and the name by which the file is cataloged at the remote site.

The link entry will contain the absolute pathname of the file and a mailbox that represents the remote nodes file transfer work station.

When the file is to be accessed from the local site, the requestor will provide the patnname of the link and will be given the file transfer mailbox and the absolute path name for the file. The requestor (e.g., Resource Manager) must then use some predefined protocol to actually allocate the file at the remote site.

2.5.4 Related_Files

Within the files cataloged in FMS there are or may be groups of two or more files that are related to each other in such a fashion that they must be maintained at the same update base. The Index file and the Data file of an Indexed Sequential file are a good example of such related files, however, other user data files may have the same sort of relationship. For such files, if one of the files is restored to a prior update state, all of the related files must also be restored to the same update state. In GCOS III, FMS has no "knowledge" of this relationship and the entire burden of maintaining the correct update state of such files falls on the person performing the RESTORE or ROLL BACK. Since the RESTORE is usually performed by a System Administrator who may not have direct knowledge of the relationship of the files and since GCOS IV provides a physical RESTORE by volume for which the user may not directly know just which files are being restored, this burden may well be excessive.

As a solution to this problem, FMS provides an attribute on each directory that is interpreted to mean that all files subordinate to this directory are related. The RESTORE Utility will interrogate this attribute on any implicit file restore (any RESTORE other than one with the -ONLY option) to ensure that all related files are also restored.

It will remain the user's responsibility to create the directory with the "related" attribute and to create all related files subordinate to the directory.

2.6 Naming_Conventions

2.6.1 Entry_Name

An entryname consists of at least 1 and not more than 32 characters. This name may consist of upper or lower case alphabetic characters (A-Z, a-z), digits (0-9), hyphen (-), periods (.), and underscores (_). However, the first character of the name must not be a hyphen or an underscore.

2.6.2 Person_Id

The person_id (person identifier) is made up of at least 1 and no more than 12 characters. The name may consist of upper and lower case alphabetics (A-Z,a-z), numbers (0-9), hyphens (-), and underscores (_). How-

ever, the first character must not be a hyphen or an underscore.

2.6.3 Account_Id

The account name consists of at least 1 and no more than 12 characters. The name can be made up of upper and lower case alphabetics (A-Z,a-z), numbers (0-9), hyphens (-), and underscores (_). The first character must not be a hyphen or an underscore.

2.6.4 Volume_Identifier

The volume identifier is at least 1, but no more than 6 characters. the name may consist of upper and lower case alphabetics (A-Z,a-z), and numbers (0-9).

2.6.5 Volume_Set_Name

The volume set name must consist of at least 1 and not more than 12 characters. The name may be made up of upper and lower case alphabetics (A-Z,a-z), and numbers (0-9).

2.6.6 Internal_File_Name

The internal file name may consist of from 1 to 8 alphabetic (A-Z,a-z) or numeric (0-9) characters.

2.7 Cataloged_File_Placement

2.7.1 Mass_Storage_File

Cataloged disk files will always reside on the same Volume Set as its defining catalog structure. However, FMS does provide options at file creation to allow the user to place a file on specific volumes within the Volume Set. An option is also provided to request that the file be spread across volumes within the Volume Set.

If neither the specific volume option nor the spread option is specified, FMS will seek space on the volume within the Volume Set that contains the most available space.

If a volume constraint is specified for a file and enough space is not available on the specified volume, the file create request is denied. When FMS selects the volume, denial occurs only when insufficient space is available on any volume in the volume set.

If a file is grown, the request may specify the volume(s) on which space for growth is obtained. If a specific volume was named, growth is constrained to that volume. When a specific volume is not specified an attempt is made to grow the file on the volume that contain the last extent of the file. If space is unavailable there and a specific volume was not named, space is sought on the volume within the set with the most available space. Finally, if unavailable there, the request is desired.

2.7.2 Tape_Files

Cataloged tape files are always assigned to a specified Tape Volume Set. Options are available to request that the file be assigned to a predefined volume set or on a new volume set that will be created from available scratch tapes.

The description of the cataloged tape file is always placed on the Disk Volume Set indicated by the pathname.

2.8 Mass_Storage_Space_Control

There are three places where the amount of mass storage space to be used can be specified. These are:

1. Account Record - the maximum amount of space on the Volume Set that can be used both for catalogs and files cataloged under the account.

2. Directory - the maximum amount of space on the disk Volume Set that can be used both for catalogs and files cataloged under the directory.

3. File description - the maximum amount of space to be assigned to the disk file (but not for a file on magnetic tape).

Each maximum can be specified to be unlimited and each can be changed. When a maximum specification is changed, it cannot be changed to be less than the amount currently assigned.

The account maximum is checked on each file create or modify and each file grow. The file maximum is checked, of course,

only on each growth of that file. Where the maximum would be exceeded, the request for file create, modify, or file growth is denied.

File or directory delete releases space, thereby decreasing the amount currently assigned and increasing the amount that may be newly assigned.

The initial amount of space to be assigned a file (except for a file on magnetic tape may also be specified. When an initial amount is not specified, the maximum cannot be specified and it is assumed to be one allocation unit. If an initial amount but no maximum is specified, the initial and maximum amounts are assumed to be the same.

2.9 File_Generation_Sets

The File Generation capability allows a user to establish a cataloged set of successive, functionally related files known as a Generation Set. The files, called generation files or simply generations, are ordered as they are created so that the latest generation or previous ones may be easily retrieved. This also allows obsolete generations to be automatically deleted.

A generation set may be used to automatically retain previous versions of a file when it is updated or to collect sets of related data for subsequent retrieval and processing. The generation set addressing technique and automatic file creation allows a data collecting or updating job to be executed any number of times without requiring any change to the job control statements.

The generation set may contain files that reside on removable disks, non removable disks, or magnetic tapes. Each file will be cataloged and maintained by FMS, therefore all services provided by FMS, including prevention of unauthorized access, mass store space control, protection against device or incomplete update, etc., are available for generation files.

2.9.1 Creating_a_Generation_Set

A generation set must be established by the user by issuing a CREATE_GEN_SET command. The user must specify the name of the set and the generation control mode. A default file definition and password are optional.

2.9.1.1 Generation_Control_Modes

The user must specify one of three generation control modes for a generation set.

- o Cycle

  This type allows the user to specify the number of generations that are to be maintained in a fixed cycle. The generations can be explicitly defined using a File Create function or the generations will be dynamically created for each new generation until the set contains the number of generations specified. Once each generation in the cycle has been created, for each new output generation, the oldest generation will be overwritten to become the newest generation.

- o Generations

  In this mode, the user indicates the number of generations to be retained. The generations will not be recycled, but instead new output generations will be created based on the default file definition declared for the set. Once the number of generations exceeds the number of generations to be retained, the oldest generation will be deleted.

- o Retention Date

  In this mode, generations are retained until their retention has expired. The retention date may be expressed as a calendar date (mmddyy) or as the number of days the generation is to be retained. Each new output generation will be created based on the default file definition specified for the set. Each file will be retained until its retention period has expired and then it will be deleted.

2.9.1.2 Default_File_Definition

The user may specify the options to be used for each default file create. Any option allowed for with a File Create may be specified. The options are described with the File Create command in section 3.1.1.4.

If the options are not specified, the system default options will be used. The system default initially defines a mass storage file that consists of one 512 byte control interval, but the default may be modified by the System Administrator.

2.9.2 Creating_a_Generation_File

A generation file may be explicitly or dynamically created for a generation set. If the generation control specified for the set is number of generations or retention period, a new generation will be created each time a new generation is required. In this case, the default file definition specified for the generation set will determine the attributes of the new generation, e.g. size, media, etc.

If the cycle control mode is specified, the user may choose to predefine each generation using the File Create function. This option allows the user to assign specific tape volume sets to each generation or to spread the disk generations across volumes of the volume set. However, if any of the generations of to set are not defined by the time it is required for output, a generation will be created using the default file definition.

When a File Create is used to explicitly create a generation, the name of generation set must be used as the pathname. Otherwise the create command is the same as that used for any non generation file.

2.9.3 Addressing_Generation_Files

A generation file may be addressed as a relative or absolute generation.

2.9.3.1 Relative_Generations

The generations in the set are maintained in an order that represents the relative age of the files within the set. This allows files to be referenced using a relative generation number.

When a new generation is created it becomes relative generation 0. When the next generation is created, it becomes generation 0 and the previous generation becomes -1. Similarly when the next generation created it becomes relative generation 0 and the previous generations are shifted by 1.

In order to address a generation file using a relative file number, the number preceded by an asterisk (*) must be appended to the generation set name. For example:

GEN_SET_A*0 indicates the last generation

GEN_SET_A*-1 indicates the previous generation

GEN_SET_A*+1 indicates a new generation

In addition, two alternate forms are allowed. The key word *LAST may be substituted for *0 and *New for *+1.

When relative generation numbers are used within a multiple activity job, all referencing must be relative to the status of the generation set at the beginning of the job.

2.9.3.2 Absolute_Generations

A unique identifier, called an absolute generation number, is also assigned as each file generation is created. Once assigned to a generation, this number never changes. This allows a specific file generation to be accessed regardless of its relative position in the set.

The first generation created following the creation of the generation set will be assigned absolute generation number 1. Each subsequent generation for the set will be assigned the next higher absolute generation number. This number will be incremented from 1 to 9999 and then will start over at 1.

To address a generation using the absolute generation identifier, the number, preceded by an asterisk and an upper case G must be appended to the generation set name. For example:

GEN_SET_A*G0015 specifies the 15th generation

2.9.4 Allocating_Generation_Files

A generation file may be allocated to a job using a GET job control statement or dynamically during execution of a program by issuing a "GEMORE" call.

- o Input

  For input, the generation may be addressed using either a relative or absolute generation name. The generation set's password and access control list will be checked to ensure that the requestor has the proper permission to read the specified generation. If permission is granted, the generation will be allocated for read if it is not currently allocated to any other job.

o Input Concatenation

Concatenation will allow two or more generations or all generations of a set to be processed by a program as one contiguous file. The order of concatenation may be indicated by specifying a series of relative generation numbers separated by commas. The key word ALL may be used to indicate that all generations in the set are to be concatenated. For example:

GEN_SET_A*0,-1,-3

will cause the last generation to be read, followed by relative generation -1, and then generation -3.

GEN_SET_A*ALL

will cause each generation of the set to be read, starting from the oldest generation to the most recent generation.

o Output

For output, the relative name is always used when creating a new generation. However, if the intent is to rewrite an existing generation then its absolute generation name must be used.

The generation set's password and access control list will be checked to determine whether the requestor has permission to write to the file. If so, the allocation of an output generation will proceed based on the generation control mode for the set.

o Cycle

If the number of generations specified have not been assigned to the set, a new generation will be created based on the default file definition. This new generation will be registered in the generation set catalog record and the file will be assigned to the job. Once all generations have been created, the oldest generation will be allocated as the new output file.

o Generation or Retention Date

A new generation will be created based on the default file definition for the set and it will be assigned to the job.

2.9.5 Deallocating Generation Files

The action to be taken during deallocation depends on the disposition code specified when the generation was allocated.

2.9.6 Catalog Maintenance

Generation files may be maintained like any other cataloged file. They may be listed, deleted, and their attributes modified using the standard FMS commands. The only difference is that absolute generation numbers must be used to address the generations.

2.9.7 Generation File Statistics

Information about the generation set and generation file, e.g. absolute and relative generation number, volume identification, media type, creation date, last allocation date, etc., will be maintained for each generation of the set. This information will be available to the user as part of the information returned by the LIST function.

To provide an audit trail, an accounting record will be written to the Statistical Collection File each time a new generation is created or deleted.

2.10 File Attributes and Overriding Rules

The file management system and file content managers will maintain an attribute block for each cataloged file in the system. The purpose of this attribute block is to provide information that would otherwise need to be specified on job control statements each time the file was accessed. This information need only be provided once when the file was written. The information will be maintained as long as the file exists. Some attributes may subsequently be permanently modified or temporarily overridden for the duration of a job.

2.10.1 Types of Attributes

There are three types of data commonly referred to as attributes:

- o File Content Attributes

  File Attributes are those characteristics which specify the physical and logical properties of a file, such as control interval size or recording mode. These attributes do not vary by user, and are fixed for a given file. Since they are needed for any access to the file, they are saved in the catalog for the file when the file is built. These saved attributes are used to ensure that subsequent requests to access the file are legitimate.

- o File Access Attributes

  File Access Attributes are properties of the accessing program, such as Access Level. For instance, a program may be written to access control intervals, or it may be written to access records, but it may not be written to access both control intervals and records at the same time. This type of attribute must be specified at Open time; it is not saved and cannot be overridden.

- o File Use Attributes

  File Use Attributes are properties relating to the use of a file, such as the number of buffers to be used in reading or in writing the file. This type of attribute is independent of both the file and the program. For example, a program may use 10 buffers for a file during one execution and 100 buffers during the next execution, without changing the program or the file. These attributes are not saved, but can be overridden in some cases, as explained later.

2.10.2 Specification of Attributes

There are three methods or times to specify attributes. The resolution of conflicting specifications is done according to the Overriding Rules, as explained later. Not all attributes can be specified in all three ways. The three ways to specify attributes are:

- o Compile (or Open)

  Compile (or Open) attribute specifications are passed to UFAS in the File and Access Definition (FAD). These attributes are placed in the FAD by the compiler or by the corresponding run-time subroutines. In either case, they

are presented to UFAS by the caller when the file is opened.

o Job Control Statements

Attribute specifications are supplied by the user in the command language for the job. These specifications are obtained from GCOS IV by the UFAS Open procedure, and are merged with other specifications. These specifications are important for programs written in languages such as FORTRAN, which have little ability to specify attributes in the source program.

o File Create

File Create attribute specifications are supplied by the user at the time the file is created. They are saved in the catalog for the file until the UFAS Open procedure obtains them.

2.10.3 Overriding Rules

Overriding rules are used by the UFAS Open procedure to merge the attribute specifications from the three sources. These rules can be broken into three groups, corresponding to the three types of attributes:

o File Access Attributes

File Access attributes are really attributes of the accessing program. Therefore, they can only be specified in the File and Access Definition (FAD) at compile (or Open) time. There is no provision to specify these attributes with ECL, or in File Create. These attributes are not saved in the catalog as are other attributes. Since there is only one place to specify these attributes, no conflict is possible. Therefore, the Overriding Rule is:

-File Access attributes can only be specified in the FAD. They cannot be overridden by ECL specifications or by File Create specifications.

o File Use Attributes

File Use attributes specify the way a file is used for a particular session. Since they are independent of both the program and the file, these attributes can be specified in ECL. A default can be specified in the FAD, for a specific file, but the default is always overridden by a corresponding ECL specification, if present. These attributes are not saved in the catalog and cannot be

specified in File Create. Therefore, the Overriding Rule is:

- -A File Use attribute in the FAD (or the default, if not specified) is overridden by an ECL specification. File use attributes cannot be specified in File Create, and are not saved beyond the allocation of the file.

o File Content Attributes

File attributes belong to the file itself, and, thus, are saved in the catalog for the file during the life of the file. Some of these file attributes can be specified in the File Create primitive. The rules can be separated for existing files and for new files being built.

o New Files

Many more attributes can be overridden for new files than for existing files. Some attributes, such as key-offsets, cannot be overridden, since the accessing program depends on them. The Overriding Rule is:

- -In output processing mode, file attributes specified in the FAD are overridden by ECL specifications, if present. Either of these specifications is overridden by File Create specifications, if present.

o Existing Files

Since file attributes describe actual physical properties of a file, none can be overridden once a file is built. Some FAD or ECL specifications, if present, must agree with existing attributes. For example, if an accessing program specifies binary recording mode, and the allocated file is in ASCII recording mode, an error exists. It is apparent that any accessing program must interpret records differently for each of these two cases. Other attribute specifications in the FAD, such as control interval size, may be safely ignored, and the attribute saved in the catalog is used. This provides a level of data independence, in that a given program can access files with differing control interval size without recompilation. The overriding Rule is:

- -In Input, I/O, or Append processing mode, file attributes saved in the catalog for the file are used, and cannot be overridden. Any file attributes specified in ECL must agree with these existing attributes, or an error exists. Some file attributes specified in the FAD must agree with existing attributes, but some are ignored.

2.10.4 Modification_of_Attributes

Since File Access and File Use attributes are not saved, they need not be modified. Therefore, of concern are those File content attributes which may, or may not be modified.

Since these attributes describe actual logical and physical properties of the file, they cannot be modified without corresponding changes to the file itself. For instance, if the control interval size specification is to be changed, each control interval in the actual file must be changed to the new size. Therefore, these properties should be changed by the File Restructuring Utility. This program will make the appropriate changes to the file itself, and then will modify the existing File attribute specifications so that they agree with each other. There is no need for a user-visible File Modify function to change these attributes.

2.11 File_Security

Unauthorized use of a cataloged file may be prevented by means of an access control list, security class, and passwords. Audit records may also be specified to show file usage.

2.11.1 Access_Control

Access control allows individual users to grant or deny other users access to their files and directories at their own discretion. Access control lists (ACLs), which regulate this type of access, allow individual users to allow access on a per-user basis. The users can exercise control only within those portions of the catalog hierarchy where they themselves have the proper access rights.

2.11.1.1 Access_Identifier

In order to grant the user distinct access right it is necessary to be able to identify the different users. For this purpose, each process (tenant) has a access identifier that is fixed for the life of the process. This identifier consists of four components that are used to delimit use of the resource as follows:

- Person id - individual that may access the resource.
- Account id - account under which the resource may be accessed

- Node id - network node from which the resource may be accessed.

- Domain id - execution domain that may access the resource.

2.11.1.2 Access_Rights

The access rights are the modes in which files, directories, and link may be processed. These rights allow a user to stipulate several different kinds of access on the same entry in the catalog structure if he desires.

The access rights for directories are:

Modify (M) — the attributes of file, directories and links contained in the directory and certain attributes of the directory itself can be modified. This mode also allows existing files, directories or links contained in the directory to be deleted or listed.

Modify_Entity (X) — the directory's attributes, including the access control list, can be modified, listed or deleted.

List (L) — the directories attributes may be read by the process.

Create (C) — new files, directories and links can be built immediately subordinate to the directory.

Delete (D) — the directory and all subordinate directories, files and links may be deleted.
This permission has been included for GCOS III accommodation.

Null (N) — the process may not access the directory in anyway.

The access rights for files and file generation sets are:

Read (R) — allows data to be read from the file.

Write (W) — allows data to be written to the file. This can be used to change, delete, add, insert records in the file.

Append (A) allows data to be written to the end of a sequential/relative file.

Execute (E) allows transfer of information from the program but only for a compiler or loader.

Null (N) the process cannot access the file (description or content) in any way.

Modify_Entity (X) the file attributes, including the file's access control list can be modified, listed or deleted.

Delete (D) the file description and file space can be deleted.
This permission has been included for GCOS III accommodation.

Recover (F) allows data to be written to a file when it is abort locked or has defective space. It also allows the abort lock to be set or reset.

The access rights for links are:

Modify_Entity (X) the link's attributes can be modified and listed or the link may be deleted.

Delete (D) the link can be deleted.

Null (N) the process cannot list, modify or release the link.

2.11.1.3 Structure_of_an_ACL

The rights that different user processes have when referring to a file or directory are specified as an attribute of that file, directory, or link in the form of a list called the access control list (ACL). Each entry of the list specifies a set of processes (actually a set of access identifiers of process) and the access modes that members of that set can use when referring to the file, directory, or link. On directory ACLs, modify mode cannot appear without list mode. On file ACLs, write and execute modes cannot appear without read mode.

If the requesting user is the creator of the file or catalog for which the action was requested, the user is considered

to have permission. If the request is not from the creator, only those access modes actually granted in an ACL entry are given to the specified user process. For example, if the ACL of a file contains the modes read and execute for a specific user, then that user's progress can fetch data from the file and transfer to and execute instructions in the file, but it cannot modify data in the file (because the write access mode was not granted).

Each ACL entry designates certain access modes and identifies the users that have these access modes. Since each user process is identified by an access identifier when the process is created (explained above), it would seem logical to use that identifier to designate a user in the ACL entry. However, if access identifiers were used to specify every user process that could access a file or directory, the ACLs would become extremely cumbersome.

Groups of user processes can be identified by a process class identifier, which, like the access identifier consists of a four-component name:

<person_id>.<account_id>.<node_id>.<domain>

Although it can be identical to an access identifier, a process identifier generally is used to designate groups of processes by replacing one or more components with an asterisk (*). Since the asterisk means that any string can be in that component position, a process identifier with an specifically named components (i.e., those components that are not an asterisk). For example, rather than requiring multiple entries, the user could specify all members of an account, regardless of the person_id, network_node, or domain by using one entry:

.DEMO..* re

2.11.1.4 Matching Entries on an ACL

A single process can be a member of more than one process class. This situation can lead to ambiguities on ACLs when more than one entry applies to the same process. To eliminate this ambiguity and make ACLs more easily readable, four conventions are imposed on ACLs and their interpretation. First, no process class identifier can appear more than once on any ACL. Second, the ACL is ordered as explained below. Third, the entry that applies to a given process is the first entry on the ordered list whose process class contains the given process. Finally, if no entry exists on the list for a given process, that process has no access to the file, directory, or link. These conventions ensure that the access for every process is uniquely specified by the ACL.

To properly generate and modify ACLs, it is necessary to have some understanding of how they are ordered. The ordering rule is to distinguish between specifically named components and asterisks, beginning with the leftmost component of the process identifier. The entries to be ordered are first divided into two groups, those whose first (Person_id) component is specific (i.e., not an asterisk) and those whose first component is an asterisk. Those with a specific first component are placed first on the ACL. Within these two groups, a similar ordering is done by second (Account_id) component with the specific entries again being first. This produces four groups. Within each of these four groups, a similar ordering is done on the third (node_id) component to produce eight groups.

In order to ensure that these service processes have access, the file system routines automatically place the ACL entry:

```
*.SYSTEM.*.*    rw
```

on the ACL of every file, and the ACL entry:

```
*.SYSTEM.*.*    lmc
```

on the ACL of every directory when the file or directory is created or its ACL is entirely replaced. In this way, members of the System project are automatically granted the necessary access so that they can perform their functions; individual users need not remember to put the proper entries on all of their file and directory ACLs to make use of the service processes.

Under special circumstances, some user might not wish to use the facilities of a service process on some of his files. In this case, the user simply denies that service process access to his files by modifying the ACL entry (i.e., giving that service process null access). It is crucial that a user who elects not to receive such a system service be fully aware of the nature of the service and the consequences of his choice. For example, if the backup processes are not permitted access to a file, backup copies of the file cannot be made and the file will not survive certain types of system failure.

2.11.1.5 Initial_ACL's

In addition to automatically adding a service process entry to the ACLs of all newly created files and directories, many system commands and subroutines (e.g. create and create_dir) add an entry for the creating process to the ACL of a newly created file or directory. For convenience, the system

allows a user to specify a list of entries to be added to all newly created files or directories, in addition to entries for the creating process and the service processes. This ability eliminates the need to explicitly modify an ACL each time a new file or directory is created.

This list of ACL entries to be added to newly created segments or directories is called an initial access control list or initial ACL and is an attribute of a directory. Each directory has two sets of initial ACLs, one set for file appended to the directory and one set for directories appended to the directory. Then, the appropriate initial ACL (i.e., either the one for files or the one for directories) of the containing directory is merged with the ACL. The merging of two ACLs means that the entries are combined and sorted. If two entries on the resulting ACL contain the same process class leaving the newly added entry. In this way, the service process entry originally on the file can be overridden by placing an entry with process class identifier *.system.* on the initial ACL. Finally, any entries specified in the operation of creating the file or directory (for most system commands this is simply one entry for the creating process) are merged into the ACL. These entries override the service process and initial ACL entries if duplicate process class identifier exist.

The initial ACL entries have the same format as described above. In addition, a special character "$" can be used in place of either the person_id or the account_id components to request that the file or directory creator's account _id are to be substituted. This method must be used whenever it is necessary to create an ACL on the new file or directory that has different permissions than would automatically be provided by the create function.

The default value for the initial ACLs of a newly created directory is empty, i.e., there are no entries in the initial ACLs.

2.11.2 Domains

To be supplied.

2.11.3 Security_Class

The Security Class defines the security authorization necessary to access the object. It consists of two parts - a level and a category set.

2.11.3.1 Security_Level

The system has a fixed number of levels that are totally ordered by the relation "less than". The security level is an integer ranging from 0 (least sensitive) to 7 (most sensitive).

2.11.3.2 Security_Category_Set

The category set is a bit string of length 32 which defines the "need to know" required to access a file. For example, data may be subdivided into various categories such as engineering specifications, financial forecasts, etc. Each category would be assigned 1 bit position.

The category set of a file then defines the nature of the data it contains.

The set of all Security classes can be partially ordered. Note that not all security classes are related by the partial ordering, i.e., two processes with respective security classes <secret,atomic> and <secret,NATO> are not comparable. The security classes and the relation "less than" define a lattice since there is a minimum and maximum level and a set of categories. This will ensure that information can only flow upward or remain at the same level, unless specific rights to reduce the level have been specified.

The security class ca only be changed by the security administrator.

2.11.3.3 Security_Class_Checking

When attempting to read, Execute, or list a file, the following conditions must apply in addition to an ACL match:

- o user's security level >= file security level.
- o user's category set >= file category set.

Where the >= for the category set comparison means that the user's category set must include the file's category set or be equal to it.

For write, recovery, or modify access, the following conditions must apply:

- o user's security level = file's security level.
- o user's category set = file's category set.

The star convention does not apply to security levels and category sets.

The reason for the difference between the read and write checking is to guarantee that the security of data may only be upgraded (made more sensitive) and never down graded except by admininstrative action, not by copying it into an non secure file.

Some method must be supplied which will permit the security administrator to create a condition where a process may down grade the security of data.

2.11.4 Passwords

Unauthorized used of directories an files can also be prevented by means of passwords. The passwords can optionally be specified for any directory, file or file generation set.

If a password is specified, every reference to the names for which it is specified must be accompanied by the password. If no password is specified, however, none can accompany the name.

When a password does accompany a file or directory name, the name is written followed by a dollar sign ($) and the password. The passwords are at least 1 and not more than 12 characters. It may consist of upper and lower case alphabetics (A-Z,a-z) and numerics (0-9).

Instead of a single password being specified for a file or directory, a list of passwords can be specified, along with the time of day during which each of the passwords applies. This allows several passwords to be specified, each of which may be used during the specified time of day.

Another use of timed passwords is to constrain the interval of time during which the file may be referenced. Once a file has been allocated (by submitting a request for it during the time that it can be referenced), the file can be read or written beyond that time for as long as the file allocation remains active.

To specify a timed password, each password is followed by a colon (:), the starting time, a comma, and the ending time. Any subsequent timed passwords in the list will be preceded by a blank character. The format of timed passwords is:

-PW <password1>:<start_time>,<end_time> <password2>:<start_time>,<end_tim
<password_n>:<start_time>,<end_time>

The <start_time> and <end_time> are composed of the day of the week (MO,TU,WE,etc.) followed by the time in hours and minutes using a 24 hour clock. If the day of the week is not specified, the time will apply for each day of the week.

2.11.5 Auditing

An auditing mechanism is provided that allows still another level of security in the form of a monitoring and reporting capability based on authorized or unauthorized attempts to use files or modify the catalog structure. The following events may optionally be monitored and recorded as they occur on a specially designated system file:

The following events that can affect the security of a file will be recorded:

1. creation, modification or deletion of an account or user.

2. creation, modification or deletion of directories, links, generation sets, and file descriptions.

3. creation of a new generation in a generation set

4. setting or resetting of an abort lock for a file.

5. allocation of a file when the request is denied because required access rights is missing or if the request is accepted or denied.

6. attempt to perform an action against the file which is not permitted - such as a write to a file when only read allocation was requested and granted.

The events are recorded on a Statistical Collection File. The events can then be reported by means of a program that reads the records from the Statistical Collection file and produces reports for each record type.

To select the recording for these events, the installation enters a control card at Startup time, indicating whether or not audit records are to be produced. The same control card

is used to indicate whether the operator can select or withdraw the selection of those record types.

In addition, records are produced in response to events 4 and 5 above only if the file allocated has had auditing specified for it. The auditing specification indicates whether all or only denied allocation requests are to be recorded.

2.12 Accounting and Statistics

Accounting, in the context of FMS, consists of the recording and reporting of space utilization on disk volume sets. Disk file space statistics are maintained at the account level as well as the individual file level. The type of information desired at the account level includes the following:

1. Total space currently allocated for all cataloged files and directories, subordinate to an account.

2. Time and date of the last change to the amount of cataloged space allocated.

3. Accumulated product of cataloged file space and time up to time of last change.

When a change in space occurs - either an increase or decrease - the space currently assigned (field 1) is multiplied by the difference between the current time and time of last change in space (field 2) and the product added to the existing accumulation (field 3). Then the total space currently assigned (field 1) is changed by the increase or decrease in space that has just occurred, and the time of last change in space (field 2) is reset to the current time.

The result of this procedure is that space utilization is measured in terms of how long the space was used rather than how much is in use at the time a report of space utilization is requested.

To obtain a record of space utilization, the system administrator can submit a privileged command to cause a Statistical Collection File (SCF) record to be produced for:

1. all accounts

2. an individual account

When the record is produced, the time of last change (field 2) is reset to the current time and the accumulation (field

3) is zeroed. In this way, overflow of the accumulation and ambiguity in time is prevented by requesting a space utilization record. The field capacities are such that a record is required for this purpose at least once every two years.

In addition to producing a SCF record, the installation receives a display of the accumulation brought up-to-date when a master or pack list is prepared. The display does not zero the accumulation or reset the time of last change, however, as an SCF record production does, since it is expected that the report will be used only for information and not to prepare accounting charges, as the SCF record is used.

3.0 External_Interfaces

3.1 Software_Interfaces

The FMS Services are divided into two sets; those available to the user through ECL commands and direct calls from a user program and those services only available to System Programs such as resource manager and UFAS.

3.1.1 ECL_Commands

3.1.1.1 Create_Directory

This function will create a new directory that may be referenced by the indicated path name.

Format:

CREATE_DIR <pathname> [<ctrl_args>]

Where:

<pathname>::=

is the name of the directory being created. It may be either a relative or absolute pathname.

<ctrl_args>

are control arguments taken from the following set:

-RF ::=

indicates that the subordinate disk files are related and must be maintained at the same update base. Refer to Section 2.5.4 for additional details.

-PW<password>::=

is the password that must be supplied for each reference to the directory.

Notes:

1. The newly created directory's ACL will consist of an entry for the creator plus any entries contained in the superordinate directories Initial_ACL_DIR.

2. This function may be executed by anyone that has create permission on the superior directory.

3. The absolutized pathname must be unique.

4. All superior directories must exist.

3.1.1.2 Modify_directory

This function will modify attributes of an existing directory.

Format:

MODIFY_DIRECTORY <pathname> <ctrl_args>

Where:

<pathname>::=

is the name of the directory to be modified.

<ctrl_args>

are control arguments selected from the following set:

-PW<password>::=

is the password that must be used when accessing the directory.

-RF::=

indicates that the subordinate disk files are related and must be maintained at the same update base. Refer to Section 2.5.4 for additional details.

3.1.1.3 Delete_Directory

The function will cause the specified directory to be deleted. In addition, all subordinate directories, links and files will be deleted and space that they occupy will be returned to the system.

Format:

DELETE_DIR <pathname> <ctrl_arg>

Where:

<pathname>::=

is the name of the directory to be deleted. The star convention can be used.

<ctrl_arg>

The following control argument can be used with this function:

-PURGE or -P ::=

specifies that all subordinate files must be overwritten prior to being deleted to protect sensitive data.

Notes:

1. This function may be executed by anyone with modify or delete permission.

2. If a subordinate file to be purged resides on a tape volume set that is not mounted an indication will be made in the corresponding volume set definition entry to cause the overwrite of the data the next time the volume set is mounted.

3. If any subordinate file is busy, the release of the file will be delayed until the file is deallocated.

3.1.1.4 Create_File

This function will create a new file which may be referenced by the indicated pathname.

Format:

CREATE_FILE <pathname> [<ctrl_args>]

Where:

<pathname>::=

is the name of the file being created. It may be either a relative or absolute pathname and may include a specific generation identification

<ctrl_args>

are control arguments chosen from the following set:

-CI_SIZE or -CISZ <size>::=

is the number of bytes in a control interval. The value <nnnn> must be a multiple of 256 bytes. The default is 512 bytes.

-SIZE or -SZ <size-1>[,V<rel_vol_number>|S<number_vol>]
[<size-2>,V,rel_vol_number>]... ::=

This entry is used to express the initial size of a disk file expressed in control intervals and optionally the explicit placement of the file.

Size-1 is required to specify the number of control intervals in the first (and possibly only) section of the file. The rel_vol_number is used to indicate an explicit placement of the file segment on a specific volume of the volume set. If the field is not supplied, FMS will dictate the file placement. An S, plus a number indicates that the file is to spread equally across n volumes.

Subsequent pairs of parameters, size and rel_vol_number, may be used to explicitly control the spreading of the file to other volumes in the volume set.

Examples:

```
FC A -SIZE 100
```

Create a file A, consisting of 100 control intervals. The file space will be placed somewhere on the same volume set as the catalog.

```
FC B -SIZE 200,S
```

Create file B, consisting of 200 control intervals, spread across the volume set that contains the catalog.

```
FC C -SIZE 100,V1
```

Create file C, consisting of 100 control intervals on volume 1 of the volume set containing the catalog.

```
FC D -SIZE 50,V1 25,V2 25,V4
```

Create a file D, where 50 control intervals (CI's) will be placed on volume 1 of the volume set, 25 CI's on volume 2, and 25 CI's on volume 4.

-TAPE <vol_set_name> ::=

this option is used to indicate that the file will reside on a predefined volume set.

-STAPE [<scratch_vol_set>][,nn] ::=

is used to indicate that the file should reside on tapes obtained from a scratch volume set. If a scratch_vol_set is specified, the scratch tapes will be obtained from that volume set. If not, the tapes will be obtained from the system_scratch_vol_set. The user must have modify permission on the specified scratch_vol_set.

The number of volumes required (nn) may also be specified. The default is one tape.

-DEN<nn>::=

if scratch tapes have been requested, this option is used to indicate the tape's density. Where 2,5,8,16,64 or STD (standard) are valid options. STD is the default.

-TRK {7|9} ::=

if scratch tapes are requested, this option indicates whether 7 or 9 track format should be used. If not specified, the site standard will be used.

-MAX_SIZE or -MSZ <size>::=

the maximum size, expressed in control intervals, to which the file may be grown. The default is unlimited growth.

-INC_SIZE or -ISZ <size>::=

is the number of control intervals the file will be grown each time the file is dynamically grown. The default will be by one eighth of the current file size.

-REC_MOD or -RM ::=

-RDATE {mmddyy|+nnn} ::=

indicates that the file must be retained until after the specified date. The date may be expressed as a calendar date (mmddyy) or as a relative date (+nnn). If a relative date is specified, the retention date will be calculated and saved in mmddyy format.

-MONITOR ::=

-ROLLBACK or -RB ::=

indicates that a before image should be taken for each change to the file content and that the file will be rolled back following any program or system abort.

-JRNL or -J ::=

indicates that all changes to the file content will be journaled.

-DUP [<size>]::=

specifies that a duplicate file should be created for this file to protect against device failure. The size and location parameters allow the size and location of the duplicate file to be explicitly specified. Refer to the -SIZE option for a description of the parameters. If the size is not specified, FMS will obtain space on volumes in the volume set not assigned to the original file.

-VERIFY or -V ::=

specifies that each write to the file must be verified.

-PURGE or -P ::=

will cause the file to be overwritten whenever it is released.

-AUDIT or -AUD {ALL|DENIED} ::=

indicates that allocation requests for the file should be audited. The parameter also indicates whether all allocations or only denied allocations are to be monitored.

-PW<password>::=

is the password to be supplied when requesting access to the file.

-IFN <internal_file_name>::=

Allocate the newly created file using the specified IFN.

This option assumes that the ECL command interpreter is available.

Notes:

1. The newly created file's ACL will consist of an entry for the creator plus any entries contained in the superordinate directories Initial_ACL_File.

2. The absolutized pathname must be unique.

3. This function may be executed by anyone that has create permission on the superior directory.

3.1.1.5 Modify_File

This function allows a user to modify attributes of an existing file

Format:

MODIFY_FILE <pathname> <ctrl_args>

Where:

<pathname>::=

is the name of the file to be modified. It may be either an absolute or relative pathname and may include a specific generation identification (see naming convention section 2.5)

<ctrl_args>

are control arguments chosen from the following set:

-MAX_SIZE or -MSZ <size> ::=

The maximum size of the file expressed in Control Intervals.

-INC_SIZE or -ISZ <size> ::=

is the number of control intervals to be added to the file each time the file is dynamically grown.

-PW<password>::=

is the new password to be used when requesting access to the file.

-AUDIT or -AUD {ALL|DENIED}::=

indicates that allocation requests for the file should be audited. The parameter also indicates whether all allocations or only denied allocations are to be monitored.

-RDATE {mmddyy|+nnn}

defines a new retention period for the file. The date may be expressed as a calendar date (mmddyy) or as a relative date (+nnn). If a relative date is specified, the retention date will be calculated and the date saved in mmddyy format.

-WRITE_LOCK or -WL ::=

prevents the file from being accidentally overwritten. Once set the lock can not be removed.

-VERIFY or -V ::=

specifies that each write to the file must be verified.

-MONITOR ::=

-ROLLBACK or -RB ::=

indicates that a before image should be taken for each change to the file's content and that the file will be rolled back following any program or system abort.

-JRNL or -J ::=

indicates that all changes to the file content should be journaled.

-DUPE[<size>] ::=

indicates that a duplicate file should be created for this file to protect against device failure. Refer to the -D option on the File Create for additional information.

-PURGE or -P ::=

Specify that the file must be overwritten prior to being released.

Notes:

1. This function may be executed by anyone with modify permission for the file.

2. Any of the arguments except MAX_SIZE, INCR_SIZE, and WRITE_LOCK may be negated by preceding the argument name with an N. For example, -ND will remove duplicate file protection from the file.

3.1.1.6 Delete_File

This function will cause the specified file to be deleted.

Format:

DELETE_FILE <pathname> <ctrl_arg>

Where:

<pathname>::=

is the name of the file to be deleted. The star convention can be used.

<ctrl_arg>

The following control argument can be used with this function:

-PURGE or -P ::=

specifies that the file must be overwritten prior to being released to protect sensitive data.

-FORCE ::=

this option will cause the file to be deleted even if the retention period has not expired.

Notes:

1. This function may be executed by anyone with modify or delete permission for the file.

2. If the file being deleted resides on a tape volume set, the associated volume set definition will be deleted if the tape volume set has been dynamically created for the file. However, if the tape volume

set had been explicitly created through Media Management, the volume set definition will not be deleted but will be modified to remove the file's relationship with the volume set.

3. If the file is busy, the deletion of the file will be deferred until the file is deallocated.

4. If the file is to be purged and it resides on a tape volume set that is not mounted an indication will be made in the corresponding volume set definition to cause the overwrite of the data the next time the volume set is mounted.

5. If the current file of a generation is to be released, the next oldest generation will be made current.

3.1.1.7 Create_Link

This function creates a link entry with a specified name pointing to a specified directory or file. This link may be used as an indirect pointer to a :

- o directory or file under another account
- o directory or file within the same account
- o directory or file on another disk volume set
- o file at another site

Format:

CREATE_LINK <pathname> <pointer> -[MBX<mailbox name>]

Where:

<pathname>::=

is the name of the link entry to be created

<pointer>::=

is the absolute pathname of the file or directory to be referenced whenever this link entry is used.

-MBX<mailbox_name>::=

is the mailbox name of the file transfer workstation at the target file site. This entry must be provided anytime the file being referenced is located at a remote site.

Notes:

1. The absolutized pathname must be unique.

2. The pointer and mailbox will not be validated during the creation of the link.

3. This function may be executed by anyone that has create permission on the superior directory.

4. The creator will be assigned list, modify and delete permission in the created link's ACL.

3.1.1.6 Modify_Link

This function will cause the pointer or mailbox to be altered in an existing link entry.

Format:

MODIFY_LINK <pathname>[<pointer>][-MBX<mailbox name>]

Where:

<patnname>::=

is the name of the link entry to be modified.

<pointer>::=

is the new absolute pathname to be referenced whenever the link entry is used.

-MBX<mailbox name>::=

is the new mailbox name of the file transfer work station at the target file site. This entry must be provided whenever the file being referenced is located at a remote site.

Notes:

1. The absolutized pathname must exist.

2. The new pointer and/or mailbox will not be validated by this function.

3. This function may be executed by the link's creator or anyone with modify permission.

3.1.1.9 Delete_Link

This function deletes the specified link entry. It does not delete the entry pointed to by the link.

Format:

DELETE_LINK <pathname>

Where:

<pathname>::=

is the name of the entry to be deleted.

Notes:

1. This function may be executed by the links creator or anyone with modify or delete permission for the link.

3.1.1.10 Create_Generation_Set

This function creates a new file generation set that may be referenced by the specified pathname.

Format:

CREATE_GEN_SET <pathname> <ctrl_args>

Where:

<pathname> ::=

is the name of the file generation set to be created.

<ctrl_args>

are optional control arguments to be selected from the following set:

-PW <password> ::=

is the new password to be appended to the file generation set name when accessing either the generation set or any member of the set.

-CYCLE <nnn> ::=

is the number of generations to be maintained in a fixed cycle.

-GENS <nnn> ::=

specifies the number of generations that are to be maintained in the generation set. The generations will be maintained as a dynamic set, where new generations will be created based on the default file definition for the set and the expired generations will be released from the set.

-RDATE {mmddyy|+nnn} ::=

is a request to retain each generation until the specified date. The date may be expressed as a calendar date (mmddyy) or as a relative date (+nnn). If a relative date is specified, the retention date will be calculated when each new generation is created.

-DFC /<file create options>/ ::=

defines the option to be used for each default file create. The allowed options are described with the File Create function.

If this option is not supplied, the system default will be used. This default initially specifies a mass storage file that consists of one 512 byte Control Interval, but may be modified by the System Administrator.

-MAXG <nnn> ::=

This option is used to limit the number of generations to be retained. It may use anytime the the generation set is controlled by a retention date (-RDATE).

The number of generations specified may be from 1-256. The default will be 30.

Notes:

1. The newly created generation set's ACL will consist of an entry for the creator plus any entries contained in the superordinate directories Initial_ACL_File.

2. The absolutized pathname must be unique.

3. This function may be executed by anyone that has create permission on the superior directory.

4. No more than one generation type option (-CYCLE, -GENS, or -RDATE) may be specified for a generation set. If no type option is specified, the system default will be used. The default is initially -CYCLE 3 (fixed cycle with three generation files), but may be modified by the System Administrator.

3.1.1.11 Modify_Generation_Set

This function will cause the specified options for a generation set to be updated.

Format:

```
MODIFY_GEN_SET <pathname> <ctrl_args>
```

Where:

<pathname> ::=

is the name of the generation set definition to be modified.

<ctrl_args>

are control arguments selected from the following set:

-PW <password> ::=

is the new password to be appended to the file generation set name when accessing either the generation set or any member of the set.

-CYCLE <nnn> ::=

is the adjusted number of generations to be maintained.

-GENS <nnn> ::=

is the new number of generations to be maintained.

-RDATE {mmddyy|+<nnn>} ::=

defines a new retention period for members of the generation set. The date may be expressed as a calendar date (mmddyy) or as a relative date (+nnn). The modified retention period will only apply to generations created after the change. The existing generations retention periods will not be modified by this function.

-MAXG <nnn> ::=

specifies the maximum number of generations that should be retained in a dynamic generation set.

-DFC /<file create options>/ ::=

provides a new set of option for any default file create. The allowable options are described with the File Create command. This new definition will completely replace the previous default definition for the generation set.

Notes:

1. This function may be executed by anyone with modify permission for the generation set.

2. The three generation type options (-CYCLE, -GENS, and -RDATE) can only be used to modify the associated argument (number of days or number of generations. They may not be used to modify the generation set type.

3. If the number of generations being maintained is reduced by the -CYCLE, -GENS, or -MAXG options, the oldest generations will be deleted.

3.1.1.12 Delete_Generation_Set

This function will cause the specified generation set and it's members to be deleted.

Format:

DELETE_GEN_SET <pathname> <ctrl_args>

Where:

<pathname> ::=

the name of the generation set to be deleted. The star convention must not be used with this function.

<ctrl_args>

The following control arguments can be used with this function.

-PURGE or -P ::=

indicates that the members of this generation set must be overwritten prior to being released to protect sensitive information.

Notes:

1. This function may be executed by anyone with modify or delete permission.

2. If a subordinate file to be purged resides on a tape volume set that is not mounted an indication will be made in the corresponding volume set definition entry to cause the overwrite of the data the next time the volume set is mounted.

3. If any subordinate file is busy, the release of the file will be delayed until the file is deallocated.

3.1.1.13 Add_Name

Add an alternate name to an existing directory, file or link.

Format:

ADD_NAME <pathname> <new_name>

Where:

<pathname>::=

is the name of the directory, file or link for which the additional name is to be added. The star convention can be used.

<new_name>::=

is the additional name by which the file or directory is to be known. The equal convention can be used.

Notes:

1. The absolutized new_name must be unique.

2. This function may be executed by anyone with modify permission for the entry.

3. Cataloged tape files may not have alternate names.

3.1.1.14 Delete_Name

This function will delete a name from a directory, file or link.

Format:

DELETE_NAME <pathname>

Where:

<pathname>::=

is the name that is to be deleted. The star convention can be used.

Notes:

1. Only the specified name is deleted, not the associated file, directory or link. If the execution of the command would remove the last name from the entry the request will be denied.

2. This function may be executed by anyone with modify permission for the entry

3.1.1.15 Rename

This function replaces a name of a directory, file or link with a new name, without affecting any other names the entry may have.

Format:

RENAME <pathname> <new name>

Where:

<pathname>::=

is the existing name that is to be replaced. The star convention can be used.

<new_name>::=

is the name that replaces the entryname portion of the pathname. The equal convention can be used.

Notes:

1. If the last entry name of the pathname is a star name it indicates that more than one entry name may be replaced. In this case the new_name must use the equal name convention.

2. This function may be executed by anyone with modify permission for the entry.

3. The absolutized new name must be unique.

4. Cataloged tape files can not be renamed.

3.1.1.16 Add_File_Space

This function is used to obtain additional space for an existing disk file.

Format:

ADD_SPACE <pathname>

Where:

<pathname>::=

Is the name of the file to be grown

<ctrl_args>

optional control arguments selected from the following set:

-SIZE<size>[,<rel_vol_number>]::=

the size specifies the number of control intervals to be added to the file. The optional rel_vol_number is used to indicate an explicit placement of the new file extent.

-DUP<size>[,<rel_vol_number>]::=

is used to explicitly control the placement of an additional extent for a duplicated file. If the argument is not specified and the file is duplicated, FMS will grow the duplicate file.

-MAX_SIZE or -MSZ <size>::=

this is the maximum size to which the file may be grown. If this option is used without a size specified, unlimited growth will be allowed.

Notes:

1. This function will only grow the named file. Therefore, when an indexed file is grown, the index file may also need to be grow.

2. This function may be used by anyone with modify or write permission for the file.

3.1.1.17 Release_File_Space

This function causes the space assigned to a named disk file to be reduced.

Format:

RELEASE_FILE_SPACE <pathname> <ctrl_args>

Where:

<pathname>::=

Is the name of the file to be reduced.

<ctrl_args>

optional control arguments taken from the following set:

-SHRINK or -SHR ::=

will cause all unused allocated space to be released.i.e., the extent of the allocated space will truncated at the next allocatable space unit following the current end of file. This option is only applicable to sequential files.

-SIZE or -SZ <size>::=

specifies the number of control intervals to be retained. The remainder of the file's space will be released.

-MAX_SIZE or -MSZ <size>::=

the maximum size, expressed in control intervals to which the file may be grown. The default size will be unlimited growth if a specific size is not specified.

Notes:

1. The modify volume set command should be used to add additional volumes to a tape volume set.

2. This function may be used anyone that has modify or write permission on the file.

3.1.1.18 Set_Abort_Lock

This function will either set or reset the abort lock for a specified file.

Format:

SET_ABORT_LOCK <pathname> {-ON|-OFF}

Where:

-ON ::=

will set the abort lock ON. Thus, preventing allocation to the file unless the allocation requests specifies QUERY or RECOVERY.

-OFF ::=

will set the abort lock OFF.

3.1.1.19 Change_Working_Directory

This function changes the user's current working directory to that specified by the indicated path.

Format:

CHANGE_WORKING_DIRECTORY <pathname>

Where:

<pathname>::=

identifies the new working directory. This may either be an absolute pathname or a relative pathname that is relative to the working directory at the time that the command is issued.

Notes:

1. If a pathname is not supplied the current working directory will be set to the default home directory defined in the user profile.

2. In case pathname is syntactically incorrect, and thereby is not a valid directory pathname, a message is returned indicating the first part of pathname which was not understood. In this case, the working directory is the same as it was before the incorrect pathname was given.

3.1.1.20 Set_ACL

This function is used to add or modify an access control entry in the indicated file, file generation set or directory.

Format:

```
SET_ACL <pathname> <access_right> <access-identifier>
                  [<ctrl_args>]
```

Where:

<pathname>::=

is the pathname of the file, file generation set or directory to which the command applies. If pathname is -WD, the working directory is used. The star convention can be used.

<access_right>::=

is a valid access mode for directories or files, depending on the object of the pathname.

<access_identifier>::=

is an access control entry name that must be of the form:

<person_id>.<account_id>.<node_id>.<domain>

Where any of the components may be replaced with an "*".

<ctrl_args>

Control arguments selected from the following set:

-DIR or -D ::=

specifies that ACL's for directories should be modified.

-FILE or -F ::=

indicates that ACL's for files or file generation sets should be modified.

Notes:

1. Any existing ACL entries with a matching access identifiers will receive the <access_right>. If no match is found an entry is added to the ACL.

2. The user must have modify access right on the specified directory or file.

3. If the access identifier is omitted, the user's <person_id>.<account_id>.* is used.

4. If a file is a member of a generation set an ACL will exist only for the generation set not for each member of the set. Therefore a suffix specifying a specific generation may never be used with the pathname on this command.

3.1.1.21 Delete_ACL

The Delete_ACL function removes entries from the access control lists (ACL's) of files or directories.

Format:

DELETE_ACL <pathname> <access_identifier> <ctrl_args>

Where:

<pathname>::=

is the pathname of a file, file generation set or directory. The star convention can be used. If pathname is -WD or omitted, the working directory is used. If pathname is omitted, the access identifier can not be specified.

<access_identifier>::=

is an access control entry name that must be of the form:

<person_id>.<account_id>.<node_id>.<domain>

where any component may be replaced by an "*".

Notes:

1. Any existing ACL entries with a matching access identifier will be deleted.

2. The user must have modify access right for the specified directory or file.

3. If the access identifier is omitted, the user's <person_id>.<account>.* is used.

3.1.1.22 Set_Initial_ACL

The function is used to add or modify an entry to a file or directory initial access control lists in a specified directory. The file initial ACL contains ACL entries to be placed on files created subordinate to the specified directory. The directory initial ACL contains the ACL entries to be placed on any directories created subordinate to the specified directory.

Format:

```
SET_INITIAL_ACL <pathname> <access right>
                <access identifier>[<ctrl_args>]
```

Where:

<pathname>::=

Specifies the directory in which the initial ACL is to be altered. The star convention can be used.

<access_right>::=

is a valid access mode for either directories or files, depending on whether the file or directory initial ACL is being updated.

<access_identifier>::=

is an access control entry name that must be of the form:

<person_id>.<account_id>.<node_id>.<domain>

Where any of the components may be replaced by a "*".

<ctrl_arg>

Control arguments chosen from the following:

-DIR or -D ::=

specifies that the initial ACL for directories is to be modified.

-FILE or -F ::=

specifies that the initial ACL for the file or file generation set is to be modified.

Notes:

1. Any existing initial ACL entries with a matching access identifier will receive the <access right>. If no match is found an entry is added to the specified initial ACL.

2. The user must have modify access right for the specified directory.

3. If neither of the control arguments (-DIR or -FILE) is specific, the initial ACL for file is modified.

3.1.1.23 Delete_Initial_ACL

This function is used to delete entries on the file or directory initial access control lists.

Format:

```
DELETE_INITIAL_ACL <pathname> <access-identifier>
                   [<ctrl_args>]
```

Where:

<pathname>::=

is the pathname of a directory. The star convention can be used.

<access_identifier>::=

is an access control entry name that must be of the form:

<person_id>.<account_id>.<node_id>.<domain>

where any of the components may be replaced by a "*". The <person_id> or <account_id> may be replaced with a "$" to cause the creator's person_id or account_id to be substituted when a directory or file is created.

<ctrl_arg>

Control arguments chosen from the following:

-DIR or -D ::=

specifies that the initial ACL for directories is to be modified.

-FILE or -F ::=

specifies that the initial ACL for file is to be modified.

Notes:

1. Any existing ACL entries with a matching access identifier will be deleted.

2. The user must have modify access rights for the specified directory.

3.1.1.24 List

This function provides a listing of catalog entries immediately subordinate to a specified directory.

Format:

LIST <pathname> [<ctrl_args>]

Where:

<pathname>::=

The star convention can be used.

<ctrl_args>

control arguments selected from the following set:

-DIR or -D ::=

list only directories subordinate to the defined entry.

-FILE or -F ::=

list only files subordinate to the defined directory.

-LINK or -L ::=

list only links subordinate to the defined directory.

-ALL or -A ::=

list all entries subordinate to the defined directory.

-EXP ::=

list all files subordinate to the specified directory whose retention periods have expired.

-BUSY ::=

list all files subordinate to the current directory that are currently allocated.

-DTL ::=

list the attributes of the selected entries.

-BF ::=

list only the name type and size of the selected entries.

-DTM ::=

list entries in order by the date/time last modified. The most recent will be listed first.

-DTU ::=

list entries in order by the date/time last used. The most recent will be listed first.

-SR ::=

list the entries in ascending alphabetic sequence.

-RV ::=

reverse the order in which the entries are listed. This option may be used with -DTM , -DTU , and -SR.

Notes:

1. The following options (-DIR, -FILE, -LINK, -ALL, -EXP, and -BUSY) are used to determine which entries are to be listed.

   While the remaining options, specify the degree of detail required and the order of presentation.

2. *** identify the permissions required ***

3.1.1.25 List_ACL

This function lists the entries in the access control list (ACL) of a file or directory.

Format:

LIST_ACL <pathname> <access_identifier> [<ctrl_args>]

Where:

<pathname>::=

is the name of a directory or file. If the key word -WD is used or the pathname omitted, the working directory is used. If the pathname is omitted, an access identifier cannot be specified. The star convention can be used.

<access_identifier>::=

is an access control name that must be in the form:

<person_id>.<account_id>.<node_id>.<domain>

All matching ACL entries will be listed. If the <access_identifier> is omitted, the entire ACL will be listed.

Notes:

1. The user must have modify or list access rights for the specific directory or file.

2. If the command is invoked with no arguments, the entire ACL on the working directory is listed.

3.1.1.26 List_Initial_ACL

This function is used to list the entries of the file or directory initial access control list in a specified directory.

Format:

```
LIST_INIT_ACL <pathname><access_identifier>
                    [<ctrl_args>]
```

Where:

<pathname>::=

is the name of a directory. If the name is omitted or -WD specified , the working directory is used. If the pathname is omitted, an <access_identifier> cannot be specified. The star convention can be used.

<access_identifier> ::=

is an access control name that must be in the form:

<person_id>.<account_id>.<node_id>.<domain>

All initial ACL entries with matching names are listed. If the <access_identifier> is omitted, all entries on the requested initial access control list are listed.

<ctrl_args>

optional control arguments may be selected from the following set:

-ALL or -A ::=

cause all entries on the requested initial access control list to be listed. This argument is redundant if <access_identifier> is omitted. It overrides the <access_identifier>, if both are specified.

-DIR or -D ::=

specifies that the directory initial access control list is to be modified.

-FILE or -F ::=

specifies that the file initial access control list is to be altered.

Notes:

1. The user must have list access for the specified directory.

3.1.1.27 List_Working_Directory

This function will list the absolute pathname of the user's current working directory.

Format:

```
List_Working_Directory
```

Notes:

1. There are no options for this function.
2. A user need not have any access rights to use this function.

3.1.1.28 Walk_Subtree

The WALK_SUBTREE command causes the rest of the command line to be executed at each level of the catalog structure subordinate to the specified directory.

Format:

WALK_SUBTREE <pathname> <command_line> <ctrl_args>

Where:

<pathname>::=

is the name of the starting directory. If -WD, then use the current working directory.

<command_line>::=

the command line to be executed. Enclose the command in quotes if embedded separators are included.

<ctrl_args>

optional control arguments taken from the following set:

-FIRST <n> ::=

specifies the first level of the catalog structure to be used. Default is 1, i.e. the directory specified by the pathname.

-LAST <n> ::=

specifies the last level of the catalog structure to be used. Default is 999(all levels).

-BRIEF or -BF ::=

suppress the printing of the directory names during the execution of each <command_line>.

-BU ::=

execute the <command_line> from the last level to the first level in reverse order from the normal first to last order.

3.1.2 Directory_Active_Functions

The following Active Functions are initiated as a result of an active string detected in an ECL command line. They are immediately evaluated and will return a character string representing information about entities in the Directory hierarchy. The information that is returned will replace the Active Function name in the command line.

3.1.2.1 LWD

Format: [LWD]

returns the current working directory pathname.

3.1.2.2 FILES

Format: [FILES [<directory>] <star_name>]

returns a list of simple file names separated by spaces. The returned file names are those that match using the star convention as expressed by <star_name>. The current working directory is used unless a <directory> is specified.

3.1.2.3 DIRS

Format: [DIRS [<directory>] <star_name>]

same as FILES, except that this function applies to directories.

3.1.2.4 LINKS

Format: [LINKS [<directory>] <star_name>]

same as FILES, except that this function applies to links.

3.1.2.5 HOME_DIR

Format: [HOME_DIR]

returns the pathname of the user's home directory.

3.1.2.6 ENTRY

Format: [ENTRY <pathname>]

returns the entry name portion of the absolute pathname developed from the specified pathname.

3.1.2.7 STRIP

Format: [STRIP <pathname> <str>]

returns the absolute pathname of the specified pathname with the the last component removed. If str is specified, the last component is removed only if it matches the <str> value.

3.1.2.8 SPE

Format: [SPE <pathname> <str>]

same as STRIP except that the it returns the entryname portion of the absolute pathname.

3.1.2.9 DIR

Format: [DIR <pathname>]

returns the directory portion of the absolute pathname derived from the specified pathname.

3.1.2.10 VOLID

Format: [VOLID <pathname>]

returns the root volume id of the absolute pathname for the specified pathname.

3.1.3 Program_Services

Any program can request FMS services in the course of execution of that program. These services include those available through ECL commands, Directory Active Functions, and services described below which are only available to programs.

3.1.3.1 Requesting_Program_Services

A CLIMB to the Command Interpreter is used for each call for service except WALK_SUBTREE. The first parameter describes a command block. The second parameter describes the service to be executed, expressed in ECL syntax. The third parameter describes a segment where the output from the function will be stored.

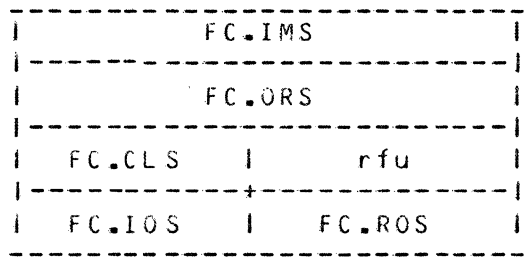
Format

```
ICLIMB CMDI,3
```

Parameters

- descriptor 0 : frames the Command Block
- descriptor 1 : frames the command line
- descriptor 2 : frames the output area

Command Block



Input

FC.CLS Command line size
FC.IOS Initial output size
FC.ROS Result output size

Output

FC.IMS Immediate status
FC.ORS Original status

Status Codes

The status codes returned will vary based on the command initiated.

3.1.3.2 Read_File_Attributes

This function retrieves the file attribute block for a named file. The request must specify either the internal file name if the file is allocated or the external file name. The third descriptor of the call defines where to store the obtained attributes.

Format:

Read_File_Attr [<pathname>|-IFN<ifn>]

Where:

<pathname> ::=

pathname of the file whose attributes are to be obtained.

-IFN<ifn> ::=

is the internal name assigned to an allocated file.

Notes:

1. The -IFN option may be used only if the file is allocated. Pathname may be used for any file, allocated or not.

3.1.3.3 Return_Allocation_Information

This function allows a process to obtain information about the mass storage files currently allocated to that process.

The third descriptor of the call defines where to return the allocation information. This information includes a summary of the number of mass storage files allocated and file protection services being used. This will be followed by a list which contains one entry for each allocated file. Each entry specifies the

Internal File Name, Allocation type (R, W, A, E, test mode etc.) and file protection options.

Format:

Return_Alloc_Info [-IFN<ifn>]

Where:

-IFN<ifn> ::=

is the internal name assigned to an allocated file.

Notes:

1. If -IFN is specified, allocation information is returned only for that file.

2. If -IFN is omitted, allocation information will be returned for all files assigned to the user.

3.1.4 System_Services

The FMS System Services consist of a set of functions available only to system processes such as Peripheral Allocator and access methods such as UFAS and IDS.

These services are requested via a CLIMB to the FMS domain with the function code selecting the particular function to be called. The first parameter always describes a command block. Additional parameters may be required depending on the function requested. A description of the parameters and command block is provided with the description of the individual system functions.

3.1.4.1 File_Allocation

Before any operation can be performed on the contents of a file (such as open, read or write), the file must be allocated to the user. This function, which results from a GET command or from a resource template interrogation, provides the means by which the Peripheral Allocator requests allocation for cataloged files.

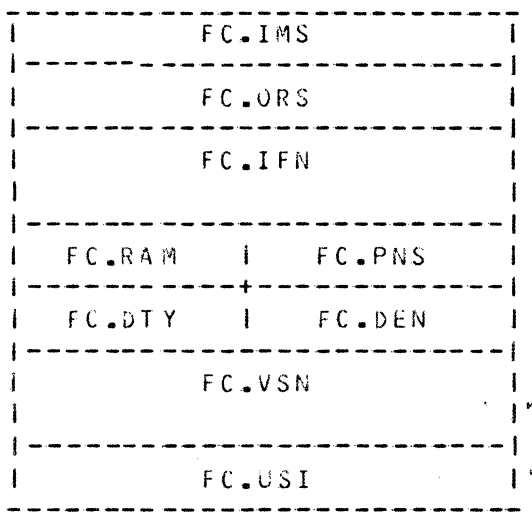
Format

```
ICLIMB   FMS,3,FALL.F,EAX0
```

Parameters

- descriptor 0 : frames the Command Block
- descriptor 1 : frames the pathname
- descriptor 2 : frames space where the PAT will be built

Command Block



Input

FC.IFN Internal name for the file
FC.RAM Requested access mode
FC.PNS Pathname size

Output

FC.IMS Immediate status
FC.ORS Original status
FC.APG Access permissions granted
FC.DTY Device type
FC.DEN Density (for tape files only)
FC.VSN Volume Set name
FC.USI User supplied information

Status Codes

4 File is busy
15 File is abort locked or contains defective space
25 File is on a disk pack that is not currently mounted
53 Read only access requested for a null content file

Function

The catalog structure will be searched for a match on pathname and the file's ACL will be checked to determine whether the user may access the file in the requested access mode (FC.RAM). If permission is granted, this function will build a PAT. This PAT will be used by IOS to map file addresses to device and device address, and to constrain operations on the file to those granted by allocation.

On a successful allocation, this function will return:

- o Volume Set Name

The pathname may reference any cataloged file. That includes a generation group or an individual generation file.

3.1.4.2 File_Allocation_Test

This function is used by the Peripheral Allocator to determine the availability of a cataloged file without reserving the file. It determines whether the file exists and if the user has permission to access the file.

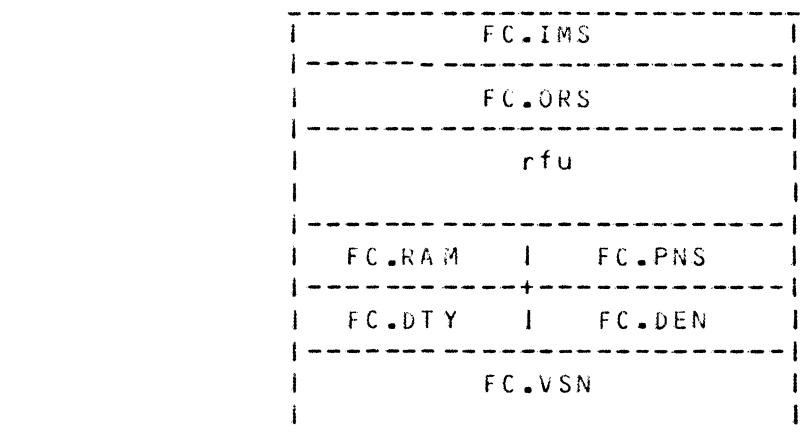
Format

```
ICLIMB   FMS,2,FTAL.F,EAX0
```

Parameters

- descriptor 0 : frames the Command Block
- descriptor 1 : frames the pathname

Command Block



Input

FC.RAM Requested access mode
FC.PNS Pathname size

Output

FC.IMS Immediate status
FC.ORS Original status
FC.DTY Device type
FC.DEN Density (for tape files only)
FC.VSN Volume Set Name

Status Codes

4 File is restore locked
15 File is abort locked and access request is not for query or recovery
25 File is on a disk pack that is not currently mounted
26 File is on a tape that may may or may not be mounted
53 Read only access requested for a null file

Function

The catalog structure will be searched for a match on pathname and the file's ACL will be checked to determine if the user may access the file in the requested access mode. If the user has permission to access the file this function will return:

- Volume Set Name

o Device Type

o Density (for tape file only)

3.1.4.3 File Deallocation

This function is used by the Peripheral Allocator to deallocate a cataloged file upon termination of a run type command, execution of a REMOVE command and upon termination of a process.

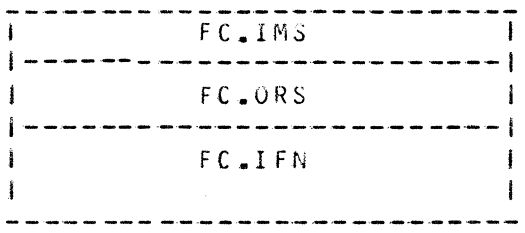
Format

```
ICLIMB   FMS,1,FDAL.F,EAX0
```

Parameters

- descriptor 0 : frames the Command Block

Command Block



Input

FC.IFN Internal name for the file

Output

FC.IMS Immediate status
FC.ORS Original status

Status Codes

24

Function

The user's PAT Pointers will be searched for a matching IFN. If a match exists, the count of outstanding allocations against the file will be decreased and the PAT removed.

If the file was written to, the change date and time is updated in the file description. If a new generation file was created and the job terminated normally, the current pointer is updated to reflect the new generation file and the absolute generation number is increased by one.

3.1.4.4 File_Grow

This function is used by the buffer manager to dynamically grow a mass storage file when a user is writing a sequential file and has requested a write to a control interval that is beyond the end of the current file.

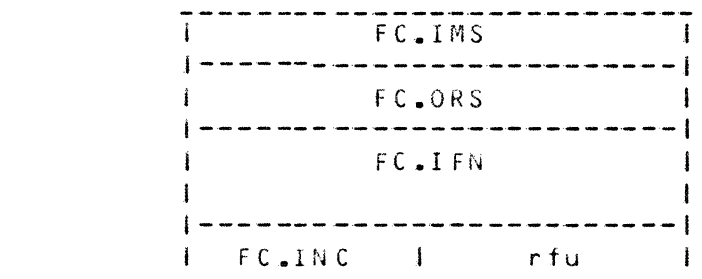
Format

```
ICLIMB  FMS,1,FGRO.F,EAXO
```

Parameters

- descriptor 0 : frames the Command Block

Command Block



Input

FC.IFN Internal file name
FC.INC Number of CI's to be added

Output

FC.IMS Immediate status
FC.ORS Original status

Status Codes

Function

TBS

3.1.4.5 File Reallocation

To provide economical recovery in case of system failure, processes in execution at the time of the failure can be restarted without resubmitting the job and re-executing previous processes. On such a process restart, cataloged files must be reallocated, for the busy counts on system restart may have been reset, removing any indication of allocation from the file description for cataloged files.

This function will allow a PAT for the file to be retrieved by the process restart facility for a file. It may also be used to allow a file allocation to be continued from one process to the next. FMS will obtain the type of allocation previously granted from the PAT and PAT Pointer and try to grant the same type now.

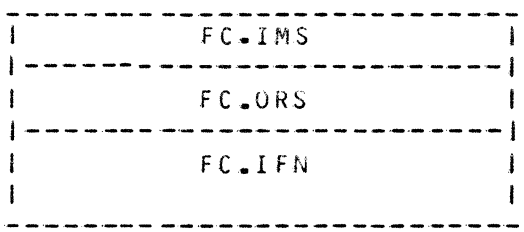
Format

```
ICLIMB    FMS,1,FRAL.F,EAX0
```

Parameters

- descriptor 0 : frames the Command Block

Command Block



Input

FC.IFN Internal name for the file

Output

FC.IMS Immediate status
FC.ORS Original status

Function

function

3.1.4.6 Update_P_A_I

This function is used by IOS when a seek is issued to a part of a cataloged file is attempted to which the descriptors in the PAT do not apply. FMS will obtain the descriptors from the File Descriptor for the file for that part of the file sought and place them in the PAT.

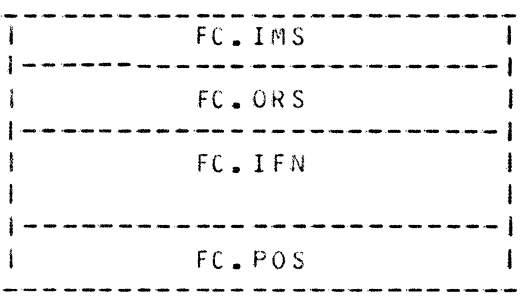
Format

```
ICLIMS    FMS,2,UPAT.F,EAX0
```

Parameters

- descriptor 0 : frames the Command Block

Command Block



Input

FC.IFN Internal name for the file
FC.POS Position
Control Interval number to be addressed by the first entry in the PAT.

Output

FC.IMS Immediate status
FC.ORS Original status

Function

The user's PAT pointers will be searched for a matching internal file name (FC.IFN). If a match exists and it describes a mass storage file, this function will retrieve the File Description and determine which Space Descriptors are necessary to map the part of the file sought. This calculation is base on the position (FC.POS) specified in the command block.

The selected Space Descriptors will be stored in the user's PAT and word 3, bits 14-35 will contain the number of the first llink described for random files or llinks between the first llink described and the sought position. In addition, word 2, bit 10 (first descriptor not in memory) will be set.

3.1.4.7 Write_File_Attributes

This function is used to write a block of attributes for the named file.

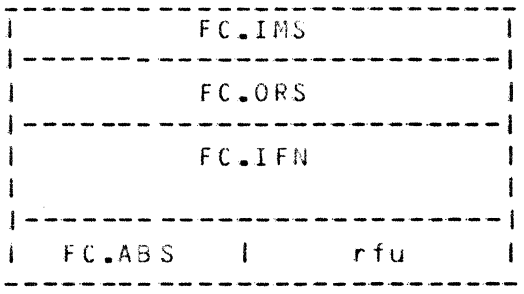
Format

```
ICL1MB    FMS,2,FWFA.F,EAX0
```

Parameters

- descriptor 0 : frames the Command Block
- descriptor 1 : frames the attribute block

Command Block



Input

FC.IFN Internal name for the file
FC.ABS Attribute block size

Output

FC.IMS Immediate status
FC.ORS Original status

Status Codes

24

Function

The user's PAT pointers will be searched for a matching internal file name (FC.IFN). If a match exists, the File Description for the file will be retrieved and the file's attributes moved from the attribute block to the File Description and it will be written to the catalog structure.

3.1.4.8 Mark_Space_Defective

This function is used to withdraw the defective mass storage space from subsequent assignment and to mark the mapping information in the PAT and cataloged File Description to show that the file space is defective. The function is called by the Exception Processing Subsystem when after failing to recover from a check character data alert, the operator responds with a withdraw (W) option.

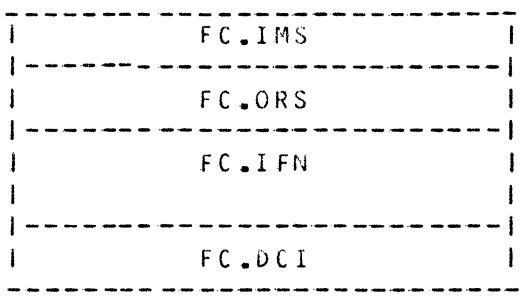
Format

```
ICLIMB   FMS,1,FMSD.F,EAX0
```

Parameters

- descriptor 0 : frames the Command Block

Command Block



Input

FC.IFN Internal name for the file
FC.DCI Defective Control Interval

Output

FC.IMS Immediate status
FC.ORS Original status

Status Codes

24

Function

Description to be supplied

3.1.4.9 Mark_Space_Usable

This function is used by a process to remove the defective space indication from the specified space descriptors. In order to use this function the process must have recovery permission for the file.

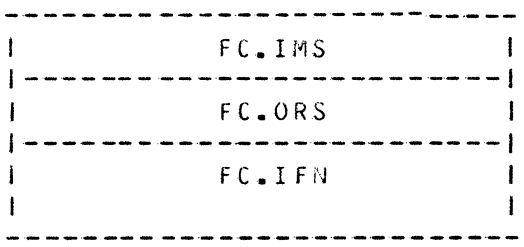
Format

```
ICLIMB   FMS,2,FMSU.F,EAXO
```

Parameters

- descriptor 0 : frames the Command Block
- descriptor 1 : frames a list of control intervals

Command Block



Input

FC.IFN Internal name of the file

Output

FC.IMS Immediate status
FC.ORS Original status

Space Descriptor List

Status Codes

24

Function

Description to be supplied.

3.1.4.10 Mark_Data_Usable

This function is used by a process to remove the defective data indication from the specified space descriptors. In order to use this function the process must have recovery permission for the file.

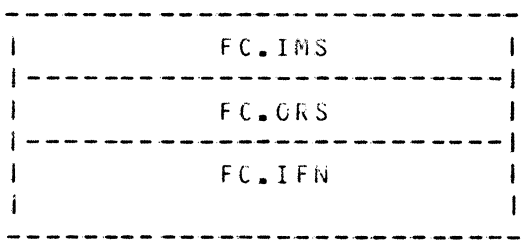
Format

```
ICLIMB   FMS,2,FMDU.F,EAXO
```

Parameters

- descriptor 0 : frames the Command Block
- descriptor 1 : frames a list of control intervals

Command Block



Input

FC.IFN Internal name of the file

Output

FC.IMS Immediate status
FC.ORS Original status

Space Descriptor List

Status Codes

24

Function

Description to be supplied.

3.1.4.11 Identify_Defective_Space

The identify defective space function is used by a process to obtain a list of spaces on a file that are marked defective in the cataloged file description. It also provides information to assist in the recovery of data in the defective space. This information includes date and time of last change to the file content, volume serial number of the volume with the most recent file save, location of any file protection data such as journaled afters or a duplicate file copy.

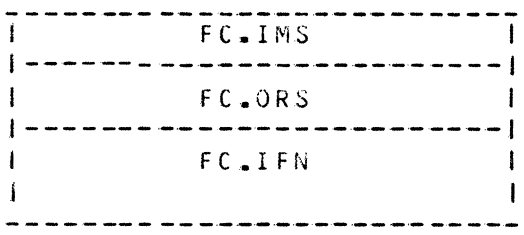
Format

```
ICLIMB   FMS,2,FIDS.F,EAX0
```

Parameters

- descriptor 0 : frames the Command Block
- descriptor 1 : frames space for a list of defective control intervals

Command Block



Input

FC.IFN Internal name for the file

Output

FC.IMS Immediate status
FC.ORS Original status

Defective Control Interval List

Status Codes

24

Function

Description TBS

3.1.4.12 Replace_Defective_Space

This function is used by a process to replace mass storage device space(s) that was previously marked defective. It replaces the defective space with newly assigned space and then modifies the mapping information in the PAT and in the cataloged file description to reflect the replacement space. In order to call this function the process must have recovery permission on the file.

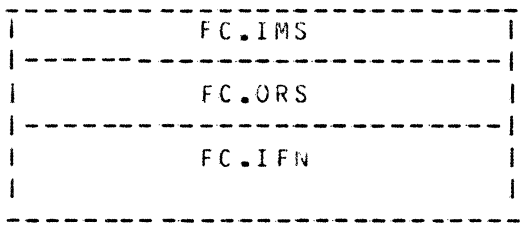
Format

```
ICLIMB    FMS,2,FRDS.F,EAX0
```

Parameters

- descriptor 0 : frames the Command Block
- descriptor 1 : frames a list of defective contron intervals

Command Block



Input

FC.IFN Internal name for the file

Output

FC.IMS Immediate status
FC.ORS Original status

Defective Control Interval List

Status Codes

24

Function

This function will use the list of defective space descriptors to search the file description for the defective descriptor. The descriptor will be replaced by a new descriptor in the File Description and also in the PAT if the file is allocated.

After the descriptor has been replaced, the content will be marked defective if the Mark Data Defective option is used.

3.1.4.13 Sequential File Position

This function will return the current file position in a sequential file. It should only be used when the first descriptor for the file is not in the PAT.

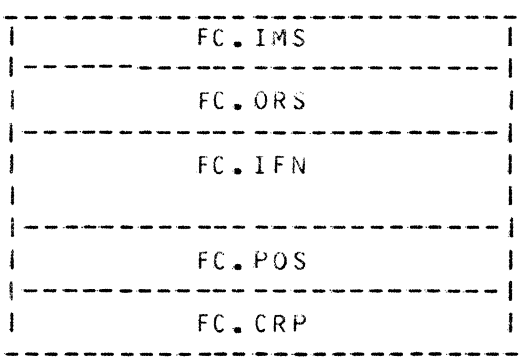
Format

```
ICLIMS    FMS,2,FSFP.F,EAX0
```

Parameters

- descriptor 0 : frames the Command Block

Command Block



Input

FC.IFN Internal name for the file
FC.POS Position value (must be non zero)

Output

FC.IMS Immediate status
FC.ORS Original status
FC.CRP Current relative position in file

Status Codes

24

Function

description to be supplied

3.1.5 Status Codes and Diagnostic Messages

Status codes are set in the Command Block by each module detecting the condition.

The following table lists the status codes and their corresponding diagnostic message.

Table 1. Status Codes and Diagnostic messages

Status Code (Octal)	Diagnostic Messages
01	Account not registered
03	Incorrect access permission
04	File busy; try again
05	Incorrect pathname
10	No space for file on Volume Set YYY...YYY
11	Nonunique name
14	Incorrect or missing password
15	File is abort locked
27	File has defective space
28	File has defective data
31	Access granted to an I-D-S 1 file
44	Illegal option specified
53	Read access denied (null file)
71	Incorrect Control Interval specified
75	File is restore locked

.spb