COPY 5 OF 5

```
**********************************************************************************************
**********************************************************************************************
**
**
**
**      JJJ    III    M    M   M    M   Y    Y
**       J      I     MM  MM   MM  MM   Y    Y
**       J      I     M M M   M M M M    Y Y
**       J      I     M M M   M M M M     Y
**    J  J      I     M   M   M   M M     Y
**    J  J      I     M   M   M   M M     Y
**     JJ      III    M   M   M   M M     Y
**
**
**
**    RRRR    EEEEE  V    V     1      999             III    N    N  FFFFF    000
**    R   R   E      V    V    11     9   9      _      I     NN   N  F       0   0
**    R   R   E      V    V     1     9   9     __      I     N N  N  F       0   0
**    RRRR    EEEE   V  V       1      9999   _____     I     N N  N  FFFF    0   0
**    R R     E      V V        1         9     __      I     N N  N  F       0   0
**    R  R    E      VV         1        9       _      I     N  NN  F       0   0
**    R   R   EEEEE  V        111    999             III    N   N   F        000
**
**
**
**********************************************************************************************
**********************************************************************************************
```

Subject : Applications Libraries, V-mode and R-mode

Release : Rev. 19.0

Date : December 12, 1981

1. New Functionality

none

2. Problems Fixed

A. The following Polers have been fixed :

29597 a missing key, A$FLOW, has been inserted into SYSCOM>A$KEYS.INS.PL1

33282 UNIT$A now returns FALSE on a bad unit number

80457 In spite of the description in the Subroutines Manual, EDAT$A will return a European date with periods as separators. and not with slashes. This aspect appears in the documentation contained within the source file for this routine.

ABSTRACT

Rev 19 Avail reports in physical records (1040 words). An option, '-NORM', is available to obtain reports in normalized (440 word) records.

Rev 19 Avail reports in physical records (1040 words). An option, '-NORM', is available to obtain reports in normalized (440 word) records. Examples of use are:

```
AVAIL
AVAIL  -NORM
AVAIL   NAME
AVAIL   NAME -NORM
AVAIL   *
AVAIL   * -NORM
AVAIL   -LDEV nn
AVAIL   -LDEV nn -NORM
AVAIL   ZERO
        :
AVAIL   SEVENTEEN
AVAIL   ZERO -NORM
        :
AVAIL   SEVENTEEN -NORM
```

Bugs_fixed

AVAIL * no longer prints garbage if there is no comments field in SYSTEM>DISCS. [POLER 45591]

* NOTE: All products have been modified to conform to master disk standards. For a description of these modifications, please read INF019>STANDARDS.RUNO.

ABSTRACT

This document describes the changes to Batch for revision 19.0 of the master disk, from the 18.2 version.

The changes are in some ways extensive; most of them, however, do not directly affect the end user of Batch. Changes are mostly internal to Batch, and also to the operator/administrator interface to Batch (the BATCH program). Also included in this release is group name and project name support for the ACL/User Profile environment of PRIMOS rev 19.0. In addition, Batch is now secured against users being able to spawn jobs as other users, or being able to change job information on jobs which don't belong to them.

This version of Batch will not run on pre-rev 19.0 PRIMOS. Nor will a pre-rev 19.0 Batch run on a rev 19.0 PRIMOS. Also, this version requires re-initialization of the entire batch database due to changes in the format of the database.

1 CHANGES TO THE FILE AND UFD STRUCTURE OF BATCH

The BATCH, BATCHQ and CMDNCO ufds on the master disk (and as built and initialized by Batch itself) are all changed at rev 19.0.

1.1 Changes to the BATCH directory

In BATCH, all files are now in compliance with the file naming standard. Also, the file "MONITR" has been changed to "MONITOR.FTN". Insert files ("B$QCOM", "B$COMN", etc.) now have the suffix ".INS.FTN".

Seven new insert files exist: "PARSE.INS.PMA", "PARSE_VECTOR.INS.PMA", "B$SWIN.INS.FTN", "B$SWIN.INS.PL1", "B$COMN.INS.PL1", "B$LOGO.INS.FTN", and "B$LOCK.INS.FTN".

The build files "C_BATCH" and "C_LIST" are now known as "BATCH.BUILD.CPL" and "BATCH.LIST.CPL".

The "INIT.FTN" module has been replaced with an "INIT.SPL" module. There are three other new modules for the INIT program named "PARSE$.PMA", "SNUM.SPL" and "INIT_SUBS.FTN". The "PARSE$.PMA" uses the two new insert files "PARSE.INS.PMA" and "PARSE_VECTOR.INS.PMA" mentioned above.

1.2 Changes to the BATCHQ directory

The REV. 19 BATCH subsystem is designed to take advantage of ACL protection. It grants particular rights to privileged users, BATCH_SERVICE (the BATCH monitor), SYSTEM, and BATCH administrators, and grants varying, lesser rights to other users. This allows them to use the system, but not to access each other's jobs or to access BATCH files and directories.

If the MFD in which BATCHQ resides is a password directory, BATCHQ itself must be a password directory. However, neither BATCHQ nor its subdirectory Q.CTRL may have passwords. If these directories are given passwords, the BATCH subsystem becomes inoperable. Therefore, no security exists on a system on which BATCH is running and in which BATCHQ is a password directory.

All of the "*prog" static-mode image files ("*MONITR", "*FIXBAT", "*INIT") have been changed to "prog.SAVE". The program names are as follows: "MONITOR.SAVE", "INIT.SAVE", "FIXBAT.SAVE".

The command file "PH_GO" has been changed to "START_BATCH_MONITOR.COMI". The command files "C_RSET" and "C_BDIF" no longer exist; their functionality has been replaced entirely by the new INIT program.

A new subufd in BATCHQ, named "OTHER", will now be generated by the INIT program in addition to "Q.CTRL" and "CIFILE". This subufd holds the files "IN.USE" and "VALID.", which used to reside in BATCHQ. OTHER is initialized as a password directory, but may be converted to an ACL directory if desired; however, all Batch users must have "ALL" access to BATCHQ>OTHER. Because of that restriction, it is probably more secure as a password directory. Its password is set to the same password as BATCHQ>CIFILE has.

The CIFILE subufd in BATCHQ must be a password directory at rev 19.0. It will also have a default password of "OSIRAS" as released on the master disk. Since Batch software is solely responsible for creating, deleting and using BATCHQ>CIFILE and BATCHQ>OTHER (which has the same password) via the new INIT program, the administrator need not know the password. However, he may change it by editing Batch source code and rebuilding Batch. See the section on the new INIT program.

The INIT program, when run, will automatically delete all old (pre-rev 19.0) files (*MONITR, C_RSET, etc.).

Not all files are in compliance with the file naming standard in BATCHQ - examples are "Q.CTRL", "CIFILE", "SEMFILE", "LOCK.B". These are run-time generated files and ufds.

1.3 Changes to the CMDNC0 directory

In CMDNC0, the static-mode image run files are now named "JOB.SAVE", "BATCH.SAVE", "BATGEN.SAVE", and "$$.SAVE". The files "JOB", "BATCH", "BATGEN" and "$$" should be deleted.

2 INSTALLING BATCH

The build file for Batch automatically installs everything in the right places (BATCHQ and CMDNC0). The BATCHQ ufd must contain the "START_BATCH_MONITOR.COMI" file, which is present on the U1 partition in BATCHQ, as the Batch build file will not create this.

Once everything is installed, the Batch database must be initialized. However, this is only possible if the disk on which BATCHQ resides has been converted to a rev 19 partition by FIX_DISK, and if the MFD has been converted to an ACL ufd. When this is done, the command:

```
R BATCHQ>INIT -RSTQ
```

will initialize Batch. Then, BATGEN should be run to set up the queues for that system. The Batch subsystem now supports 16 queues instead of 6. Once the queues are defined, the Batch subsystem is ready to run. Now the startup sequence for Batch in C_PRMO must be changed.

In C_PRMO, the sequence:

```
PH BATCHQ>PH_GO
CHAP -nn rlv ts
```

is obsolete and no longer supported. At rev 19.0, Batch has a new mechanism for starting up the monitor. With the old mechanism, the above sequence would be performed, and then when the monitor sent the message "Waiting for BATCH SYSTEM -START" to the system console, the command:

```
BATCH SYSTEM -START
```

would be issued. This is all obsolete. The new sequence in C_PRMO should be as follows:

```
BATCH -START -RLV rlv -TS ts
```

where <rlv> and <ts> are decimal numbers, not octal numbers as they were in the CHAP command. This command replaces the old sequence, and no longer requires the "BATCH SYSTEM -START" command to be issued later on after the monitor sent a message. Also, this command can be executed before the system date and time are set, if desired.

The Batch monitor is no longer spawned as user "SYSTEM". It is spawned as user "BATCH_SERVICE". Whether or not this username is in the User Validation File (UVF) is unimportant at this revision; however, the Batch subsystem now recognizes as privileged users both "SYSTEM" and "BATCH_SERVICE". These users can display and modify all user's jobs in the Batch subsystem.

Because the username of Batch is now "BATCH_SERVICE", messages sent by that user to the console will no longer have the text "*BATCH* " in front of them. The exceptions are when it is not clear that the message is coming from the Batch monitor, as is the case with database invalidation messages. Also, "Executing xxx for user yyy" messages have "*BATCH* " in front of them still, because they are sent by the user process spawned to run the batch job.

When the monitor is in operation, the message "Monitor in operation" will be sent to the console. However, no action need be taken at this point - the message is simply a notification that the monitor is finished fixing the database (by running FIXBAT) and is ready to process Batch jobs.

The BATCH command has been essentially rewritten at rev 19.0. It will be discussed in further detail later in this document.

As a result of the startup sequence change, the message "Warning: the batch monitor is still awaiting start-up instructions from the operator, so jobs are not yet being processed" is no longer necessary and does not exist.

3 ADMINISTRATIVE CHANGES

3.1 Batch Administrator

The new INIT program allows the specification of the Batch Administrator when run. If not specified, it uses the current user (see the section on the new INIT program for details).

It uses this information to set up the access on the Batch database. A "Batch administrator" is defined as any user with "ALL" access to BATCHQ. Therefore, since users SYSTEM and BATCH_SERVICE are given "ALL" access to BATCHQ, they are automatically Batch administrators.

INIT.SPL has been changed so that the BATCH administrator is granted full access rights to the BATCH subdirectories in additon to the BATCHQ ufd before attempting to delete the subdirectories. This is necessary if any of the subdirectories contain any BATCH files, e.g., queue control files or command files.

A Batch administrator can run BATCHQ>FIXBAT and BATCHQ>INIT. He can also do the BATCH commands -STOP, -PAUSE and -CONTINUE. However, only users logged in as SYSTEM or BATCH_SERVICE can manipulate other users jobs. Only SYSTEM or BATCH_SERVICE can use the -HOLD and -RELEASE options on the JOB command. And only the system console can do BATCH -START and successfully abort other users' running jobs. That is, only the system console may abort a running job.

The Batch administrator is also given full access to BATCHQ>BATDEF, so he can change the queue configuration anytime. All other users are given "R" access (read-only). "R" access is necessary for all users who submit Batch jobs, as the JOB program needs to have access to BATCHQ>BATDEF.

3.2 Copying Into BATCHQ>BATDEF

As of rev 19.0, only BATGEN should be used to copy a new queue configuration into BATCHQ>BATDEF. Do not use FUTIL and never use COPY. This is because BATGEN does not delete and re-create BATCHQ>BATDEF when it copies into it. It only overwrites the data.

Therefore, it leaves the specific ACL which the INIT program places on BATCHQ>BATDEF in place. If this is disturbed, users will be unable to submit jobs or do just about anything with Batch. The error messages will probably be "Insufficient access rights. BATDEF missing".

Also, by using BATGEN, the Batch monitor is immediately notified, causing potentially faster turnaround on queue deletions and such.

3.3 Database Error Logging

The format of "BATCHQ>ERROR." is changed. Entries are now appended to the end of the file. Previously, they overwrote whatever entry was already there.

Also, the format of an entry is itself changed. It now occupies only one line in the file, not two. In addition to the information describing what part of Batch received the error and what the error code was, the username and usernumber (in parentheses) is also logged.

For example. the entry:

```
   JOHNSON (12) JOB(1004) error=1
```

Means that user JOHNSON (user number 12) got a fatal database error "End of file" (error code 1) in the JOB program at statement number 1004.

4 OTHER GENERAL CHANGES

4.1 Batch Now in V-Mode

All of the Batch subsystem now executes in V-mode. All programs now do a "RETURN" to exit to Primos (except when errors occur, or when the "QUIT" command is given in BATGEN command mode), so a subsequent "START" command will produce a "Program halt at xxx" message. The old error messages "No restarts allowed" are gone.

4.2 Maximum Quota Exceeded in BATCHQ

Batch now handles the new "Maximum quota exceeded" error at rev 19.0 the way it handles "Disk full" errors. This will be expounded on a bit more in the section which describes changes to the monitor.

4.3 Long Usernames - 32 Characters

This revision of PRIMOS supports 32-character usernames, and Batch has been modified to support it also.

In the "BATCH -DISPLAY" tabular report, the "User" field will be dynamically adjusted to reflect the longest username in the table (minimum of 8 characters).

In the "JOB -STATUS" tabular report, usernames are assumed to be 8 characters or less in length. If one is more than 8 characters long, it will be printed on the line by itself, and the job information will be printed on the next line. Multiple occurances of that long username that are contiguous will be optimized so that the username is not output repeatedly.

4.4 Software Interrupt Functionality

The software interrupt functionality (SW$INT) has been integrated into REV19.0 BATCH. All references to BREAK$ to inhibit or enable QUITS have been replaced with calls to SW$INT to effectively create critical sections.

4.5 Solution To The BATCH Database Deadlock

All attempts to attain the BATCH subsystem database lock are now singularly threaded through a named semaphore in order to assure prompt service to those processes waiting to obtain the subsystem. This change along with a change to JOB$0.PLP which closes the queue file unit before exiting with a file-in-use error is meant to address the BATCH deadlock situation resulting when multiple, simultaneous accesses to the same BATCH queue file occur. The file "SEMFILE" in the BATCHQ ufd is associated with the database lock named semaphore.

All BATCH subsystem commands and programs now contain onunits to handle the QUIT$ condition resulting from a terminal quit. This onunt simply closes the named semaphore if the user had opened it.

4.6 Batch Monitor Semaphore-Driven

The Batch monitor is now semaphore-driven. Semaphore number -15 is notified by Batch programs when they want the monitor to wake up. In a dormant system, the monitor will automatically wake itself up every 10 minutes.

By making the monitor semaphore-driven, turnaround time for job startup and the detection of aborted jobs should be smaller, and the cost of the Batch monitor on a dormant system should be greatly reduced.

4.7 Database Timeouts

The timeout on files that a Batch program is trying to open is now 60 seconds instead of 30.

4.8 Database Locking Mechanism Change

The Batch subsystem makes use of an 8-word area from 6001/67770 through 6001/67777 inclusive. This area is to control access to the database lock over multiple program invocations (such as a long chain of JOB commands).

4.9 Message State No Longer Reset

The message "(Batch) I have reset your message state to -ACCEPT" is now gone. When Batch needs to send a message to the system console, and the sending user has an overly inhibitive message state, Batch will reset it, then send the message, and then set the state back to what it was.

4.10 System Date and Time Errors

The error message "System time must be set first. (INIT$B)" is now "Date and time not set. (Batch)", and is only issued for the BATGEN and JOB commands, and the INIT program in BATCHQ. The BATCH command can now be run even if the date and time are not set, and the FIXBAT and MONITOR programs in BATCHQ will, as usual, just wait for the date and time to be set before they start running.

4.11 Fixed POLERS

POLER #37550 has been fixed. This was a bug which prevented execution of jobs from a queue if a previously-defined queue contained nothing but held jobs.

4.12 Temporary Files Used For Job Initiation

Temporary files generated by the Batch monitor through which Batch jobs are spawned will be given "$REST:R" access by that monitor so that the spawned job may open the file for reading. This means that the spawned job may not change the access on the unit on which that temporary file is open to reading & writing (or writing only) via the K$CACC key to SRCH$$.

4.13 Error Messages

Most error messages which output the names of files in BATCHQ or its subufds now put "BATCHQ>" in front of those names. For example, the message that the monitor sends when one of its files gets closed is now:

    BATCHQ>OTHER>IN.USE not open.

4.14 Subsystem-Wide Abbreviations

The Batch subsystem now has abbreviations for most of the options, commands and subcommands (in BATGEN). These abbreviations will be listed for each command described below.

Also, the corresponding "Usage:" texts have been modified to have the options be in upper/lower case, where the upper-case characters define the abbreviations for the option.

5 CHANGES TO THE BATGEN PROGRAM

5.1 Error Handling

When BATGEN reports an error while in command or subcommand mode, it will now call the subroutine SS$ERR, which will cause the "ER!" prompt to be generated (and command input will be paused) if input is coming from a command file. Otherwise, it will do nothing.

See the description of this routine in the subroutine guide or the Primos manual for more detailed information.

5.2 Initial Values for CPU/Elapsed Time Limits

The initial values for CPTIME and ETIME, when adding a queue, are now "None" for both default and maximum parameters. "None" also means "Infinite".

5.3 Abbreviations

Abbreviations for the BATGEN program are as follows:

-STATUS = -ST, -DISPLAY = -DP, FILE = FI or FIL, BLOCK = BLK, UNBLOCK = UNBLK, DELETE = DL, MODIFY = MOD, DISPLAY = DP, STATUS = ST, CPTIME = CPT, ETIME = ET, FUNIT = FU, PRIORITY = PRI, QUIT = Q QU or QUI, TIMESLICE = TS, RLEVEL = RLV, and RETURN = RTN.

6 CHANGES TO THE INIT PROGRAM

|At rev 19.0, the INIT program is rewritten, and now does all of the |functions that were previously performed by the "C_RSET" and "C_BDIF" |files in BATCHQ. Those files are no longer in existence.

| 6.1 Usage

| To invoke INIT, do:

| R BATCHQ>INIT [-RESET_QUEUES] [-ADMINISTRATOR user]

| The "-RESET_QUEUES" option may be abbreviated to "-RSTQ", and the | "-ADMINISTRATOR" option may be abbreviated to "-ADMIN".

| "-ADMINISTRATOR" specifies the Batch administrator for this system. | The <user> parameter should be the name of a user who can login to | the system. If the "-ADMINISTRATOR" option is not present, the | current logged-in user is assumed to be the administrator.

The Batch administrator is given "ALL" access to the BATCHQ ufd and all sub-ufds and files. Users "SYSTEM" and "BATCH_SERVICE", also "Batch administrators", are also given "ALL" access to BATCHQ.

If "-RESET_QUEUES" is specified, a new BATDEF file will be created in BATCHQ with no defined queues. If it is not specified, the old one will be left as is. Since the rev 19.0 BATDEF and previous versions of BATDEF are incompatible, this option must be used when the new system is first initialized.

Note: the INIT program does not actively prevent the "wrong" users from running it. But since only Batch administrators will have access to the program, and only they will have sufficient access to BATCHQ to actually initialize anything, there should be no problems with misuse of INIT.

6.2 What INIT Does

The INIT program first obtains the database lock, to wait for other users who might be in the middle of updating the Batch database (which will be wiped out anyway).

It then opens the file "BATCHQ>INIT.BATCH" for reading & writing. This file is held open for the duration of initialization. It is also held open by the Batch monitor while it is running. This way, it is guaranteed that INIT will not be able to run while the monitor is running, and that the monitor cannot be started up while INIT is running.

It then deletes all old (pre-rev 19.0) files which may be hanging around in BATCHQ. These files are: "*INIT", "*FIXBAT", "*MONITR", "C_RSET", "C_BDIF", "PH_GO", "IN.USE", and "VALID.". Also deleted are "INITIALIZE_DATABASE.COMI" and "RESET_DATABASE.COMI", which were present in the pre-release version of rev 19.0 Batch.

The BATCHQ subufds "Q.CTRL", "CIFILE" and "OTHER" are then deleted. Then, the BATCHQ files "MON.SR", "MON.PA", "MON.ST", "ERROR.", "EXECUT", "QUEUE" are all deleted.

Then, if "-RESET_QUEUES" was specified, "BATCHQ>BATDEF" is deleted.

Now, it is time to rebuild the database. The "EXECUT" and "QUEUE" files are built and written in BATCHQ. If "-RESET_QUEUES" was specified, a null queue configuration is written out in "BATCHQ>BATDEF".

Then, the database is unlocked so that the database lock file, "LOCK.B", may be deleted. After it is deleted, the sub-ufds are re-created.

The "Q.CTRL" sub-ufd is created as an ACL ufd.

The "CIFILE" and "OTHER" sub-ufds are created as password directories, with the current Batch password as the owner password, and six blanks as the non-owner password.

Then, the files "MON.ST", "LOCK.B", and "ERROR." are re-created in BATCHQ.

Now, INIT sets up the access on objects inside BATCHQ.

On "Q.CTRL", it gives Batch administrators (including, of course, users SYSTEM and BATCH_SERVICE) "ALL" access; it gives users in group ".BATCH$" "ALL" access; and gives all other users ($REST) "LUR" access. The group ".BATCH$" is reserved for Batch and PRIMOS use only. It is part of the security mechanism which prevents user A from changing the attributes of a job belonging to user B. It also prevents user A from causing his job to run while logged in as user B.

Then, an access category named "RW.ACAT" is created. It gives "ALL" access to Batch administrators, and "RW" access to $REST users. The files "LOCK.B", "EXECUT", "QUEUE", and "ERROR." are put in access category "RW.ACAT". This is because all users need to be able to read and write those files to run the JOB command.

The "BATDEF" file is then given a specific acl, in which Batch administrators have "ALL" access, and all other users have "R" (read) access.

The access is now all in place. The database is initialized. INIT signals this by re-creating the file "BATCHQ>OTHER>VALID.". It then closes "INIT.BATCH" and returns the user to PRIMOS.

6.3 Changing the Batch Password

As released, the password on the CIFILE and OTHER sub-ufds in BATCHQ is "OSIRAS". Osiras is the God of the Netherworld, and the Judge of the Dead, from Greek mythology. He is also the brother and husband of Isis, the Goddess of Fertility.

However, if it is desired to change the password, that is done as follows:

```
OK, ATTACH_BATCH  /* Attach to source ufd.
OK, ED_B$LIBF.FTN  /* Edit the fortran library for Batch.
EDIT
LOCATE_---_BATCH_PASSWORD_---  /* Find the password.
      CALL MOVE('OSIRAS',SUBPAS(1,2),3)  /* --- BATCH PASSWORD ---.
MODIFY_/OSIRAS/newpas/  /* Change the password.
      CALL MOVE('newpas',SUBPAS(1,2),3)  /* --- BATCH PASSWORD ---.
FILE

OK, R_BATCH.BUILD  /* Rebuild Batch.
```

It would be wise to set the access on the Batch ufd so that no unauthorized users could read files in it, now that the password has been changed.

The password ("newpas" in the example) must be 6 characters in length or less.

7 CHANGES TO THE FIXBAT PROGRAM

7.1 Check For Password Ufd

The BATCHQ>CIFILE subufd cannot be an ACL ufd. The INIT program explicitly creates CIFILE as a password ufd. However, FIXBAT also checks CIFILE whenever it fixes the database to make sure it is not an ACL ufd. If it is, the following error message will be displayed:

```
BATCHQ>CIFILE cannot be an ACL ufd.  Do R BATCHQ>INIT (FIXBAT)
```

7.2 Unauthorized Usage

FIXBAT no longer prevents unauthorized usage explicitly; the "No right. Must be logged in as SYSTEM" message no longer exists.

Instead, the access on BATCHQ (including FIXBAT.SAVE) prevents unauthorized use automatically.

7.3 Quits Enabled

FIXBAT now correctly enables quits when -STARTUP is specified. Previously, this caused the Batch monitor to run with quits inhibited, which prevented it from receiving asynchronous signals from the condition mechanism, such as "PH_LOGO$". Now, the monitor runs with quits enabled.

| The "VALID." file (in BATCHQ>OTHER) is now opened before command output is turned on (if logging is specified), so multiple monitors spawned at once will correctly output the message "Multiple monitors illegal" rather than an obscure "O_LOG" error.

8 CHANGES TO THE JOB PROGRAM

8.1 User Profile Support

Group and project names are assigned to submitted Batch jobs based on the group and project names of the submitter at submit time, i.e. when JOB is invoked.

8.2 Change to Display of Job Information

The "JOB -DISPLAY" option now displays the home ufd of the job.

8.3 Change to Warning Message

The message "Your job, #qnnnn, could be in queue <queue>, but may not execute due to the afore-mentioned error" is changed so that the word "afore-mentioned" is replaced by "following". The error message is now printed after this message, and causes an "ER!" prompt (which suspends command input), rather than before the message and raising no error indication.

8.4 Change to Unauthorized Use Message

The one remaining "No right. Must be logged in as SYSTEM" message in the entire Batch subsystem is output when a user, who is not logged in as SYSTEM or BATCH_SERVICE, attempts a -HOLD or -RELEASE operation via the JOB command.

Therefore, the message has been changed to read "No right. Must be logged in as SYSTEM or BATCH_SERVICE".

8.5 Change to Submission Error Message

The error message "Command file required as first argument on submission" now reads "Command or CPL file required as first argument on submission".

8.6 Abbreviations

Abbreviations for JOB options are as follows:

-CANCEL = -CAN, -CHANGE = -CHG, -ABORT = -AB, -RESTART = -RST, -STATUS = -ST, -DISPLAY = -DP, -ACCOUNTING = -ACCT, -HOME = -HO, -CPTIME = -CPT, -ETIME = -ET, -PRIORITY = -PRI, -QUEUE = -QUE, and -FUNIT = -FU.

Notice that the -ACCT option is now considered to be an abbreviation of the -ACCOUNTING option.

9 CHANGES TO THE BATCH PROGRAM

The BATCH program has been rewritten. The syntax "BATCH SYSTEM -option" is now obsolete and, if used, will produce an error message.

9.1 Usage

The BATCH command is now invoked as follows:

For all users:

```
BATCH -DISPLAY
      -STATUS
```

For Batch administrators:

```
BATCH -STOP
      -PAUSE
      -CONTINUE
```

For the system console:

```
BATCH -START [-RLEVEL x] [-TIMESLICE y]
```

9.2 Changes to Display Batch Status

BATCH -DISPLAY works as it did before. It has a minor change; when the total number of waiting and held jobs is printed, the number of queues that had waiting and held job is also displayed ("(r queues)").

9.3 The New Short-Status Command

BATCH -STATUS prints a one-line description of the status of the Batch subsystem. The information on this line describes the number of waiting and held jobs, the number of queues that have waiting and held jobs, and the number of executing jobs. If there are both waiting and held jobs and executing jobs, the total number of active batch jobs will also be printed. If there are no active batch jobs, "No batch jobs" will be displayed.

9.4 Unauthorized Use

As with FIXBAT, the BATCH program no longer explicitly prevents unauthorized requests to stop, pause or continue the Batch monitor. The message "No right. Must be logged in as SYSTEM" is gone.

Instead, the security is now present in the way the INIT program sets access on BATCHQ. To stop, pause or continue the Batch monitor, a user must have "ALL" access to BATCHQ. Only Batch administrators have this access.

9.5 The Stop Request

BATCH -STOP stops the Batch monitor. The monitor, when it sees this request, will send the message "Operator stop" and will logout. (Note: this message no longer rings the bell).

9.6 The Pause and Continue Functions

BATCH -PAUSE and BATCH -CONTINUE work exactly as they did when they were BATCH SYSTEM -PAUSE and BATCH SYSTEM -CONTINUE.

9.7 Starting Up the Batch Monitor

BATCH -START starts up the Batch monitor. If specified, -RLEVEL and -TIMESLICE specify the CHAP command parameters to be given to the Batch monitor. The defaults are 1 for -RLEVEL, and 20 for -TIMESLICE. The arguments to these options are decimal. The -RLEVEL value must be between 0 and 3, and the -TIMESLICE value must be between 1 and %9. The options "-START", "-RLEVEL" and "-TIMESLICE" may be specified in any order.

9.8 Abbreviations

Abbreviations for options of the BATCH command:

-DISPLAY = -DP, -STATUS = -ST, -RLEVEL = -RLV, and -TIMESLICE = -TS.

10 CHANGES TO THE BATCH MONITOR PROGRAM

10.1 Forced Logout Now Handled

The Batch monitor now handles the new rev 19.0 "LOGOUT$" condition. As a result, force logging out the monitor once will not immediately log it out, but as soon as the monitor gets a chance, it will send the message "Force logout by operator" and log itself out. If this message is sent, then the database is completely intact.

If a forced logout does not immediately cause this message to be sent, then a second force logout of the monitor will cause it to immediately log itself out, which will leave the database in an unknown state. The Batch database will then be unusable until FIXBAT or INIT is run.

10.2 Phantom Logout Info in Log File

When Batch jobs log out, the message "At hh:mm:ss; Phantom nn (ab)normal logout." is entered into the Batch monitor log file (unless those jobs were not spawned by the current monitor process). These messages are asynchronous to any activities the Batch monitor may be performing, so the corresponding "++Finished:id" and "Job xx aborted/completed" messages may not directly follow it in the log file.

It is even possible (with short jobs) for an "Executing xx" message to appear after its corresponding "At hh:mm:ss; Phantom nn logout" message.

10.3 Other Change to Log File

The log file "0_LOG" kept by the Batch monitor has had some format changes. The date is now output whenever it changes as follows: "Date: mm/dd/yy". The time of day is now only output when it is different from the time of day when the last line was output.

10.4 Monitor Should Not Be Restarted

The Batch monitor should not be shut down and started up again while Batch jobs are executing. If it is, the Batch monitor will not have fast turnaround on those jobs terminating. In other words, a job may complete, but the Batch monitor may not recognize that it completed for up to 10 minutes.

This is because the Batch job notifies its spawning process when it terminates, but if the spawning process has logged out, it does no such notify. The Batch monitor, which now sleeps up to 10 minutes at a time between checks, is driven to check for job abortions by those notifies.

10.5 Maximum Quota Exceeded is Handled

When the maximum quota for BATCHQ is exceeded by the Batch monitor (usually when it tries to spawn a job), it will send the message "My quota is exceeded. Please help me." to the system console. This message parallels the "My disk is full. Please help me." that the monitor sends when it gets a disk full error.

In both cases, the Batch monitor will not attempt to spawn another job for at least 5 minutes. It may even wait longer if it is not explicitly woken up (notified). A quick way to notify the monitor is to do "BATGEN BATCHQ>BATDEF" followed by "FILE".

Table of Contents

BUG FIXED:
- CMPF will now close files by unit number instead of by file name.

There are no user visible changes.

The files necessary to build the CONCAT utility have been updated to conform to rev 19 standards.

* NOTE: All products have been modified to conform to master disk standards. For a description of these modifications, please read INF019>STANDARDS.RUNO.

Subject: CONVERT_PROFILE

Release: 19.0

Date: January 21, 1982

New Functionality

CONVERT_PROFILE is a one-time tool which aids in the setup of a new databases required at PRIMOS revision 19 for User Profiles.

A complete description of User Profiles and CONVERT_PROFILE may be found in the rev document.

Problems Fixed

N/A.

Outstanding Problems

N/A.

Environment

CONVERT_PROFILE is designed for use on rev 19.0 or later systems only.

Build and Install Procedure

Standard.

Subject: COPY

Release: 1.02

Date: 8/19/82

1 New Functionality

New command.

2 Problems Fixed

None.

3 Outstanding Problems

User gets lots of questions on same object. Multiple and extra error messages. Like "File delete protected" when user answered no to force question. No way to do a directory merge operation (UFDCPY). Segment directories are treated as a single object. This means that there is no way to move files into or out of a segment directory.

4 Environment

Needs PRIMOS 19.0.65 or greater. (19.0.respin)

5 Installation and Build Procedures

Needs fsulib.build.cpl to be run first. Other than that it is standard.

6 Usage

The usage information for the command follows.

Usage: COPY source_pathname [target_pathname] [options...]

source_pathname pathname of object to be copied.

target_pathname pathname of output object.
(If omitted <source_path> entryname is used.)

options may be selected in any order from the list below.

Option descriptions:

-QUERY, -Q
Specifies that the command is to request that the user resolve unexpected or potentially dangerous situations. This is the default.

-NO_QUERY, -NQ
Specifies that the command is NOT to request the user's permission but to attempt to resolve those situations in the most intuitive fashion.

-COPY_ALL, -CA
Specifies that COPY is preserve ALL attributes when copying an object. The default is to set the attributes to their system defaults (unless one of the following described attribute copying arguments was specified).

-DTM
Specifies that the original date/time modified is to be preserved.

-PROTECT, -PRO
Specifies that the original protection keys, passwords, and access rights of any object copied is to be preserved.

-QUOTA
Specifies that the max quota information for a directory that is copied is to be preserved.

-RWLOCK, -RWL
Specifies that the concurrency lock setting is to be preserved.
If any of the above are omitted, the default is to set the omitted attribute to its system default. When a directory tree is copied, the above rules apply to ALL subentries contained within the directory.

-LEVELS, -LV dec
Specifies that COPY is only to copy down to the level specified by "dec" when copying a directory tree. "dec" is a decimal integer from 0 to 999. If "-LEVEL" is omitted, the default is to copy the entire tree; if "dec" is omitted, the default is 0 (only copy the top level, the directory entry itself and none of its subentries).

-DELETE, -DL
Specifies that COPY is to delete an object once it has been successfully copied. The default is not to delete.

-SAM
Specifies that all FILES (i.e., sam or dam files) copied are to be converted to sam files.

-DAM
Specifies that all FILES copied are to be converted to dam files.
The default is not to perform any conversions. Note that "-sam" and "-dam" options do NOT apply to directory tree subentries.

-INCREMENTAL, -INC
Specifies that COPY is only to copy those objects whose dump bit is OFF, i.e. those files that have not been saved to tape.

-REPLACE
Specifies that COPY is only to copy those objects that exist in the source AND destination locations. The default is copy all objects selected from the source directory.

-FORCE
Specifies that COPY is to force delete rights when deleting objects. If "-FORCE" is not specified, COPY will request the user's permission to force delete an object (unless "-NO_QUERY" was specified).

-REPORT, -RPT
Specifies that COPY is report the results of each successful copy operation.

Subject: COPY_DISK

Release: 19.00

Date: August 19, 1982

1 New Functionality

A new format BADSPT file has been defined in which an individual bad record can be flagged, rather than the whole track containing the bad record. COPY_DISK has been modified to use this new format.

Badspot handling has been added to COPY_DISK so that records are not written to badspots but are mapped to the first available free record on the target partition.

In order to ensure that badspots are handled correctly, the following guidelines should be followed:
1) The Record Availability Table for each source partition should be correct. To ensure this is so, FIX_DISK can be run.
2) There should be enough free records on a source partition being copied for records falling on badspots an the target partition to be relocated to.
3) To be safe, it would be wise to keep copies elsewhere in filestore of all BADSPT files, in case of accidental loss.

2 Cosmetic Difference

Because the disk copy is not complete until after the VERIFY phase is completed as tidying up of pointers in the MFD may be required, the messages COPY STARTED, COPY COMPLETED, VERIFY STARTED and VERIFY COMPLETED have been removed as a user may be tempted to break-in during the verify phase thinking that all copying has finished.

3 Environment

To clean up a disk after badspot handling has taken place, the FIXRAT replacement FIX_DISK must be used. If FIX_DISK is not available then badspot handling must not be performed. The new command line option -NO_BADS has been provided to turn off badspot handling.

Example: COPY_DISK -NO_BADS -NOVERIFY

The default for COPY_DISK is now no verify. If the user wishes to verify he must specify the new command line option -DO_VERIFY.

4 Performance Improvement

A performance improvement of about 250% has been made for all processors below a P750 when copying partitions with 1040-word blocks. COPY_DISK assumes it is running on a P750 with burst mode disk controller. If this is not so, then to achieve the performance improvement a new option, -LOWEND, should be specified.

Example: COPY_DISK -NOVERIFY -LOWEND

WARNING: The use of the -LOWEND option with a P750 will slow down a disk copy.

5 Error message

'IF YOU DO NOT WISH TO CONTINUE WITHOUT BADSPOT
HANDLING YOU WILL NEED TO RE-MAKE PARTITION xxxxxx'

This message is sent whenever the target partition cannot accommodate the source partition (usually occurs when the source was full and the target has more badspots than the source). The message will appear in conjunction with the message -

'NO FREE RECORDS ON PARTITION xxxxxx'

Subject: CPMPC

Release: MASTER DISK RELEASE 19.0 (JUNE)
PRIMOS 19.0
CPMPC 16.2

Date May 13, 1981

1. New Functionality
MODIFIED TO CONFORM TO FILE NAME STANDARDS

2. Problems Fixed
NONE REPORTED

3. Outstanding Problems
NONE

4. Environment
CPMPC REQUIRES PRIMOS 16.2

5. Installation and Build Procedures
FOLLOW THE STANDARD PRIME BUILD PROCEDURE

Subject: CRMPC

Release: MASTER DISK RELEASE 19.0 (JUNE)
PRIMOS 19.0
CRMPC 16.2

Date May 13, 1981

1. New Functionality
MODIFIED TO CONFORM TO FILE NAMING STANDARDS

2. Problems Fixed
NONE REPORTED

3. Outstanding Problems
NONE

4. Environment
CRMPC REQUIRES PRIMOS 16.2

5. Installation and Build Procedures
FOLLOW THE STANDARD PRIME BUILD PROCEDURE

Subject: DELETE

Release: 1.01

Date: 8/19/82

1 New Functionality

The command has been extended to allow for deletion of all entries in a directory.

2 Problems Fixed

Fixed printing of names in error messages. Added checking and error messages about unknown types of file system objects. Fixed bad reporting on errors.

3 Outstanding Problems

User gets lots of questions on same object. Multiple and extra error messages. Like "File delete protected" when user answered no to force question.

4 Environment

Needs PRIMOS 19.0.65 or greater. (19.0.respin)

5 Installation and Build Procedures

A copy of delete.run must be in cmdnc0 before starting master disk build. Needs fsulib.build.cpl to be run first. Other than that it is standard.

6 Usage

The usage information for the command follows.

Usage: DELETE target_tree [options]

target_tree treename of object to be deleted.

options may be selected in any order from the list below.

Option descriptions:

-QUERY, -Q
Specifies that DELETE is to request that the user resolve unexpected or potentially dangerous situations. This is the default.

-NO_QUERY, -NQ
Specifies that DELETE is NOT to request the user's permission but to attempt to resolve those situations in the most intuitive fashion.

-FORCE
Specifies that DELETE is to force-delete a delete protect object. If "-force" is not specified, then DELETE will request the user's permission to force delete a protected object (unless "-no_query" was specified).

-REPORT, -RPT
Specifies that DELETE is report the results of each successful operation.

Subject: ED

Release: 19.0

Date: December 14, 1981

New Functionality

ED has been updated to meet the revision 19.0 standards.

User Visible Bug Fixes

The bugs described by the following Polers have now been fixed.

Polers 29369,40212,82493.

These all describe a bug in which the command C/ //G, when applied to a line ending in a space, hung ED. Now fixed, it will do the change correctly.

Polers 29963.

This describes a bug in which ED will not let you FILE out to a file unless you have Delete access on the file. The problem is that, until Rev 19.0, Delete access is required in order to be able to truncate a file, and truncation is often necessary in case the file you are writing out is shorter than what's on the disk. This is now fixed to file correctly at Rev 19.0.

Polers 33128, 80479.

These both describe a bug in which MODE with an illegal parameter drops you into Box mode, in some cases. Now fixed, it will say BAD MODE.

Polers 33936, 34683, 34837.

These all describe a bug in which the sequence D,N,OOPS would result in

the deleted line being written back in OVER the line that followed it, wiping out the latter. Now fixed, it will not wipe out the following line, and in fact will function exactly as if the Next command had not been given.

Polers 37202, 45579.

These describe, rather obliquely, a bug in which positioning around in a file with lines containing more than 128 consecutive spaces would sometimes cause ED to create a new line consisting of a duplicate of the last character(s) of the previous line. Now fixed.

Polers 82205.

This describes a bug which crashes ED when the Where command is used in Box mode. This was caused by an error in the build file which started causing other problems at Rev 19.0. Now fixed (by accident).

Internal Bug Fixes

ED now compresses up to 256 consecutive spaces. The previous limit was 128.

Outstanding Problems

None.

Environment

This revision of ED should be built and run on revision 19.0 or later PRIMOS.

Build and Install Procedure

This program may be built and installed by resuming the file ED>ED.Build.CPL.

ABSTRACT

This document describes the Changes made to EDB and LIBEDB for PRIMOS release 19.0

EDB and LIBEDB for rev 19.0

1 EDB

EDB will no longer reference the obsolete routine SETSIZ during initialization. This means that EDB will no longer execute properly under PRIMOS II. Support for development work under PRIMOS II was dropped at rev 15 for most of PRIME's software. EDB does NOT support the new file suffix naming convention at 19.0. This has been OKed by SDI.

2 LIBEDB

LIBEDB has been modified to modify object text blocks up to 256 words long instead of the old 64 word limit. This is in anticipation of PRIME translators being able to put out larger sized blocks containing larger sized groups. LIBEDB does NOT support the new file suffix naming convention at 19.0. This has been OKed by SDI.

* NOTE: All products have been modified to conform to master disk standards. For a description of these modifications, please read INFO19>STANDARDS.RUNO.

Subject: EDIT_PROFILE

Release: 19.0

Date: January 21, 1982

New Functionality

EDIT_PROFILE is a new product which supports the User Profiles functionality of PRIMOS revision 19. It enables System and Project Administrators to create and manipulate the validation files which are now required before users may log into the system.

EDIT_PROFILE is described in detail in the rev document.

Problems Fixed

N/A.

Outstanding Problems

N/A.

Environment

EDIT_PROFILE runs only on rev. 19.0 or later systems.

Build and Install Procedure

Standard.

Event_log

The new command Event_log turns OS and network logging on and off. The format is;

EVENT_LOG {-net} {-on | -off}

If the -net option is present, network logging is affected, otherwise OS logging is affected. If both -on and -off are omitted, -on is assumed. Turning OS logging on causes a file to be opened in the UFD LOGREC*. The name of the file is LOG.mm/dd/yy, where mm, dd, and yy are the numbers of the date the command is issued. This file may be specified as a -input file to LOGPRT.

For network logging, the UFD used is PRIMENET*, and the file names are of a similar format.

NOTE: The administrator must create the ufd LOGREC* before turning OS logging on.

FIXRAT

1. Modified to ignore Rev. 19 ACL's and QUOTA's.

2. Modified to conform to Rev. 19 Master Disk Standards.

* NOTE: All products have been modified to conform to master disk standards. For a description of these modifications, please read INF019>STANDARDS.RUNO.

MAKE

Clarified questions asked about physical device type.
"80 OR 300MB STORAGE MOD" -> "STORAGE MODULE OR CMD"
Inform user these are YES or NO questions.

Fix_disk is a new program at Revision 19 which replaces FIXRAT. Documentation may be found in the Primos Revision Document.

User-visible changes:

1) Event logging is re-enabled if necessary when running fix_disk with the -comdev option.

2) -Interactive option will allow user to rebuild DSKRAT if the disk version number is an unknow or illegal value.

3) When fix_disk prints out allowable options, it will now print out the description for the -Interactive option (Which was omitted before).

4) When running fix_disk with -comdev option on a disk with Priority Acls, fix_disk will restore the Priority Acls on the disk after adding it.

5) When rebuilding DSKRAT file, Fix_disk now asks "Storage Module or CMD" instead of "80 or 300 MB".

6) Fix_disk nows makes sure that the three reserved words in a file entry are in fact zero, as these words will be used by File Attributes.

Internal bug fixes:

1) Fix_disk will not compress the MFD if the BOOT entry is missing.

2) Fix_disk will no longer hang on certain bad ufd entries.

3) Fix_disk will now check for disk read error in all places where DAM files are handled.

4) Fix_disk was giving erroneous Directory/Tree Used Incorrect messages due to a bug in the quota checking code.

5) If a baospot is detected and there is no BADSPT file defined on the current partition, Fix_disk will no longer print a spurious "Added to BADSPT file" message.

At Rev 19.0, FUTIL ignores unrecognized file types and ECW values. It also ignores old partition files, i.e. there is no longer any support for old partitions.

FUTIL now returns severity code number in case of FATAL error.

FUTIL now reports file sizes in 1024-word records, not 440-word records. To get it to report in old-style 440-word records, do "FUTIL -NORM". If the token after "FUTIL" is not recognized, the message "Usage: FUTIL [-NORM]" will be printed, but command mode will still be entered to preserve compatibility. However, the new record size (1024 words) will be in effect in that case.

Also, the FROM, TO and ATTACH commands in FUTIL are fixed to scan for cisk volumes by name correctly. They will use RDEN$$ to read the first entry in the MFD, which is the disk record allocation table (RAT), and is named after the volume name.

The old way was to try and open the desired volume name for reading via SRCH$$, and check the returned TYPE field to make sure that one of the "special" bits in the left half were set.

However, revision 19.0 (new ACLs) of PRIMOS may not return the left-half bits of TYPE any longer, mainly because the documentation specifies that the left half will be zero (via implication).

This new method used by FUTIL is similar to methods used by other software which must run under DOS. The build procedure and source files for FUTIL now conform to the rev 19.0 master disk standard.

This is FUTIL 19.0 in its entirety.

HELP Prints information at terminal

HELP [command-name]

If HELP is invoked with a command name, information about that command is printed at the terminal. If HELP is invoked without a command name, it prints a list of commands for which information is available.

The format of command description is:

COMMAND_NAME Brief description

[Abbreviation - if any]

Command line syntax

Text describing command, options, etc. At the end of the listing are any references to further information and the date the HELP information was created or updated.

Information is also available for certain categories or groupings of commands, such as compilers, loaders, etc.
May 1981
OK, como -e

For each chargeable product, there is a cominput file to copy files needed to run the product from the Master Disk to system directories on the user's command disk. Each cominput file is named PRODUCT.INSTALL.COMI (example: COBOL.INSTALL.COMI). Each installation file resides in the directory containing the run files of the product (example: directory COBOL).

In addition to copying commands, libraries, and shared segments, the installation file copies the PRODUCT.SHARE.COMI file from the product directory to ufd SYSTEM, if the product uses shared segments.

In some cases, the number of files to be installed during an initial installation of a product may be greater than the number of files required for a reinstallation of a product. In addition, an initial installation may require that a directory be created on the user's command disk but a reinstallation will not require this. In those cases, there is a separate cominput file provided for inital installations called PRODUCT.INITINSTALL.COMI. As an example, the installation file for COBOL is listed below.

```
/* COBOL.INSTALL.COMI, COBOL, JPC-RHB, 04/01/80
/* installs COBOL into system directories
/* Copyright (C) 1980, Prime Computer, Inc., Wellesley, MA  02181
/*
FUTIL
FROM COBOL>CMDNCO
TO CMDNCO
COPY COBOL.SAVE,NCOBOL.SAVE
FROM COBOL>SYSTEM
TO SYSTEM
COPY C2014A,C2014B,C4000,C02016
FROM COBOL
TO SYSTEM
COPY COBOL.SHARE.COMI
FROM COBOL>SYSOVL
TO SYSOVL
COPY C$$COD
FROM COBOL>LIB
TO LIB
COPY VCOBLB.BIN,NVCOBLB.BIN
QUIT
CO -CONTINUE 6
CO -END
```

For each chargeable product that uses shared segments, there is a cominput file called PRODUCT.SHARE.COMI (example: COBOL.SHARE.COMI) that contains commands that will install the shared files when invoked at the supervisor terminal. Each command share file resides in the directory containing the product (example: COBOL). These cominput files assume that the installation cominput files have copied all required share files and the share cominput file from the directory containing them to ufd SYSTEM. The group of SHARE commands is preceded by the command OPRPRI 1 and followed by the command OPR 0. SHARE

cominput files to install shared libraries must include the shared library package number. As an example, the share command file for COBOL is shown below.

```
/* COBOL.SHARE.COMI, COBOL, JPC-RHB, 04/01/80
/* shares COBOL compiler and library
/* Copyright (C) 1980, Prime Computer, Inc., Wellesley, MA  02181
/*
OPR 1
SHARE SYSTEM>C2014A 2014 700
SHARE SYSTEM>C2014B 2014 700
R SYSTEM>C4000 1/3
SHARE SYSTEM>CO2016 2016
SHARE 2014
OPR 0
CO -CONTINUE 6
CO -END
```

The file C_PRMO.TEMPLATE is supplied in ufd PRIRUN on the Master Disk. This file contains commands to invoke each of the PRODUCT.SHARE.COMI cominput files. A user must examine the file and delete those commands that invoke SHARE.COMI files that the user has not purchased. After the file has been modified, it must be copied to CMDNCO and named C_PRMO.

```
/* C_PRMO.EXAMPLE, PRIRUN, JNK, 07/21/81
/* TEMPLATE FOR MAKING C_PRMO FILE FOR BRINGING UP PRIMOS
/* Copyright (C) 1980, Prime Computer, Inc., Wellesley, MA  02181
CONFIG -DATA                  /* specify CONFIG file after -DATA
ADDISK                        /* specify local disks to be added
AMLC TTY                      /* specify AMLC lines
OPR 1                         /* SHARE REQUIRES OPR 1
SHARE SYSTEM>ED2000 2000      /* SHARE the editor  -  ED
SHARE SYSTEM>S2050 2050 700/* BRING UP SHARED LIBRARIES
R SYSTEM>S4000
SHARE 2050
OPR 0
PROP PRO -START             /* START SPOOLER PHANTOM
PH BATCHQ>PH_GO             /* STARTUP BATCH MONITOR
CO SYSTEM>BASICV.SHARE.COMI 7     /* SHARE BASICV COMPILER
CO SYSTEM>COBOL.SHARE.COMI 7      /* SHARE COBOL COMPILER AND LIBRARY
CO SYSTEM>DBG.SHARE.COMI 7        /* SHARE DEBUGGER
CO SYSTEM>DBMS.SHARE.COMI  7         /* SHARE DBMS
CO SYSTEM>DPTX-DSC.SHARE.COMI 7    /* SHARE DPTX-DSC
CO SYSTEM>DPTX-TCF.SHARE.COMI 7    /* SHARE DPTX-TCF
CO SYSTEM>FED.SHARE.COMI 7         /* SHARE FED
CO SYSTEM>FORMS.SHARE.COMI 7         /* SHARE FORMS LIBRARY
CO SYSTEM>F77.SHARE.COMI 7           /* SHARE F77 COMPILER
CO SYSTEM>MIDAS.SHARE.COMI 7         /* SHARE MIDAS LIBRARY
```

```
CO SYSTEM>PASCAL.SHARE.COMI 7          /* SHARE PASCAL COMPILER
CO SYSTEM>PL1G.SHARE.COMI 7      /* SHARE PL1G COMPILER
CO SYSTEM>POWERPLUS.SHARE.COMI 7              /* SHARE POWER
CO SYSTEM>VISTA.SHARE.COMI 7                  /* SHARE VISTA
CLOSE 7
OPR 1
SHARE SYSTEM>SP2121 2121
R SYSTEM>SP4000 1/10             /* SHARE SPL LIBRARY
SHARE SYSTEM>S$2167 2167
R SYSTEM>S$4000 1/12             /* SHARE SPOOL LIBRARIES
OPR 0
/* SET THE DATE AND TIME **********
CO -END
```

The following command files exist in ufd SYSTEM on the Master Disk. They are for doing installations of an entire Master Disk. These files must be edited to include the MFD names and passwords. They should be run in the order listed.

```
CREATE.STD.COMI
CREATE.ALL.COMI
INSTALL.STD.COMI
INSTALL.ALL.COMI
```

CREATE.STD.COMI should be run if the disk you are installing TO does not have standard master disk directories such as CMDNCO, SYSTEM, LIB, etc.

```
/* CREATE.STD.COMI, SYSTEM, DJF, 08/10/81
/* Creates standard master disk system directories
/* Copyright (c) 1981, Prime Computer, Inc., Natick MA 01760
/*
/* /* this file creates the system directories required for
/* an initial installation of Primos and unchargeable software
/*
A MFD XXXXXX   /* ADD MFD NAME + PASSWD OF DISK YOU WISH TO INSTALL ON
CREATE BATCHQ
CREATE CMDNCO
CREATE DOS
CREATE HELP*
CREATE LIB
CREATE PRIRUN
CREATE SEG
CREATE SEGRUN*
CREATE SPOOLQ
CREATE SYSCOM
CREATE SYSOVL
CREATE SYSTEM
CREATE TOOLS
CREATE RJSPLQ*              /* NEEDED ONLY FOR RJE INSTALLATIONS
```

```
CO -END
```

CREATE.ALL.COMI creates 'special' system directories which are needed to install some chargeable software. For example, to install FORMS you must have a top level FORMS* directory. The user should edit this file and delete the lines which reference software that has not been purchased.

```
/* CREATE.ALL.COMI, SYSTEM, JPC-JNK, 07/21/81
/* creates system directories needed to install chargeable software
/* Copyright (C) 1980, Prime Computer, Inc., Wellesley, MA  02181
/*
ATTACH MFD                  /* PASSWORD SHOULD BE ADDED
CREATE DBMSLB               /* REQUIRED FOR DBMS INSTALLATION
CREATE FORMS*               /* REQUIRED FOR FORMS INSTALLATION
FUTIL                       /* REQUIRED FOR INITIAL INSTALLATION ONLY
FROM FORMS>FORMS*
TO FORMS*
UFDCPY
QUIT
CREATE FED*                 /* REQUIRED FOR FED INITIAL INSTALLATION
CREATE TOOLS                /* REQUIRED FOR PL1G INSTALLATION
CREATE POWER*               /* REQUIRED FOR POWER OR POWERPLUS INSTALLATION
FUTIL                       /* REQUIRED FOR INITIAL INSTALLATION ONLY
FROM POWERPLUS>POWER*
TO POWER*
UFDCPY
QUIT
CREATE POWERCM              /* REQUIRED FOR POWER OR POWERPLUS INSTALLATION
CREATE FAM                  /* REQUIRED FOR PRINET OR X.25 INSTALLATION
CREATE PRIMENET*            /* REQUIRED FOR PRINET OR X.25 (FAMII) INSTALL
CREATE VISTA*               /* REQUIRED FOR VISTA INSTALLATION
CREATE FTSQ*                /* REQUIRED FOR FTS INSTALLATION
CREATE RJSPLQ*              /* REQUIRED FOR RJE INSTALLATION
co -continue 6
co -end
```

INSTALL.STD.COMI installs all uncharceable software.

```
/* INSTALL.STD.COMI, SYSTEM, JPC-JNK, 08/06/81
/* Installs PRIMOS and utilities from the Master Disk
/* Copyright (C) 1980, Prime Computer, Inc., Wellesley, MA  02181
/*
/* change XXX to the current release number, 19.0 would be M190U1
/*
FUTIL
FROM <MXXXU1>CMDNCO
TO CMDNCO
```

```
UFDCPY
FROM <MXXXU1>DOS
TO DOS
UFDCPY
FROM <MXXXU1>LIB
TO LIB
UFDCPY
FROM <MXXXU1>SPOOLQ
TO SPOOLQ
UFDCPY
FROM <MXXXU1>SYSCOM
TO SYSCOM
UFDCPY
FROM <MXXXU1>SYSTEM
TO SYSTEM
UFDCPY
FROM <MXXXU1>SYSOVL
TO SYSOVL
UFDC
F <MXXXU1>PRIRUN
TO PRIRUN
UFDCPY
F <MXXXU1>BATCHQ
TO BATCHQ
UFDCPY
FROM <MXXXU1>SEGRUN*
TO SEGRUN*
UFDC
F <MXXXU1>TOOLS
TO TOOLS
UFDC
F <MXXXU1>SEG
TO SEG
UFDC
FROM <MXXXU1>HFLP*
TO HELP*
UFDC
FROM <MXXXU1>RJSPLQ*
T RJSPLQ*
UFDCPY
QUIT
CO -END
```

INSTALL.ALL.COMI installs all chargeable software by running the individual install files. The user must edit this file and delete the lines which reference software which has not been purchased.

```
/* INSTALL.ALL.COMI, SYSTEM, JPC-JNK, 07/21/81
/* installs all products from the Master Disk
/* Copyright (C) 1980, Prime Computer, Inc., Wellesley, MA  02181
/*
```

```
/* NOTE -- When installing  PRINET & X.25 pauses are encountered
/*         while running the command files to allow you to delete
/*         existing segment directories.  If you are running the
/*         install files as part of this master all.install.comi
/*         command file it is necessary to type 'co continue 22'
/*         rather than 'co continue' to resume the command file
/*         properly after a pause.
CO BASIC>BASIC.INSTALL.COMI 22
CO BASICV>BASICV.INSTALL.COMI 22
CO COBOL>COBOL.INSTALL.COMI 22
CO DBG>DBG.INSTALL.COMI 22
R DBMSEX>DBMS.INSTALL.CPL
CO DPTX-DSC>DPTX-DSC.INSTALL.COMI 22
CO DPTX-TSF>DPTX-TSF.INSTALL.COMI 22
CO DPTX-TCF>DPTX-TCF.INSTALL.COMI 22
CO RJSPLQ*>RJECOM.INSTALL.COMI 22
CO EM1004>EM1004.INSTALL.COMI 22
CO EM200UT>EM200UT.INSTALL.COMI 22
CO EM7020>EM7020.INSTALL.COMI 22
CO EMGRTS>EMGRTS.INSTALL.COMI 22
CO EMHASP>EMHASP.INSTALL.COMI 22
CO EMX80>EMX80.INSTALL.COMI 22
CO EMXBM>EMXBM.INSTALL.COMI 22
CO FORMS>FORMS.INSTALL.COMI 22
CO FED>FED.INSTALL.COMI 22
CO FTN>FTN.INSTALL.COMI 22
CO FTS>FTS.INSTALL.COMI 22
CO F77>F77.INSTALL.COMI 22
CO MIDAS>MIDAS.INSTALL.COMI 22
CO PASCAL>PASCAL.INSTALL.COMI 22
CO PL1G>PL1G.INSTALL.COMI 22
CO POWERPLUS>POWERPLUS.INSTALL.COMI 22
CO PRINET>PRINET.INSTALL.COMI 22
CO RPG>RPG.INSTALL.COMI 22
CO VISTA>VISTA.INSTALL.COMI 22
CO X.25>X.25.INSTALL.COMI 22
CLOSE 22
CO -END
```

There are no internal changes to the LABEL command at this release. This release is being provided only so that LABEL will conform to all file header and master disk release standards.

* NOTE: All products have been modified to conform to master disk standards. For a description of these modifications, please read INF019>STANDARDS.RUNO.

Subject: LATE

Release: MASTER DISK RELEASE 19.0 (JUNE)
PRIMOS 19.0
LATE 17.4

Date May 13, 1981

1. New Functionality
MODIFIED TO CONFORM TO FILE NAMING STANDARDS

2. Problems Fixed
NONE REPORTED

3. Outstanding Problems
NONE

4. Environment
LATE REQUIRES PRIMOS 17.4

5. Installation and Build Procedures
FOLLOW THE STANDARD PRIME BUILD PROCEDURE

Subject: LD

Release: 1.03

Date: 8/19/82

1 New Functionality

New command.

2 Problems Fixed

None.

3 Outstanding Problems

None.

4 Environment

Needs PRIMOS 19.0.65 or greater. (19.0.respin)

5 Installation and Build Procedures

Needs fsulib.build.cpl to be run first. Other than that it is standard.

6 Usage

The usage information for the command follows.

Usage: LD pathname [wild1 ... wild15] [options]

pathname specifies both the directory to be listed, and the first wildcard name. For example, "a>b>@.list" would specify entries in the directory A>B whose names match "@.LIST". If pathname is omitted, "@@" is assumed; that is, all entries in the current directory are selected.

wild1...15 specify additional wildcard names. An entry is selected if it matches either the entryname part of pathname or one of the wildi.

options may be selected in any order from the list below.

Option descriptions:

-NO_HEADER, -NHE
specifies that the header line is not to be output. The header line contains the pathname of the directory listed, the access rights (in parentheses), the records used by this directory if available, and the quota used if this is a quota directory.

-NO_SORT, -NSORT
specifies that the entries listed not be sorted. The default is to sort by ascending NAME within TYPE. TYPEs are always sorted according to the order: file, segment directory, directory, access category.

-SORT_DTM, -SORTD
specifies that the entries be sorted by descending DTM within TYPE. -sort_name must not also be specified.

-SORT_NAME, -SORTN
specifies that the entries be sorted by ascending NAME only not within TYPE). -sort_dtm must not also be specified.

-REVERSE, -RV
specifies that the sort order be reversed from its default. Note that if the sort order of TYPEs is never affected.

The default is to select all entries that match the supplied wildcards. If any of the following options are supplied then only those entries that are protected in that manor will be selected. Note: more than one protection selection may be specified.

-SPECIFIC_PROTECTED, -SPECP
specifies that those entries that are specific protected will be selected.

-DEFAULT_PROTECTED, -DFLTP
specifies that those entries that are default protected will be selected.

-CATEGORY_PROTECTED, -CATP [CAT_NAME]
specifies that those entries that are protected by the access category "cat_name" will be selected. If "cat_name" is missing then all entries that are protected by access categories will be selected.

-DETAIL, -DET
specifies that all attributes be displayed for each entry selected. From left to right these are:

access rights available to this user (for password directories, the protection keys are displayed);

size of entry in physical disk records;

quota of entry in physical disk records (directories only);

type of entry;

setting of concurrency lock on entry (" " for system, excl" for N readers or 1 writer, "updt" for N readers and 1 writer, and "none" for N readers and N writers);

incremental dump switch ("dmp" if the entry has been dumped);

delete-protection switch ("pr" if protected);

date-time modified;

name of entry;

and type of protection (name of access category protecting entry, or (Specific) for specific protected, or blank for protected by default);

The default output format is to list only the name of each entry, four across. To print a subset of "detail" format information, use one or more of the following options.

-PROTECT, -PRO
specifies that protection information (access rights, delete-protect switch, and type of protection) be printed for each entry.

-DTM
specifies that date-time-modified be printed for each entry.

-SIZE
specifies that size information (size of entry, quota for directories only) be printed for each non-access category entry. A size of -1 will be reported for any entry for which the user does not have R (or L) permission.

-SINGLE_COLUMN, -SGLCOL
is useful only if the default (names only) format is used. In this case, specifies that names are to be printed one per line instead of four per line.

LIB7 Rev. 19.0

A change was made to support pathnames with passwords.

ABSTRACT

This document describes the Changes made to LOAD for PRIMOS release 19.0

LOAD for rev 19.0

1 LOAD

There is one bug fix to LOAD for rev 19.0. Load will now set the Warning flag and return code whenever the code passes through ERK with a non-zero error code.

* NOTE: All products have been modified to conform to master disk standards. For a description of these modifications, please read INF019>STANDARDS.RUNO.

Subject: LOGPRT

Release: REV19.0

DATE: June 29, 1982

1 New Functionality

The functionality added to the LOGPRT command for REV19 provides for three new event types for logging System events and five new event types for logging Network events.

The command line to invoke LOGPRT is as follows ([] indicates optional parameter):

R TOOLS>LOGPRT [<outtreename>] [opt> <opt> ...]

<outtreename> The destination for LOGPRT's output. If 'TTY' is specified, the output will be to the user's terminal. If <outtreename> is omitted, output will be to the file 'LOGLST' (or 'NETLST' for network) in the home UFD. Any other specification will be taken as a treename to which the output will be directed.

<opt> An option keyword, possibly followed by subfields. All option keywords begin with a hyphen and may be abbreviated to a unique left substring (with the exception of the -PURGE option).

The modifications which were made only affect the -TYPE option:

-TYPE t1 t2 ...Process entries only of the indicated types. The new types added for the rev19.0 version of LOGPRT are:

System Event Types

MCHECK Machine checks (not including memory parity)

GUIET Primos entering Quiet machine check mode
       (happens after 1024 ECC errors since cold start)

BADENT Bad LOGREC entries

Network_Event_Types

When specifying network event types, the -NET option must be specified before the -TYPE option, otherwise LOGPRT will try to match the system types.

NPXTHR	NPX was throttled on transmit or receive
NPXRCV	NPX got an unanticipated receive status
NPXCLR	NPX master's circuit was cleared with an unexpected clearing cause
NPXSEQ	NPX found a sequence error in bounce detect
NPXCON	NPX got an unexpected circuit status in call setup

2 Bug_Fixes

Input logging files for both system and network events are no longer located in the directory CMDNCO on logical disk zero. System input logging files are located in the ufd 'LOGREC*' on logical disk zero. Network input logging files are located in the ufd 'PRIMENET*' on logical disk zero. System logging file names are in the form 'LOG.MM/DD/YY', and network logging file names are in the form 'NET_LOG.MM/DD/YY', where 'MM/DD/YY' is either the date on which a cold start of the machine was performed or when an EVENT_LOG -ON command was issued from the system console to enable event logging.

The input event logging file may be specified by including the pathname of the file after the '-INPUT' option on the LOGPRT command line. If the network event logging file is specified, '-NET' should be the first option on the command line. If an input event logging file is not specified, LOGPRT uses the most recently created log file found in the respective directories, LOGREC* or PRIMENET*.

The 'HELP' display now causes the tty screen to scroll. Type anything but upper- or lower-case 'q', 'qu', 'qui', or 'quit' to continue display.

DSWPARITY checking now dependent on whether processor is either a 750 or 850. Code assumes bit interpretation is similar.

'D' board now is 'J' board in DSWPARITY.

LFERNEXT now taken in positive sense, therefore, XOR in code not done.

'DELETE' option now correctly spools output file before deleting it as described in 'HELP' display.

1 THE MAGNET SUBSYSTEM

1.1 INTRODUCTION TO MAGNET

To transfer data on a magnetic tape from a non-Prime operating system to PRIMOS and vice versa, you must write the magnetic tape in an interchange format that both operating systems can understand. MAGNET is the PRIMOS subsystem that allows you to read and write magnetic tapes in these various interchange formats. MAGNET is intended primarily for users who have had prior experience with magnetic tapes.

Some of the facilities that the MAGNET subystem provides are:

- Declaration and modification of I/O objects
- Linkage of I/O objects with PRIMOS global variables
- Translation to and from various character sets
- User-definable character sets
- Seven-track binary packing and unpacking
- Tape positioning
- Physical tape copying
- Logical data transfer between devices with support of multiple destinations
- Support of unlabelled and ANSI and IBM standard labelled tapes
- Support of fixed- and IBM, ANSI, and Prime variable-length records
- Support of old (pre-rev 18.4) MAGNET features for the READ, WRITE, COPY, and POSITION subcommands
- Support for tape operations within Batch mode (operator intervention)
- Tape-to-spooler and disk-to-spooler support
- Break key handling

1.2 INVOKING MAGNET

To invoke MAGNET, you must be at PRIMOS command level. Type MAGNET after the PRIMOS prompt (OK,). At this point there are several PRIMOS command line options you can specify. -SILENT causes only MAGNET severity 2 or 3 errors to be printed at your terminal. You may also specify -USER or -OPERATOR (may be abbreviated to -OPR). These two options direct the printing of MOUNT and DISMOUNT messages on either the operator's terminal or your terminal, respectively. For detailed information on these options, see Appendix B.

After you type MAGNET and possibly a mode option, the subsystem prints a release number and gives you a prompt. For example:

```
OK, MAGNET
[MAGNET, Rev. 4.4]
>
```

After the MAGNET prompt (>), you type a subcommand line that contains one or more of of the 19 MAGNET subcommands. Depending on the subcommand, the line may also contain one or more object-names and/or one or more options. The MAGNET subcommand line format is as follows:

```
> subcommand   object-name(s)   option=(value),...option=(value)
```

There are a few points to remember when typing subcommand lines:

- Input can be up to 1,000 characters long.
- You can use either upper or lowercase alpha characters or a combination of both.
- Free format is allowed on the subcommand line. (Spacing is optional.)

1.3 MAGNET SUBCOMMANDS

There are 19 subcommands in MAGNET. Fourteen of these are the basic MAGNET subcommands, which define, manipulate, and transfer data within the subsystem. These subcommands are described in the beginning of this section. The five other subcommands are used less frequently and are described later in this section under ADVANCED MAGNET. Table 7-1 lists each MAGNET subcommand, its type, and its function.

Table 7-1. MAGNET Subcommands

Subcommand	Type	Function
/*	----	Puts comments in your MAGNET dialog
CLOSE	Data Definition/ Manipulation	Closes a disk, tape or spool file
COPY	Data Transferral	Copies one or more physical files from one tape to another
DECLARE	Data Definition/ Manipulation	Associates one or more options with an object-name
DELETE	Data Definition/ Manipulation	Deletes one or more declared object-names
DISPLAY	Data Definition/ Manipulation	Shows all options associated with a declared object-name
LIST	Data Definition/ Manipulation	Lists all declared object-names
LOAD	Data Definition/ Manipulation	Loads a translation table
MODIFY	Data Definition/ Manipulation	Changes an option value or adds an option to an object-name
MOVE	Data Transferral	Moves a logical file from a source object to one or more destination objects

Table 7-1. MAGNET Subcommands (continued)

Subcommand	Type	Function
NOISY	----	Allows printing of severity 1 messages
POSITION	Data Transferral	Positions a tape to a specific file number and record number
QUIT	----	Exits from MAGNET and returns you to PRIMOS command level
READ	Data Transferral	Reads a file from a tape and transfers it to a disk file
RENAME	Data Definition/ Manipulation	Renames declared object-names
SAVE	Data Definition/ Manipulation	Saves options associated with an object-name in a PRIMOS global variable
SILENT	----	Disallows printing of severity 1 messages
TRANSLATE	Data Definition/ Manipulation	Specifies the translation associated with an object-name
WRITE	Data Transferral	Writes a disk file onto tape

1.4 DATA DEFINITION AND MANIPULATION SUBCOMMANDS

There are ten MAGNET subcommands that define and manipulate data within the subsystem. Seven of these sucommands, DECLARE, MODIFY, LIST, DISPLAY, DELETE, RENAME, and TRANSLATE are described in the following paragraphs. The other 3 data definition and manipulation subcommands are described later in this section under ADVANCED MAGNET.

1.4.1 The DECLARE Subcommand

You use DECLARE to associate one or more characteristics with one object-name. The subcommand line format is as follows:

```
                              SPOOL=(spool_file_name)
                              TAPE=(devno)
> DECLARE  object-name        DISK=(pathname)  [,OPTION=(value)...]
                              EXTERNAL=(gvar)
                              LIKE=(object-name)
```

An object-name identifies either a tape, disk, or spool file. You always specify an object-name with the DECLARE subcommand. It is good practice to create object-names that are mnemonic. You can use any combination of up to 32 alpha characters, numerals, and the two characters period (.) and underscore (_). Any of these characters may appear first in the object-name. You may declare up to 100 object-names.

An OPTION provides information that describes an object-name. The first option you specify on the DECLARE subcommand line must always be TAPE, DISK, SPOOL, LIKE or EXTERNAL. Other options, if any, follow. Table 7-2 lists each MAGNET subcommand option, its type, applicable subcommand(s), and its function.

There are 42 possible options you can specify with the DECLARE subcommand. Twenty-three of these are the basic options that you use for most operations in MAGNET. These are described here. The other 19 are options that you may use less frequently. These are described later in this section under ADVANCED MAGNET.

1.4.1.1 The TAPE Option:

You specify the TAPE option if the object-name preceding it on the subcommand line refers to a tape file. The value you assign to this option is the logical device number of the tape drive on which the tape is mounted (a number from 0 to 7). You must assign your tape drives at PRIMOS command level before you invoke MAGNET. For more information, see the ASSIGN command in Section 4, USER CONTROL OF TAPE DRIVES. In the following example, the DECLARE subcommand defines the object-name TFILE to be a tape file. It is located on logical device (tape drive) number 7.

```
> DECLARE TFILE TAPE=(7)
>
```

Table 7-2. MAGNET Subcommand Options

Option	Type	Applicable Subcommand(s)	Function
ACCESS	Tape	DECLARE, MODIFY	Identifies the ACCESS field on level 1 labels
AMOUNT	----	COPY	Specifies the number of files to copy
AT	Spool	DECLARE, MODIFY	Specifies location at which to print a spool file
BFACTOR	Tape	DECLARE, MODIFY	Specifies the number of logical records per physical tape block
BUFFERS	Tape	DECLARE, MODIFY	Specifies the number of buffers being used
BYPASS	Tape	DECLARE, MODIFY	Indicates to MAGNET whether or not to ignore tape labels and identify its files by number
CHARACTERS	Tape	DECLARE, MODIFY	Specifies whether 1 or 2 characters per word are to be read or written
CONTROL	Spool	DECLARE, MODIFY	Specifies the type of line-printer carriage control desired
COPIES	Spool	DECLARE, MODIFY	Specifies the number of copies to be printed on a line-printer
CREATEN	Tape	DECLARE, MODIFY	Identifies the creation date field for Level 1 file labels
DEFER	Spool	DECLARE, MODIFY	Specifies the time a spool file is to be

printed

Table 7-2. MAGNET Subcommand Options (continued)

Option	Type	Applicable Subcommand(s)	Function
DENSITY	Tape	DECLARE, MODIFY	Specifies density of the tape being read or written
DISK	Disk	DECLARE, MODIFY, LOAD	Specifies the location of a disk file
EXCHANGE	Tape	DECLARE, MODIFY	Indicates whether, for each logical record, high and low order bytes within words are to be exchanged
EXPIRE	Tape	DECLARE, MODIFY	Identifies expiration date fields for level 1 labels
EXTERNAL	----	DECLARE, MODIFY, SAVE	Indicates that more options can be found in an external PRIMOS global variable
FILEID	Tape	DECLARE, MODIFY	Identifies the labelled file to be read or written
FILENO	Tape	DECLARE, MODIFY	Specifies a file number
FORM	Spool	DECLARE, MODIFY	Specifies the type of line-printer forms to be used for printing
FORMAT	Disk Tape Spool	DECLARE, MODIFY	Identifies the type of records being read or written
GENERATION	Tape	DECLARE, MODIFY	Identifies level 1 generation fields on labels

Table 7-2. MAGNET Subcommand Options (continued)

Option	Type	Applicable Subcommand(s)	Function
ITEMS	----	LIST	Specifies what type of objects should be listed
LABELS	Tape	DECLARE, MODIFY	Identifies the type of labels on a tape
LEVEL	Tape	DECLARE, MODIFY	Specifies the amount of labelling on a tape
LIKE	----	DECLARE, MODIFY	Identifies another object-name whose options, option values, and translation edit tokens are to be duplicated
LINENOS	Spool	DECLARE, MODIFY	Specifies whether line numbers should be printed on line-printer output
LINES	----	LOAD	Specifies the number of lines in a user translation table
LRECL	Disk Tape Spool	DECLARE, MODIFY	Specifies the number of bytes in each logical record
MAXIO	Tape	DECLARE, MODIFY	Limits the maximum I/O transfer size
MODE	----	POSITION	Instructs MAGNET to position either absolutely or relatively
NEXTCHAIN	Disk Tape Spool	DECLARE, MODIFY	Declares the next object in a chain of objects

Table 7-2. MAGNFT Subcommand Options (continued)

Option	Type	Applicable Subcommand(s)	Function
OFFSET	Tape	DECLARE, MODIFY	Specifies the size of a field, at the beginning of physical records, that contains control characters to be ignored
OWNER d	Tape	DECLARE, MODIFY	Identifies the owner i field on a VOL1 label
PARITY	Tape	DECLARE, MODIFY	Identifies the parity of a seven-track tape
POSTACTION	Disk Tape Spool	DECLARE, MODIFY	Specifies action to perform when closing a tape, disk, or spool file
PREACTION	Disk Tape spool	DECLARE, MODIFY	Specifies action to perform when opening a tape, disk or spool file
PREVCHAIN	Disk Tape Spool	DECLARE, MODIFY	Declares the previous object in a chain of objects
PRINT	----	COPY	Prints the size of the first physical record in each tape file
PROTECT	Tape	DECLARE, MODIFY	Prevents writing on a tape
RECORDNO f	Tape	DECLARE, MODIFY	Specifies the number o physical records to be spaced forward or backward

Table 7-2. MAGNET Subcommand Options (continued)

Option	Type	Applicable Subcommand(s)	Function
SEQUENCE	Tape	DECLARE, MODIFY	Identifies the file sequence field on Level 1 file labels
SPOOL	Spool	DECLARE, MODIFY	Defines a spool file
TAPE	Tape	DECLARE, MODIFY	Defines a tape object
TRACKS	Tape	DECLARE, MODIFY	Specifies the number of tracks on a tape
TYPE	----	LOAD	Identifies the type of a translation table
VERSION	Tape	DECLARE, MODIFY	Identifies level 1 generation version fields
VISUAL	Tape	DECLARE, MODIFY	Identifies the external visual id of a tape reel
VOLSER	Tape	DECLARE, MODIFY	Identifies the volume serial id field on a VOL1 label

1.4.1.2 The SPOOL Option:

You specify the SPOOL option if the object-name preceding it on the subcommand line refers to a spool file. The value you assign to this option must be an alpha string of no more than 32 characters length. The spool option value is printed as a banner or your spooler output. In the following example, the object-name PRODUCTS is a spool file. The option value, AUTOS, will appear as the banner on the line-printer output:

```
> DECLARE PRODUCTS SPOOL=(AUTOS)
>
```

1.4.1.3 The DISK Option:

You specify the DISK option if the object-name preceding it on the subcommand line refers to a disk file. The value you assign to this option must be a PRIMOS filename or pathname. In the following example, the object-name DFILE is a PRIMOS disk file, SOCSECNUMS, in the UFD SFA:

```
> DECLARE DFILE DISK=(SFA>SOCSECNUMS)
>
```

1.4.1.4 The LRECL and BFACTOR Options:

LRECL is the logical record length option. It specifies the number of bytes in each logical record in fixed-length format, or the maximum number of bytes in each logical record in variable-length format.

BFACTOR is the blocking factor option. It specifies the number of logical records per physical tape block. For more information on logical record sizes and blocking factors, see the discussion on RECORDS, GAPS, AND BLOCKS in Section 1, MAGNETIC TAPES.

LRECL and BFACTOR together specify the maximum size of a tape file's physical records. You obtain this maximum size by multiplying the value of LRECL by the value of BFACTOR. The maximum size can be no larger than 12,288 bytes (this number is smaller if you are using ANSI or IBM variable-length records). BFACTOR is strictly a tape option; you use it only with tape objects. LRECL is for tape, disk, and spool objects. You must always specify an LRECL value for all objects, no matter what type (tape, disk or spool) they are. If you do not specify BFACTOR it defaults to the value of 1.

For disk and spool objects, LRECL specifies the size in characters of each record in a binary file or the maximum size in characters of each record in an ASCII file. The value must be an even number between 2 and 12,288. In the following example you declare X to be a disk file, SUEFILE. Each logical record in the disk file is 80 characters long:

```
> DECLARE X DISK=(*>SUEFILE), LRECL=(80)
>
```

Note

With a binary disk file, you must also specify FORMAT=(FIXED). For ASCII files you should specify FORMAT=(VAR/PRIME). For more information, see the discussion of the FORMAT option later in this section.

In the next example, the tape file EMPNAMES has a blocking factor of 10 logical records per physical record. Each logical record contains a maximum of 80 bytes. Therefore, the maximum physical record size is 800 bytes:

```
> DECLARE EMPNAMES TAPE=(3), LRECL=(80), BFACTOR=(10)
>
```

1.4.1.5 The TRACKS Option:

You specify the TRACKS option to identify the number of tracks on your tape. The two possible values are either 7 or 9, for seven- or nine-track tapes respectively. TRACKS is strictly a tape option, and it defaults to the value of 9. In the following example, a tape object, EMPNAMES, is a nine-track tape located on logical device number 6.

```
> DECLARE EMPNAMES TAPE=(6), TRACKS=(9)
>
```

1.4.1.6 The DENSITY Option:

You use DENSITY to specify the density of the tape that you are reading or writing. For seven-track tapes, you can specify either 200, 556, or 800 (bpi) as the value. For nine-track tapes, you can specify either 800, 1600, or 6250 (bpi) as the value. DENSITY is solely a tape option, and it defaults to the value of 800. In the following example, DATAFILE is a tape object located on logical device number 2. It is a seven-track tape with a density of 556 bpi.

```
> DECLARE DATAFILE TAPE=(2), TRACKS=(7), DENSITY=(556)
>
```

1.4.1.7 The FORMAT Option:

You use the FORMAT option to identify the type of records you are reading from or writing to your tape, disk or spool file. There are four possible values you can specify for tape objects:

- FIXED (for fixed-length records)
- VAR/ANSI (for ANSI standard variable-length records, nine-track only)
- VAR/IBM (for IBM standard variable-length records, nine-track only)
- VAR/PRIME (for Prime variable-length records)

There are two possible values you can specify for disk or spool objects:

- FIXED (for fixed-length records, binary disk files)
- VAR/PRIME (for Prime variable-length records, ASCII disk files)

With Prime variable-length records, the logical record length equals the physical record length; you can specify any size up to a maximum of 12,288 bytes. For tape objects it is important to note that you must always specify a value of 1 for the BFACTOR option for this format only. Prime variable-length tape records have no record control word (RCW) or block control word (BCW). (For more information on the Prime variable-length tape format, see Section 2, Prime Variable-Length Format.)

For tape objects, FORMAT defaults to FIXED for a value. For disk and spool objects, FORMAT defaults to VAR/PRIME for a value. In the following example, CANDY is a tape object located on logical device number 5. It is a nine-track tape with fixed-length records, each of which may attain a maximum 100 characters:

```
> DECLARE CANDY TAPE=(5), TRACKS=(9), FORMAT=(FIXED),
LRECL=(100)
>
```

In the following example, X is a spool object with banner SUEFILE. It is a Prime variable-length file with records that may attain a maximum of 80 characters :

```
> DECLARE X SPOOL=(SUEFILE), LRECL=(80), FORMAT=(VAR/PRIME)
>
```

Note

With disk or spool files you must also specify a logical record length. For more information see the discussion of the LRECL option in this section.

1.4.1.8 The PARITY Option:

PARITY specifies either odd or even parity for a seven-track tape. It is a seven-track tape option only; nine-track tapes are always written in odd parity. There is no default for this option. You must always specify it when you declare a seven-track tape object. In the following example, STATS is a tape object located on logical device number 2. It is a seven-track tape with odd parity:

```
> DECLARE STATS_TAPE=(2), TRACKS=(7), PARITY=(ODD)
>
```

1.4.1.9 The PROTECT Option:

This option is the software equivalent of a write-enable ring on a reel of magnetic tape. PROTECT prevents you from writing on your tape. You specify either YES or NO as the value. If you do not specify PROTECT, it defaults to NO. This option is for tape objects only. In the following example, VITAL_INFO is a tape object located on logical device 0. It is protected from writing:

```
> DECLARE VITAL_INFO TAPE=(0), PROTECT=(YES)
>
```

1.4.1.10 The LABELS Option:

You use the LABELS option to identify the type of tape labels that you are using. There are three possible values you can specify:

- NONE (you are not using any type of labels)
- ANSI (you are using ANSI standard labels)
- IBM (you are using IBM standard labels)

LABELS is strictly a tape option. If you do not specify it, it defaults to NONE. In the following example, INCOME is a tape object located on logical device number 6. It is a nine-track tape with ANSI standard labels written at 800 bpi:

```
> DECLARE INCOME_TAPE=(6), LABELS=(ANSI)
>
```

1.4.1.11 The LEVEL Option:

You use the LEVEL option to identify the number of labels present on your tape. LEVEL limits the number of labels written on output; it has no effect whatsoever on input. There are four possible values that you can specify:

- 0 (for an unlabelled tape)
- 1 (for Level 1 labels only: HDR1, EOF1, or EOV1)
- 2 (for Level 1 labels and HDR2, EOF2, and/or EOV2 labels only)
- 3 (for Level 1 labels, HDR2, EOF2, EOV2, and user labels only)

For more information on these levels of tape labels, see Section 3 MAGNETIC TAPE LABELS in this manual.

Note

Prime does not yet support user label processing. You may specify a value of 3 but MAGNET treats it as if you had specified a value of 2.

LEVEL is for tape objects only. If you do not specify it, the default is 0. In the following example, MYTAPE is a tape object located on logical device number 3. It has two levels of IBM standard labels:

```
> DECLARE_MYTAPE_TAPE=(3),_LABELS=(IBM),_LEVEL=(2)
>
```

1.4.1.12 The FILEID Option:

You use the FILEID option to identify the name of the labelled file you wish to read or write. FILEID specifies the file-id field of Level 1 labels. The value you specify for this option is a string of no more than 17 characters. If you specify less than 17 characters, the value is blank-padded on the right. The string can be any mixture of the following:

- Upper and/or lowercase alpha characters
- Numbers
- Any other characters except right and left parentheses, equal signs, commas, apostrophes, and blanks

FILEID is solely a tape option. If you do not specify it, the

default is 17 blanks. In the following example, SYSIN is an IBM standard labelled file located on logical device number 2. The file-id field of the Level 1 labels is SOCSECNUMS:

```
> DECLARE SYSIN TAPE=(2), LABELS=(IBM), FILEID=(SOCSECNUMS)
>
```

1.4.1.13 The FILENO Option:

FILENO identifies the number of the file you wish to work with on your tape. It is strictly a tape option for unlabelled files or for labelled files whose labels are being bypassed. (For more information, see the description of the BYPASS option.) Alternatively, you use the FILENO option with POSITION, a MAGNET subcommand, to specify absolute or relative file positioning on a tape. (For more information, see the POSITION subcommand description later in this section.)

If you use this option to specify an unlabelled tape file or a labelled file whose labels are bypassed, the value must be a number greater than or equal to 0. (A value of 0 indicates that you wish to work with the file at your current position on the tape.) If you use FILENO for positioning, the value may be less than 0.

In the following example, OUTFILE is an unlabelled tape object located on logical device number 1. You are working with the second file on the tape:

```
> DECLARE OUTFILE TAPE=(1), LABELS=(NONE), FILENO=(2)
>
```

1.4.1.14 The OWNER Option:

You use the OWNER option to identify the owner identificaton field on a VOL1 label. The value you specify depends on whether your tape is ANSI or IBM labelled. For both types of labelled tapes, the value is a string of characters. You can specify up to 14 characters for ANSI labelled tapes and up to 10 characters for IBM labelled tapes. If you specify a value that contains less than the maximum number of characters for either format, the value is blank-padded on the right. As for the FILEID option, the string can be any mixture of the following:

- Upper and/or lowercase alpha characters
- Numbers
- Any other characters except right and left parentheses, equal signs, commas, apostrophes, and blanks

OWNER is strictly a tape option. If you do not specify this option for an ANSI labelled tape object, it defaults to 14 blank characters. The default for an IBM labelled tape object is 10 blank characters. In the following example, WAGES is an ANSI labelled tape object located on logical device number 4. The owner identification field is SUSANADLEY:

```
> DECLARE_WAGES_TAPE=(4), LABELS=(ANSI), OWNER=(SUSANADLEY)
>
```

1.4.1.15 The VOLSER Option:

VOLSER identifies the volume serial identificaton field (VOLSER number) on a VOL1 label. The value is a string of no more than six characters that can be a mixture of any of the following:

- Upper and/or lowercase alpha characters
- Numbers
- Any other characters except right and left parentheses, equal signs, commas, apostrophes, and blanks

If you specify less than six characters, the value is blank-padded on the right. The VOLSER option is solely for tape objects. If you do not specify it, the default is six blank characters. In the following example, DATA_TEST is an ANSI labelled tape object located on logical device number 3. The VOLSER id is 63A:

```
> DECLARE DATA_TEST TAPE=(3), LABELS=(ANSI), VOLSER=(63A)
>
```

1.4.1.16 The RECORDNO Option:

The RECORDNO option identifies the number of physical records that you wish to space either forward or backward. You use this option only in conjunction with the POSITION and COPY subcommands. (For more information, see the descriptions of the POSITION and COPY subcommands later in this section.) The value you specify with RECORDNO is either a positive or negative number (for spacing either forward or backward, respectively). This option is for tape objects only. If you do not specify it, the default is 0. In the following example, BAR is a tape object located on logical device 0. RECORDNO indicates to the POSITION subcommand that you wish to position the tape five filemarkers backward and then 17 physical records forward:

```
> DECLARE BAR TAPE=(0), FILENO=(-5), RECORDNO=(17)
>
```

1.4.1.17 The VISUAL Option:

You use VISUAL to specify the external visual identification of the tape reel you are using. The external visual identification is located on the tape reel enclosure. Visual simplifies the identification of tape reels for MOUNT and DISMOUNT requests. The value is a string of 32 characters that can be any combination of the following:

- Upper and/or lowercase alpha characters
- Numbers
- Any other characters except right and left parentheses, equal signs, commas, apostrophes, and blanks

VISUAL is solely for tape objects. If you do not specify it, the default is 32 blank characters. In the following example, STEVE is a tape object located on logical device number 7. The external visual identification of this tape reel is MYTAPE2:

```
> DECLARE STEVE TAPE=(7), VISUAL=(MYTAPE2)
>
```

1.4.1.18 The BYPASS Option:

With BYPASS, you indicate to MAGNET whether or not to ignore the labels on your tape and then identify the tape files by number as opposed to file-ids. The value you specify is either YES or NO. There are a few important points to keep in mind when you use this option:

- o BYPASS is valid only when you are reading a tape.
- o You must always specify, with the LABELS option, the type of labels to be bypassed (either ANSI or IBM).
- o BYPASS causes MAGNET to ignore all associated label options (VOLSER, OWNER, FILEID).

This option is for tape objects only. If you do not specify it, the default is NO. In the following example, FILE2 is an ANSI labelled tape object located on logical device number 6. You indicate that you want MAGNET to ignore all the labels and then position at the second logical file on the tape:

```
> DECLARE FILE2 TAPE=(6), LABELS=(ANSI), BYPASS=(YES),
FILENO=(2)
>
```

1.4.1.19 The BUFFERS Option:

You specify the BUFFERS option to indicate how many internal buffers are being used for magnetic tape I/O. The value is a numeral, which may be 0 through 10 only. If you specify BUFFERS with a DECLARE subcommand and then later modify it with a lesser value, the extra buffers are released (returned to the free storage list).

The BUFFERS option is for tape objects only and it defaults to a value of 0. In the following example, you declare a tape object, DOT. It is located on logical device number two and has three internal I/O buffers:

```
> DECLARE DOT TAPE=(2), BUFFERS=(3)
>
```

Note

To use a specific tape object with the READ, WRITE, or MOVE data transferral subcommands, you must always specify the BUFFERS option with a value greater than 0. (These subcommands are described in detail later in this section.)

1.4.1.20 The AT option:

The AT option specifies the location at which to print a spool file. It is strictly a spool option whose value is a string of 32 characters that can be any combination of the following:

- Upper and/or lowercase alpha characters
- Numbers
- Any other characters except right and left parentheses, equal signs, commas, apostrophes, and blanks

In the following example, the object-name GROSS_PAY, a spool object, will be printed at a location called BOSTON:

```
> DECLARE GROSS_PAY  SPOOL=(EMPLOYEES),  AT=(BOSTON)
```

1.4.1.21 The COPIES Option:

The COPIES option specifies the number of copies of your spool file you would like printed. The value is a number from 1 to 32767. The default number of copies printed is 1. This option is provided for spool objects only. An example, in which 23 copies of the spool OBJECT GROSS_PAY (from the previous exmple) are desired is:

```
> DECLARE GROSS_PAY  SPOOL=(EMPLOYEES),  COPIES=(23)
```

1.4.1.22 The FORM option:

The FORM option specifies the type of line-printer forms you would like your spool file printed on. The value of this option is a string of no more than 6 characters that can be a combination of any of the following:

- Upper and/or lowercase alpha characters
- Numbers
- Any other characters except right and left parentheses, equal signs, commas, apostrophes, and blanks

The FORM option is provided for spool objects only and defaults to a value of 6 blank characters. In the following example, a spool object named TOTALS is declared with the AT, COPIES and FORM options:

```
> DECLARE TOTALS  SPOOL=(RECEIPTS),  COPIES=(12),
FORM=(3PART)
```

1.4.2 The LIST Subcommand

You use LIST to see what object-names you have declared. The subcommand line format is as follows:

```
                        (DISK)
                        (TAPE)
> LIST    [ ITEMS =     (SPOOL)  ]
                        (OPEN)
                        (CLOSED)
```

MAGNET prints out a chart that shows all declared object-names and the type of the object (DISK, TAPE or SPOOL). In the following example, LIST shows the three declared objects TFILE, DFILE, and SFILE to be tape, disk, and spool objects, respectively.

```
> LIST
__________object-names__________     type      open status
TFILE                                Tape      Closed
DFILE                                Disk      Closed
SFILE                                Spool     Closed
>
```

1.4.2.1 The ITEMS Option:

The ITEMS option is used to limit the output from the LIST subcommand. You can direct LIST to display all tape objects, disk objects or all spool objects previously declared. In addition, you can direct LIST to print only those objects that are not currently in use (CLOSED) or which are in use (OPEN). An object may have been left in use if, for example, you hit the BREAK key while a data transferral operation was underway. Using the previous LIST example, you could request that only spool objects be displayed:

```
> LIST  ITEMS=(SPOOL)
__________object-names__________     type      open status
SFILE                                Spool     Closed
>
```

1.4.3 The DISPLAY Subcommand

DISPLAY shows all options associated with a declared object-name. The subcommand line format is as follows:

```
> DISPLAY object-name
```

The subcommand output shows all possible options that you can specify for an object-name. Tape options appear for tape objects, disk options appear for disk objects, and spool options appear for spool objects. Those options that you previously declared for the specified object-name appear with their values, along with those that automatically default if you do not specify them. The following is an example of DISPLAY output for a tape object, SYSIN. You first declare SYSIN with four options and their values. The LIST subcommand confirms that SYSIN is a declared tape object. Output from DISPLAY then shows all possible options for SYSIN, but only shows the values for those you declared (TAPE, FILENO, LABELS, and BYPASS) and those you did not declare that default automatically.

```
> DECLARE SYSIN TAPE=(3), FILENO=(5), LABELS=(ANSI), BYPASS=(YES)
> LIST
          object-names                    type       open status
-----------------------------------     -----      -----------
SYSIN                                     Tape       Closed
> DISPLAY SYSIN
***  Information for TAPE object SYSIN ***
PREV=                                    / NEXT=
PREACTION          =                      POSTACTION        =
TAPE               =       3              TRACKS            = 9
DENSITY            = 800                  CHARACTERS        = 2
LABELS             = ANSI                 LEVEL             =       0
BYPASS             = YES                  PROTECT           = NO
FORMAT             = FIXED                PARITY            =
LRECL              =       0              BFACTOR           =       1
OFFSET             =       0              BUFFERS           =       0
MAXIO              =   10000              VISUAL            =
FILENO             =               5      RECORDNO          =          0
EXCHANGE           = NONE                 FILEID            =
VOLSER             =                      OWNER             =
GENERATION         = 0001                 VERSION           = 00
CREATE             =   00000              EXPIRE            =   00000
ACCESS             =                      SEQUENCE          = 1
TRANSLATE     = (A*)
>
```

The next example shows DISPLAY output for a disk object, SYSOUT. You declare SYSOUT with only one option, DISK. The LIST subcommand confirms that SYSOUT is a disk object. Output from the DISPLAY subcommand shows all possible disk options for SYSOUT, but only prints values for DISK, LRECL, and FORMAT. (The values shown for LRECL and FORMAT are default values.)

```
> DECLARE SYSOUT DISK=(SUE>SOURCE>REV19>FILE2)
> LIST
          object-names                    type       open status
-----------------------------------       ----       -----------
SYSOUT                                    Disk       Closed
> DISPLAY  SYSOUT
***  Information for DISK object SYSOUT  ***
DISK  = SUE>SOURCE>REV19>FILE2
PREV=                                      / NEXT=
PREACTION          =                        POSTACTION    =
LRECL              = 0                      FORMAT        =  VAR/PRIME
>
```

The next example shows DISPLAY output for a spool object. BONUSES. You declare BONUSES with only one option, SPOOL. The LIST subcommand confirms that BONUSES is a spool object. Output from the DISPLAY subcommand shows all possible spool options for BONUSES. Note that only the SPOOL, AT and COPIES options have received values. Other options, such as FORMAT and LRECL received default values.

```
> DECLARE  BONUSES  SPOOL=(HOME_OFFICE), AT=(BOSTON), COPIES=(12)
> LIST
          object-names                    type       open status
-----------------------------------       ----       -----------
BONUSES                                   Spool      Closed
> DISPLAY  BONUSES
*** Information for SPOOL object BONUSES ***
SPOOL = HOME_OFFICES
PREV=                                     / NEXT=
PREACTION          =                       POSTACTION        =
LRECL              = 0                     FORMAT            = VAR/PRIME
AT                 = BOSTON                CONTROL           = NONE
COPIES             =     12                DEFER             =
FORM               =                       LINENOS           = NO
```

1.4.4 The RENAME Subcommand

You use RENAME to change the name of a declared object. The subcommand line format is as follows:

```
> RENAME old-object-name new-object-name
```

In the following example, EMPNOS1 is a previously declared object, as shown in the LIST subcommand output. You rename EMPNOS1 to EMPNOS2, and the new-object-name appears when you give a second LIST subcommand.

```
> LIST
          object-names                type      open status
--------------------------------      ----      -----------
EMPNOS1                               Tape      Closed
> RENAME  EMPNOS1  EMPNOS2
> LIST
          object-names                type      open status
--------------------------------      ----      -----------
EMPNOS2                               Tape      Closed
>
```

1.4.5 The MODIFY Subcommand

MODIFY changes a declared object or option. For example, you can add an option to an object-name or change the value of an option. The subcommand line format is as follows:

```
> MODIFY object-name OPTION=(value) [,OPTION=(value)...]
```

On the subcommand line, options can be in any order; you do not have to specify DISK, TAPE, SPOOL, LIKE, or EXTERNAL as the first option. All options that you can use with the DECLARE subcommand are also valid for MODIFY. There are four important points to remember when you use the MODIFY subcommand:

- You can only modify objects-names already declared.
- You can only specify tape options for a tape object.
- You can only specify disk options for a disk object.
- You can only specify spool options for a spool object.

In the following example you declare a tape object, BETA_TEST. You specify the options TAPE, TPACKS, FORMAT, DENSITY, and FILENO with their corresponding values. A LIST subcommand confirms that BETA_TEST is a declared tape object. The output from a DISPLAY subcommand shows the declared options and values for BETA_TEST. You now wish to modify BETA_TEST by adding some options to those you have already declared. With the MODIFY subcommand, you specify the additional options LABELS, LRECL, BFACTOR, and BYPASS. Again, a LIST subcommand confirms that BETA_TEST is a tape object. Now when you give the DISPLAY subcommand, the output shows not only the options and corresponding values you specified when you first declared BETA_TEST, but also those you specified when you modified it. In both cases with the DISPLAY output you also see the options that default automatically if you do not specify them.

```
> DECLARE_BETA_TEST_TAPE=(0),_TRACKS=(9),_FORMAT=(VAR/ANSI),
DENSITY=(6250),_FILENO=(2)
> LIST
          object-names                    type       open status
--------------------------------     -----      -----------
BETA_TEST                                Tape       Closed
> DISPLAY_BETA_TEST
*** Information for TAPE object BETA_TEST ***
PREV=                                     / NEXT=
PREACTION          =                       POSTACTION         =
TAPE               =       0               TRACKS             = 9
DENSITY            = 6250                  CHARACTERS         = ?
LABELS             = NONE                  LEVEL              =       0
BYPASS             = NO                    PROTECT            = NO
FORMAT             = VAR/ANSI              PARITY             =
LRECL              =       0               BFACTOR            =       1
OFFSET             =       0               BUFFERS            =       0
MAXIO              =   10000               VISUAL             =
FILENO             =               2       RECORDNO           =         0
EXCHANGE           = NONE                  FILEID             =
VOLSER             =                       OWNER              =
GENERATION         = 0001                  VERSION            = 00
CREATE             =   00000               EXPIRE             =   00000
ACCESS             =                       SEQUENCE           =     0
TRANSLATE    = (A*)
```

```
> MODIFY BETA_TEST LABELS=(ANSI), LRECL=(100), BFACTOR=(10),
BYPASS=(YES)
> LIST
          object-names                  type       open status
--------------------------------     ----       -----------
BETA_TEST                                Tape       Closed
> DISPLAY BETA_TEST
*** Information for TAPE object BETA_TEST ***
PREV=                                     / NEXT=
PREACTION          =                       POSTACTION         =
TAPE               =       0               TRACKS             = 9
DENSITY            = 6250                  CHARACTERS         = 2
LABELS             = ANSI                  LEVEL              =       0
BYPASS             = YES                   PROTECT            = NO
FORMAT             = VAR/ANSI              PARITY             =
LRECL              =    100                BFACTOR            =      10
OFFSET             =      0                BUFFERS            =       0
MAXIO              =  10000                VISUAL             =
FILENO             =              2        RECORDNO           =         0
EXCHANGE           = NONE                  FILEID             =
VOLSER             =                       OWNER              =
GENERATION         = 0001                  VERSION            = 00
CREATE             =  00000                EXPIRE             =  00000
ACCESS             =                       SEQUENCE           =     0
TRANSLATE     = (A*)
>
```

1.4.6 The DELETE Subcommand

This subcommand deletes one declared object-name. When you delete an object-name you also clear the space it occupied internally within MAGNET. The subcommand line format is as follows:

```
> DELETE object-name
```

In the following example, output from a LIST subcommand shows that you have two previously declared tape objects, STATS1 and STATS2. You delete STATS2, and a second LIST subcommand reflects the deletion.

```
> LIST
          object-names                type        open status
--------------------------------      ----        -----------
STATS1                                Tape        Closed
STATS2                                Tape        Closed
> DELETE STATS2
> LIST
          object-names                type        oper status
--------------------------------      ----        -----------
STATS1                                Tape        Closed
>
```

1.4.7 The TRANSLATE Subcommand

TRANSLATE indicates, for tape objects only, that you want particular logical records translated to or from industry standard ASCII, EBCDIC, or BCD. Translation is also available for several different seven-track binary formats. For example, suppose you have an 80-character logical record. The first 40 characters specify a person's name in EBCDIC and the last 40 characters specify binary information. You could specify a translation for the first 40 characters to Prime ASCII and the second 40 characters could be read as they are. The subcommand line format for TRANSLATE is as follows:

```
> TRANSLATE object-name (Crepetition factor] edit token  list)
```

An edit token specifies how you want particular fields of a logical record translated. An edit token is an alpha character optionally followed by any of the following:

- Upper and/or lowercase alpha characters
- Numbers
- An asterisk
- Any other characters except right and left parentheses, equal signs, commas, apostrophes, and blanks

An edit token list is a group of edit tokens separated by commas. Each edit token or edit token list may be preceded by an optional repetition factor, which is always a number. An example of an edit token is A6, which specifies a translation of six characters to industry standard ASCII. Table 7-3 lists each translation edit token and its function. Facilities are available for inserting fill characters, specifying no translation (reading or writing raw data), and for utilitzing user-defined translation tables. (For detailed information on all edit tokens and translation character set tables, see Appendix A. For detailed information on modifying user-defined translation tables, see the description of the LOAD subcommand later in this section.)

In the following example of the TRANSLATE subcommand, you are reading from a tape. You first specify a translation of six characters from industry standard ASCII to Prime ASCII (P6). You next specity a translation, for 40 pairs of fields, of six characters from industry standard ASCII to Prime ASCII (P6,40(P6). You next indicate that you want to delete the next two characters in the field (P6,40(P6,D2). Lastly, any characters that remain in the logical record are translated to Prime ASCII from industry standard ASCII (P6,40(P6,D2),P*) .

```
> TRANSLATE_BETA_TEST_(P6,40(P6,D2),P*)
>
```

If you wish to change a translation, you do it by simply giving another TRANSLATE command line. For example, to change the translation of BETA_TEST shown above, you might type the following:

```
> TRANSLATE_BETA_TEST_(B4,30(B7,D3),B*)
>
```

Remember that you can always check on the translation you specified when you give the DISPLAY command.

Table 7-3. Translation Edit Tokens

Edit_Token	Function
A	Enforces industry standard ASCII
B	Translates Prime ASCII to and from BCD code
C	Moves the column position within fields of logical records
D	Deletes characters in logical records
E	Translates Prime ASCII to and from EBCDIC code
F	Specifies fill characters
G	Specifies seven-track packing code 2424
H	Specifies seven-track packing code 4242
I	Specifies seven-track packing code 2466
J	Specifies seven-track packing code 4266
K	Specifies seven-track packing code 6246
L	Specifies seven-track packing code 6426
M	Specifies seven-track packing code 6624
N	Specifies seven-track packing code 6642
O	Specifies no translation
P	Enforces Prime ASCII
Q through U	Reserved for future use
V through Z	Specify user codes

1.5 DATA TRANSFERRAL SUBCOMMANDS

There are four MAGNET subcommands that transfer data within the subsystem. They are READ, WRITE, MOVE, and COPY. READ, WRITE, and MOVE are logical copy operations. COPY is a physical tape copy operation.

1.5.1 The READ Subcommand

You use the READ subcommand to move information in a tape file (source object) to a disk file (destination object). The tape and disk files must be previously declared object-names. (The pre-Rev 18.4 functionality of READ is still supported. See Appendix C.) The subcommand line format for READ is as follows:

```
> READ tape-object-name disk-object-name
```

In the following example you first declare a tape object, SYSIN, which is located on logical device 0. You specify the third file on the tape, which has five records per physical tape block with each record 100 characters long. By default you are working with fixed-length records. There are no tape labels. Within the MODIFY subcommand you use another option, BUFFERS, to specify that three internal buffers are being used by the tape object SYSIN. To perform a READ operation, your BUFFERS value for the tape object must be greater than or equal to 1. Next you declare a disk object, SYSOUT, which is a disk file, DIMPLE, in the UFD SFA. Finally, your READ subcommand line causes the information in SYSIN to be read into SYSOUT.

```
> DECLARE SYSIN TAPE=(0), FILENO=(3), LRECL=(100),
BFACTOR=(5)
> MODIFY SYSIN BUFFERS=(3)
> DECLARE SYSOUT DISK=(SFA>DIMPLE), LRECL=(100)
> READ SYSIN SYSOUT
>
```

1.5.2 The WRITE Subcommand

The function of the WRITE subcommand is the opposite of READ. You use WRITE to move information in a disk file (source object) to a tape file (destination object). As with the READ subcommand, the disk and tape files must be previously declared object-names. To perform a WRITE operation, the value you specify with the BUFFERS option for a tape object must be greater than or equal to 1. (The pre-Rev 18.4 functionality of WRITE is still supported. See

Appendix C.) The subcommand line format for WRITE is as follows:

```
> WRITE disk-object-name tape-object-name
```

In the following example, you are working with the same two previously declared objects, SYSIN and SYSOUT. However, you wish to move the information in SYSOUT into SYSIN:

```
> WRITE SYSOUT SYSIN
>
```

1.5.3 The MOVE Subcommand

You use this subcommand to simultaneously move information in a source object into a maximum of seven destination objects. Source and destination objects must be previously declared and may be either disk or tape objects. To perform a MOVE operation, the value you specify for the BUFFERS option must be greater than or equal to one for each tape object you use. The subcommand line format for MOVE is as follows:

```
> MOVE source-object-name destination-object-name(s)
```

In the following example you first declare your source object, NEWDATA, to be a disk file. You next declare the first of your four intended destination objects, PLACE1. This is a tape file with characteristics as shown. You then declare your next three destination objects, PLACE2 (a tape file), PLACE3 (a tape file), and PLACE4 (a disk file). These files have the characteristics shown. The output from a LIST subcommand shows the objects you have just declared. Finally, you use the MOVE subcommand to simultaneously move the information in NEWDATA to PLACE1, PLACE2, PLACE3, and PLACE4.

```
> DECLARE NEWDATA DISK=(SFA>STATS), LRECL=(80)
> DECLARE PLACE1 TAPE=(0), LRECL=(100), BFACTOR=(10),
BUFFERS=(3),FORMAT=(VAR/ANSI), FILENO=(10)
> DECLARE PLACE2 TAPE=(1), LRECL=(80), BFACTOR=(20), BUFFERS=(6)
> MODIFY PLACE2 FORMAT=(VAR/IBM), FILENO=(1)
> DECLARE PLACE3 TAPE=(5), LRECL=(150), BFACTOR=(1), BUFFERS=(9),
LABELS=(ANSI), LEVEL=(2). VOLSER=(597219), OWNER=(PARIS), FILEID=
(SRCPRG1)
> DECLARE PLACE4 DISK=(PARIS>SOURCE>PROGRAM.FTN), LRECL=(80)
```

```
> LIST
          object-names                    type       open status
-------------------------------------     ----       -----------
NEWDATA                                   Disk       Closed
PLACE1                                    Tape       Closed
PLACE2                                    Tape       Closed
PLACE3                                    Tape       Closed
PLACE4                                    Disk       Closed
> MOVE  NEWDATA  PLACE1  PLACE2  PLACE3  PLACE4
>
```

The source object specified with a MOVE subcommand may be either a tape or disk object. The destination object(s) specified may be either tape, disk, or spool objects.

1.5.4 The COPY Subcommand

You use the COPY subcommand to copy information from a source tape to one or more destination tapes. This is a physical, as opposed to a logical, copy function. (The pre-Rev 18.4 functionality of COPY is still supported. See Appendix C.)

Note

> Your tape will be positioned before it is copied. You indicate the starting position on the tape by use of the FILENO and RECORDNO options with the DECLARE and MODIFY subcommands. (These two options are discussed at the beginning of this section.) The FILENO/RECORDNO pair must be an absolute position; the values for both options must be greater than or equal to 0. If they are equal to 0, none of the tapes are pre-positioned. (For more information on tape positioning, see the POSITION subcommand later in this section.)

The format of the COPY subcommand line is as follows:

```
                                                                  (n)
> COPY source-object-name destination-object-name(s) AMOUNT=     or,
                                                                  (*)

          (YES)
    PRINT=  or
          (NO)
```

You may specify up to eight *object-names* in the subcommand line (one *source-object-name* and from one to seven *destination-object-names*). The tape objects must be previously declared and must refer to different tape drives. You must always specify the AMOUNT option in the COPY subcommand line.

1.5.4.1 The AMOUNT Option:

The AMOUNT option identifies the number of physical tape files to be copied. (Copying is done from one file marker to another file marker.)

Note

> Since the COPY subcommand performs a physical tape copy as opposed to a logical tape copy, tape labels are not recognized.

There are two possible values you can specify for the AMOUNT option, either a numerical value (n) or an asterisk (*). A numerical value identifies the number of physical files that are to be copied. The value can be a number between 1 and 32,767. If you wish to copy everything on your tape until you reach the first filemarker after the physical end-of-tape (EOT), then you specify an asterisk (*) for a value. If the physical EOT (end-of-tape) is detected, and if you did not specify AMOUNT=(*), and then a single file marker is read, copying stops and a new source tape is requested. (See Appendix B for more information.) Likewise, if the physical EOT is detected on a destination tape, a single file marker is written and a new tape is requested. (See Appendix B.)

1.5.4.2 The PRINT Option:

The PRINT option specifies that the size, in words, of the first physical record of each tape file is printed on your terminal. There are two values you can specify for this option, either YES or NO. The default value is NO.

If you specify PRINT=(YES), a table like the following is printed as the copy operation proceeds:

File #	Words Read	Total Blocks Read
1	200	250
2	350	97
3	790	1123
.	.	.
.	.	.
.	.	.

When MAGNET has completed copying a file, the total number of physical records in that file is also displayed in the third column of the table. The PRINT option replaces the "PRINT RECORD SIZES?" question in the pre-Rev 18.4 MAGNET dialog. (See Appendix C for additional information.)

The following example illustrates the use of the COPY subcommand and the AMOUNT option. You first declare a tape object, SYSIN, that is located on logical device 0. The FILENO/RECORDNO option pair instructs MAGNET to rewind the tape and then position it to file number 1, record number 1. You then declare SYSOUT, a tape object located on logical device number 1, and SYSOUT2, a tape object located on logical device number 2. With SYSOUT and SYSOUT2, the FILENO/RECORDNO option pair functions the same way as in the SYSIN declaration. You then give a COPY subcommand line which copies all of SYSIN, the source-object-name, to SYSOUT and SYSOUT2, the two destination-object-names. Remember that all of SYSIN is copied because you specified an asterisk (*) as the value for the AMOUNT option.

```
> DECLARE SYSIN TAPE=(0), FILENO=(1), RECORDNO=(1)
> DECLARE SYSOUT TAPE=(1), FILENO=(1), RECORDNO=(1)
> DECLARE SYSOUT2 TAPE=(2), FILENO=(1), RECORDNO=(1)
> COPY SYSIN SYSOUT SYSOUT2 AMOUNT=(*), PRINT=(NO)
>
```

1.6 ADDITIONAL SUBCOMMANDS

1.6.1 The POSITION Subcommand

The POSITION subcommand positions a tape to a specific FILENO/RECORDNO pair. Positioning can be either absolute (rewind the tape first then space forward) or relative (space forward or backward from your current position on the tape). Before tape positioning, both the FILENO and RECORDNO options must be set by the DECLARE or MODIFY subcommands. For absolute positioning, the values for both the FILENO and RECORDNO options must be greater than or equal to 1. For relative positioning the two options can have any values. (The pre-Rev 18.4 functionality of POSITION is still supported. See Appendix C.) The subcommand line format for POSITION is as follows:

```
                                  (ABSOLUTE)
> POSITION object-name  MODE=        or
                                  (RELATIVE)
```

The object-name is the name of the tape you wish to position and must always be a previously declared tape object. You must always specify the type of positioning (absolute or relative) with the MODE option.

1.6.1.1 The MODE Option:

You use the MODE option to specify either absolute or relative tape positioning on a POSITION subcommand line. When you use the POSITION subcommand you must always specify either ABSOLUTE or RELATIVE for the MODE option.

The following example illustrates the use of both the POSITION subcommand and the MODE option. Assume that you have four physical files with six physical records in each of the files. You first declare T1, a tape object located on logical device 0. You set the values of the FILENO and RECORDNO options to be 3 and 6, respectively. Next you give a POSITION subcommand that specifies absolute positioning for T1. This instructs MAGNET to rewind the tape, space forward two file markers, and then space forward five records.

```
> DECLARE T1 TAPE=(0), FILENO=(3), RECORDNO=(6)
> POSITION T1 MODE=(ABSOLUTE)
>
```

There are two ways to rewind your tape. In the following example, you rewind X by giving 1 as the value for both the FILENO and RECORDNO options and then specifying absolute positioning with the POSITION subcommand. You also rewind Y by giving very large, negative values for both FILENO and RECORDNO and then specifying relative positioning on the POSITON subcommand line.

```
> DECLARE X TAPE=(0), FILENO=(1), RECORDNO=(1)
> DECLARE Y TAPE=(1) FILENO=(-1000000) RECORDNO=(-1000000)
> POSITION X MODE=(ABSOLUTE)
> POSITION Y MODE=(RELATIVE)
>
```

1.6.2 The QUIT Subcommand

You use QUIT when you wish to exit from MAGNET and return to PRIMOS. The subcommand line format is as follows:

```
> QUIT
OK,
```

Note

MAGNET ignores anything you type on the subcommand line that follows QUIT.

1.6.3 The /* Subcommand

You use this subcommand to put comments in your MAGNET dialog. It is the same as the PRIMOS comment command. Your comment can be up to 1,000 characters long. The subcommand line format is as follows:

```
> /* ccccccccccccccccccc...
>
```

1.7 ADVANCED MAGNET

There are advanced MAGNET features, subcommands, and options that you will use less frequently. These features supplement the basic MAGNET functions discussed up to this point in this section.

1.7.1 Advanced Options

There are 19 additional options that you can specify with the DECLARE and MODIFY subcommands.

1.7.1.1 The EXTERNAL Option:

When you use MAGNET, there may be specific sets of options that you will want to use more than once. The EXTERNAL option allows you to save and reload these options so that you do not have to retype them each time. You can also use EXTERNAL with the SAVE subcommand. (See the description of SAVE later in this section.)

Options are stored for later use in PRIMOS global variables. You can create these sets of options at PRIMOS level by using the PRIMOS command SET_VAR. To do this, your global variable file must be activated. The following example illustrates how you can set up groups of options at PRIMOS level by using SET_VAR.

```
OK, SET_VAR .SYSIN := TAPE=(0), LRECL=(80),
BUFFERS=(2), FILENO=(1)
OK,
```

For a description of PRIMOS global variable commands, refer to The CPL User's Guide.

The value you specify for the EXTERNAL option must be a valid PRIMOS global variable name. It is a string of no

more than 32 characters. The first character in the string must always be a period (.). The other characters in the string can be any mixture of the following:

- Upper and/or lowercase alpha characters
- Numbers
- Underscores
- Periods

With the DECLARE or MODIFY subcommands, you may specify EXTERNAL first in the option list on the subcommand line. However, no matter where EXTERNAL appears in the option list, MAGNET ignores any options that follow it. If you specify EXTERNAL as the first option with a DECLARE subcommand, the TAPE or DISK options must appear first within the CPL variable.

Note

When you give the DISPLAY subcommand, any edit tokens you specified with the TRANSLATE subcommand are shown in the output. These edit tokens are not regarded as options and thus cannot be saved or loaded by the EXTERNAL option.

There are a few points to remember when you use the EXTERNAL option:

- A global variable cannot contain an EXTERNAL option. EXTERNAL can only appear within MAGNET.
- Any options that appear within a global variable overwrite any of the same options already declared or modified.
- Global variables cannot contain more than 1,024 characters.
- You may specify the same name for an EXTERNAL value and an object.

In the following example, at PRIMOS command level, you first create a global variable file, IO.GVAR. A LIST_VAR command shows that no global variables are defined. You then create two global variables for the file. These are .SYSIN and .SYSOUT. Output from a second LIST_VAR command shows that these two variables are now defined. Next, invoke MAGNET and declare two objects, SYSIN and SYSOUT. You specify the EXTERNAL option for each of these objects, the values being .SYSIN and .SYSOUT respectively. This instructs MAGNET to get the global variables values in .SYSIN and .SYSOUT and put them

into the internal variables SYSIN and SYSOUT. Next you give the LIST subcommand, which confirms that SYSIN and SYSOUT are declared objects. Finally, the DISPLAY subcommand output for both SYSIN and SYSOUT shows all the values associated with variables SYSIN and SYSOUT.

```
OK, DEFINE_GVAR IO.GVAR -CREATE
OK, LIST_VAR
No global variables are defined.
OK, SET_VAR .SYSIN := TAPE=(0),BUFFERS=(2),LRECL=(80),
FILENO=(1)
OK, SET_VAR .SYSOUT := DISK=(TESTDIR>SFA>FILE3),LRECL=(80)
OK, LIST_VAR
.SYSOUT              DISK=(TESTDIR>SFA>FILE3),LRECL=(80)
.SYSIN               TAPE=(0),BUFFERS=(2),LRECL=(80),FILENO=(1)
OK, MAGNET
[MAGNET, Rev. 19.0]
> DECLARE_SYSIN_EXTERNAL=(.SYSIN)
> DECLARE_SYSOUT_EXTERNAL=(.SYSOUT)
> LIST
          object-names                  type       open status
--------------------------------      ----       -----------
SYSIN                                 Tape       Closed
SYSOUT                                Disk       Closed
> DISPLAY_SYSIN
*** Information for TAPE object SYSIN ***
PREV=                                  / NEXT=
PREACTION          =                     POSTACTION    =
TAPE               =       0             TRACKS        = 9
DENSITY            = 800                 CHARACTERS    = 2
LABELS             = NONE                LEVEL         =     0
BYPASS             = NO                  PROTECT       = NO
FORMAT             = FIXED               PARITY        =
LRECL              =      80             BFACTOR       =     1
OFFSET             =       0             BUFFERS       =     2
MAXIO              =   10000             VISUAL        =
FILENO             =               1     RECORDNO      =         0
EXCHANGE           = NONE                FILEID        =
VOLSER             =                     OWNER         =
GENERATION         = 0001                VERSION       = 00
CREATE             =  00000              EXPIRE        =  00000
ACCESS             =                     SEQUENCE      = 0000
TRANSLATE     = (A*)
> DISPLAY_SYSOUT
*** Information for DISK object SYSOUT ***
DISK = TESTDIR>SFA>FILE3
PREV=                                  / NEXT=
PREACTION          =                     POSTACTION    =
LRECL              =      80             FORMAT        = VAR/PRIME
> QUIT
OK,
```

1.7.1.2 The LIKE Option:

This option allows you to "copy" the option values and translation edit tokens of one object to another object. For example, if you are using a multi-reel labelled tape file, you must use two or more (depending on the number of tape reels) tape objects linked by the NEXTCHAIN option (see the description of the NEXTCHAIN option below). Chained, labelled tapes usually have many options in common, such as LABELS, LEVEL, FILEID, LRECL, BFACTOR, FORMAT OWNER, TRACKS, PROTECT, ACCESS, CREATE, EXPIRE, etc. It is very tedious to retype these options and their values two or more times, so LIKE can be used to "duplicate" these options and option values. As an example, consider the following set of DECLARE and MODIFY statements:

```
> DECLARE INFILE  TAPE=(0), LRECL=(100), BFACTOR=(20),
FORMAT=(VAR/ANSI)
> MODIFY INFILE  LABELS=(ANSI), LEVEL=(2), ACCESS=(X),
OWNER=(PARIS)
> MODIFY INFILE  FILEID=(JANUARYPAY), VOLSER=(PRSAL1),
VISUAL=(0131811)
> MODIFY INFILE BUFFERS=(4), GENERATION=(0002), VERSION=(12)
> MODIFY INFILE NEXTCHAIN=(INFILE2)
> TRANSLATE INFILE (A70,6(F*),3(A6,2(F)))
> /*
> DECLARE INFILE2  TAPE=(0), LRECL=(100), BFACTOR=(20),
FORMAT=(VAR/ANSI)
> MODIFY INFILE2  LABELS=(ANSI), LEVEL=(2), ACCESS=(X),
OWNER=(PARIS)
> MODIFY INFILE2  FILEID=(JANUARYPAY), VOLSER=(PRSAL2),
VISUAL=(0131812)
> MODIFY INFILE2 BUFFERS=(4), GENERATION=(0002), VERSION=(12)
> TRANSLATE INFILE2 (A70,6(F*),3(A6,2(F)))
```

This sequence of statements can be shortened dramatically by typing:

```
> DECLARE INFILE  TAPE=(0), LRECL=(100), BFACTOR=(20),
FORMAT=(VAR/ANSI)
> MODIFY INFILE  LABELS=(ANSI), LEVEL=(2), ACCESS=(X),
OWNER=(PARIS)
> MODIFY INFILE  FILEID=(JANUARYPAY), VOLSER=(PRSAL1),
VISUAL=(0131811)
> MODIFY INFILE BUFFERS=(4), GENERATION=(0002), VERSION=(12)
> TRANSLATE INFILE (A70,6(F*),3(A6,2(F)))
> /*
> DECLARE INFILE2 LIKE=(INFILE), VISUAL=(0131812)
> MODIFY INFILE2 NEXTCHAIN=(INFILE2), VOLSER=(PRSAL2)
>
```

All options and option values and translation edit tokens associated with INFILE are duplicated in INFILE2. It should be remembered, however, that INFILE2 is an object in its own right. If the BUFFERS option should be modified in INFILE, for example, the update is not reflected in INFILE2. Another MODIFY command would have to be issued to change the BUFFERS value for INFILE2. Likewise, two DELETE commands are necessay to erase these objects and return their associated memory to the free memory pool. A LIST command will list two objects, not one, and two DISPLAY commards are necessary to view each object's option values.

1.7.1.3 The PREACTION and POSTACTION Options:

These options specify the action to be taken when opening and closing disk, spool and tape files. You can, when opening, position to the end of the disk file so that new information read into it will be concatenated at the end. It is also possible to rewind a reel of tape before searching for and opening a tape file. After a tape file has been closed, you may rewind and/or unload the tape reel. A spool file may or may not have a banner page and may or may not require a page eject upon print completion.

The PREACTION option has three possible values: APPEND, for disk objects; REWIND, for tape objects; and SUPRESS for spool objects. POSTACTION also has three possible values: REWIND and UNLOAD, both for tape objects only; and NOEJECT for spool objects only. Both PREACTION and POSTACTION default to six blanks.

In the following example, you declare a disk object, SUEA, to be a disk file, SUEFILE. You specify the PREACTION option with an APPEND value. This instructs MAGNET to position SUEFILE at its end upon opening.

```
> DECLARE SUEA DISK=(SUEFILE), PREACTION=(APPEND)
>
```

Note

The PREACTION option with the APPEND value is useful only when you are reading into a disk file.

In the next example, you declare a tape object, STEVEP, to be a tape file located on logical device number 7. Each logical record in the file is 80 characters long. You specify the PREACTION option with a REWIND value. This instructs MAGNET to rewind the tape before opening file number 23.

```
> DECLARE STEVEP TAPE=(7), PREACTION=(REWIND), LRECL=(80),
FILENO=(23)
>
```

In the following example, you declare a spool object, EXP, with banner ANIMALS, 12 copies and a print location of OZ. The PREACTION option specifies that the banner is not to be printed.

```
> DECLARE EXP SPOOL=(ANIMALS),COPIES=(12),AT=(OZ),
PREACTION=(SUPPRESS)
```

Note

The PREACTION option with the REWIND value is useful only if your tape is positioned beyond a desired location.

In the next example you are working with the same tape object, STEVEP. However, you are now working with file number 42 on the tape, and you specify the POSTACTION option with a REWIND value. This instructs MAGNET to rewind the tape after closing tape file number 42:

```
> DECLARE STEVEP TAPE=(7), POSTACTION=(REWIND)
> MODIFY STEVEP LRECL=(80), FILENO=(42)
>
```

In this final example you are again working with file number 42 in the tape object STEVEP. You specify the POSTACTION option with an UNLOAD value. This instructs MAGNET to rewind and unload (dismount) the tape after closing tape file number 42:

```
> DECLARE STEVEP TAPE=(7), POSTACTION=(UNLOAD)
> MODIFY STEVEP LRECL=(80), FILENO=(42)
>
```

1.7.1.4 The PREVCHAIN and NEXTCHAIN Options:

It is possible to link together or "chain" a number of tape, spool or disk objects. You would do this if, for example, you wanted to specify which tape should be mounted when you reach the end of the tape you are currently using. If you do not specify chaining and the end-of-tape is detected, MAGNET requests that a new tape be mounted. However, since MAGNET does not initially recognize any characteristics of the new tape, it has no way of knowing if the new tape is the correct one. Consequently, this procedure is only useful for unlabelled tapes.

If you are using labelled tapes and you expect to use a multi-reel file, you must specify chaining. When you declare your tape object, you specify data about the first reel of tape the file is on. You must also specify the NEXTCHAIN option on the DECLARE or MODIFY subcommand lines. The NEXTCHAIN option contains the name of another tape object. When end-of-tape is detected, MAGNET rewinds the tape, looks up the information in the tape object specified by NEXTCHAIN, and requests that the new tape be mounted.

When you specify a chain of tapes, you should include the BUFFERS option only for the first object in the chain. If you specify BUFFERS for any object after the first in the chain, those buffers are released when that tape object is activated. Other options that specify physical characteristics of the tape (such as TAPE, TRACKS, and DENSITY) or logical characteristics of the file (such as LRECL, BFACTOR, FORMAT, and LABELS) should have the same value among all objects in the chain. In order to identify a new tape to be mounted, you should specify an external visual identification by using the VISUAL option. (For more information, see the description of VISUAL earlier in this section.)

The NEXTCHAIN option identifies the next object in a chain. PREVCHAIN identifies the previous object, but it is not currently used by MAGNET in this release. In addition, NEXTCHAIN is currently used only for tape objects. Values of both options are names of other tape, spool, or disk objects. A tape object cannot point to a non-tape object; the same restriction applies to disk and spool objects. Both options are for tape, spool and disk objects, and the defaults are blank characters.

In the following example, you declare a tape object, TAXES, with a number of options. It is the first object in a chain, and the next object in the chain is TAXES2. TAXES2 has exactly the same option values as TAXES except for OWNER, VOLSER, and VISUAL. Note that, since you did not specify NEXTCHAIN for TAXES2, it is the last object in the chain. When end-of-tape is detected on TAXES, the tape is rewound and a message appears

on either your terminal or the operator's console (depending on whether you specify -USER or -OPERATOR at PRIMOS command level; see Appendix B). After the new tape specified by TAXES2 is mounted and either you or the operator replies to MAGNET's message, processing continues with the object TAXES2.

```
> DECLARE_TAXES_TAPE=(0), LRECL=(100), BFACTOR=(20)
> MODIFY_TAXES_DENSITY=(6250), FORMAT=(VAR/ANSI)
> MODIFY_TAXES_LABELS=(ANSI), LEVEL=(2)
> MODIFY_TAXES_FILEID=(1980), OWNER=(SIRAP)
> MODIFY_TAXES_VOLSER=(SFATAX), VISUAL=(X125)
> MODIFY_TAXES_BUFFERS=(7), MAXIO=(12000)
> MODIFY_TAXES_NEXTCHAIN=(TAXES2)
> /*
> DECLARE_TAXES2_TAPE=(0), LRECL=(100), BFACTOR=(20)
> MODIFY_TAXES2_DENSITY=(6250), FORMAT=(VAR/ANSI)
> MODIFY_TAXES2_LABELS=(ANSI), LEVEL=(2)
> MODIFY_TAXES2_FILEID=(1980), OWNER=(NEVETS)
> MODIFY_TAXES2_VOLSER=(SFATX2), VISUAL=(X126)
> MODIFY_TAXES2_MAXIO=(12000)
>
```

1.7.1.5 The ACCESS Option:

ACCESS identifies the accessibility of a tape object. Accessibility is installation-dependent. Prime software does not check the accessibility fields of the VOL1, HDR1, EOF1, and EOV1 labels. Consequently, this option is provided for tape protection purposes on other systems.

The value you specify, the access code, is installation-dependent. In other words, it is not recognized by Prime software but will be recognized by software of another manufacturer. An access code is a single character that may be any one of the following:

- An upper or lowercase alpha character
- A number
- Any other character except right or left parentheses, an equal sign, a comma, an apostrophe, or a blank

ACCESS is strictly a tape option. It defaults to a blank character, which signifies no protection. In the following example you declare a tape object, SYSIN. It is located on logical device number 3 and has ANSI standard labels. You are working with the file MYFILE on tape number 1254. The access code is Y.

```
> DECLARE_SYSIN_TAPE=(3), LABELS=(ANSI), FILEID=(MYFILE),
VOLSER=(1254), ACCESS=(Y)
>
```

Note

There is an accessibility field in both the VOL1 and HDR1 labels. These do not necessarily have the same value. The HDR1 field is set by the ACCESS option. The VOL1 field can be set by the PRIMOS LABEL command when you are initializing your reel of tape. (For more information on LABEL, see Section 6.)

1.7.1.6 The CREATE and EXPIRE Options:

These two options specify the creation and expiration dates of a labelled tape file. These dates are placed in the appropriate fields of the HDR1, EOF1, and EOV1 labels, and provide protection. Note that if the current date is earlier than the expiration date, the file cannot be overwritten. All creation and expiration dates are stored in MAGNET both internally and in the label field of the Julian format.

Note

The first file on your tape should have the latest expiration date. This is because, when you overwrite a file, all succeeding data on the tape is considered invalid.

The values you specify with CREATE and EXPIRE are one of the following:

- A date in the form YYDDD, where YY represents the year, and DDD represents the day of the year (Julian date)
- An asterisk (*), which indicates the current date
- MM/DD/YY, where MM represents the month (January is represented as 01). DD is the day of the month, and YY is the year

CREATE and EXPIRE are strictly tape options. They both default to five 0's. This indicates that the file has already expired. (You have data on the tape that may be overwritten.) In the following example, you declare VITAL to be a tape object located on logical device 0. The tape has ANSI standard labels. With three subsequent MODIFY subcommands you specify additional options for VITAL. The tape has two levels of labels, the owner of the tape is JOHN_RODDY, and the volume serial number is 847519. Finally, the creation date of the tape is the current date, and the expiration date you specify

is June 10, 1987.

```
> DECLARE_VITAL_TAPE=(0), LABELS=(ANSI)
> MODIFY_VITAL_LEVEL=(2), OWNER=(JOHN_RODDY)
> MODIFY_VITAL_VOLSER=(847519)
> MODIFY_VITAL_CREATE=(*), EXPIRE=(06/10/87)
>
```

1.7.1.7 The GENERATION and VERSION Options:

The GENERATION option specifies an edition (the most current update) of a file. The VERSION option specifies a printing (a copy of the most current edition) of a file. Both options are stored in the Generation and Generation-Version fields of the HDR1, EOF1, and EOV1 labels.

The values you specify for these two options are numbers in the form nnnn (for GENERATION) and nn (for VERSION). The valid values for GENERATION are 0001 through 9999. The valid values for VERSION are 00 through 99. GENERATION defaults to 0001 and VERSION defaults to 00.

The followng example illustrates the subcommand line for the 23rd printing of the 52nd edition of the tape object CLOUDS. It is an IBM labelled tape located or logical device number 4. You are working with the file RAINBOW.

```
> DECLARE_CLOUDS_TAPE=(4), LABELS=(IBM),
FILEID=(RAINBOW), GENERATION=(0052), VERSION=(23)
>
```

1.7.1.8 The OFFSET Option:

OFFSET specifies the length in characters of any field in a physical tape block which precedes the data portion of that block. This option is for tape objects only. The value is a number from 0 to 99. The default value is 0.

In the following example, you declare the tape object, BLUESKY, located on logical device number 1. You are working with file number 12, and each logical record is 80 characters long. The offset field preceding the data portion of each block is 10 characters long. Note that, in this example, the BFACTOR option is not specified so it defaults to a value of 1. However, because you specify OFFSET=(10), each physical block in this tape file will be 90 characters long.

```
> DECLARE_BLUESKY_TAPE=(1),FILENO=(12),LRECL=(80),
OFFSET=(10)
>
```

Note

If you specify LABELS=(ANSI) and LEVEL=(2), the value of OFFSET is stored in the Buffer Offset field of HDR2, EOF2, and EOV2 labels.

1.7.1.9 The SEQUENCE Option:

This option specifies the order of a file within a set of files created simultaneously. This information corresponds to the ANSI file sequence number and the IBM data set sequence number, both of which are found on the HDR1 label.

SEQUENCE is a tape option for labelled tape files. You use SEQUENCE only when reading from these files. The value you specify is a four-digit number, 0001 through 9999. The default value is 0001. In the following example you declare a tape object, OURS, located on logical device number 2. Each logical record is 100 characters long. The tape contains ANSI standard labels, and you are working with the labelled file with the sequence number 0023.

```
> DECLARE OURS TAPE=(2),LRECL=(100),LABELS=(ANSI),
SEQUENCE=(0023)
>
```

1.7.1.10 The CHARACTERS Option:

You use CHARACTERS when reading or writing odd-length physical blocks. All normal data transfers to or from a tape are performed in terms of words. Since a word is composed of two bytes (or characters), only an even number of characters can normally be transferred between memory and a tape. Note that an odd number of words equals an even number of characters (five words = 10 characters). To write a true odd-length physical block, the tape drive must be instructed to transfer only one character per word. To get a one-character-per-word string, a two-character-per-word string is doubled in length and each character from the original string is placed in the right-hand byte of a word. This accounts for the restrictions on the LRECL and BFACTOR options. (For more information, see the descriptions of LRECL and BFACTOR earlier in this section.)

For output, when writing a tape file, CHARACTERS causes each block to be exploded from a two-character-per-word string to a one-character-per-word string. For input, when reading a tape file, each physical block is read as a one-character-per-block record and then imploded to a two-character-per-word string.

Note

If the CHARACTERS and EXCHANGE options are used together on input, CHARACTERS is processed first. On output EXCHANGE is processed first.

CHARACTERS is strictly for tape objects. The value you specify is either 1 or 2. The default value is 2. Note that when you specify a value of 1, the value of LRECL * BFACTOR must be less than or equal to 6,143. In the following example, LEDGER is a tape object located on logical device number 4. The logical record length is 37 and the blocking factor is 11. You are working with the 20th file on the tape. Each physical block is to be written as a true odd-length physical block.

```
> DECLARE LEDGER TAPE=(4),LRECL=(37),BFACTOR=(11)
> MODIFY LEDGER CHARACTERS=(1), FILENO=(20)
>
```

1.7.1.11 The EXCHANGE Option:

Certain operating systems of other manufacturers, when reading from or writing to tape, switch the high- and low-order bytes of each 16-bit word in a physical block. The EXCHANGE option allows you to compensate for this. On output, EXCHANGE causes pairs of bytes in a physical block to be swapped. On input, EXCHANGE causes each byte in a physical block to be swapped with its neighbor before any other processing takes place.

Note

If the physical block has an odd-length, the last byte is not swapped.

EXCHANGE is a tape option only. The value you specify is either PHYSICAL or NONE. The default value is NO. In the following example you declare a tape object, JJR, located on logical device number 6. The logical record length is 20 and you are working with the first file on the tape. All bytes in each word of the physical block are to be swapped since you specified PHYSICAL as the value for EXCHANGE.

```
> DECLARE_JJR_TAPE=(6),LRECL=(20),FILENO=(1)
> MODIFY_JJR_EXCHANGE=(PHYSICAL
>
```

1.7.1.12 The MAXIO Option:

This option specifies the maximum number of characters that the tape drive will attempt to read. All tape read operations request that the tape drive read and fill a buffer of a certain size. On some Prime systems, this number varies because of I/O window sizes.

This option is for tape objects only. The value is a number between 2,000 and 12,288. The default value is 10,000 characters (5,000 words). In the following example you modify a previously declared tape object, JJR, with the MAXIO option. You specify a value of 9,500. This means that the maximum number of characters requested on input is 9,500.

```
> MODIFY_JJR_MAXIO=(9500)
>
```

1.7.1.13 The CONTROL Option:

This option specifies the type of carriage-control used by a spool file. The choices are either NONE or FORTRAN. NONE is the default. This option is valid for spool objects only. In the following example, the spool object PURPLE is declared with carriage control type FORTRAN.

```
> DECLARE PURPLE SPOOL=(COLORS), CONTROL=(FORTRAN)
>
```

1.7.1.14 The DEFER Option:

This option specifies the time to print a spool file. This option is useful if you wish to avoid printing a spool file at a busy period. The time must be provided in 24-hour format (i.e. 1:00 p.m. = 23:00 hours). The next example demonstrates a spool file which will be printed at 10:15 p.m.

```
> MODIFY PURPLE  DEFER=(2215)
>
```

1.7.1.15 The LINENOS Option:

This option allows you to direct the spooler to print line numbers for every line printed in a spool file. This is strictly an option for spool objects only and may take the values YES or NO (the default is NO). In the following example, a spool object, MIGRAINE, is declared with line numbers:

```
> DECLARE MIGRAINE SPOOL=(HEADACHE), LINENOS=(YES)
>
```

1.7.2 The SAVE Subcommand

You use SAVE to save a group of options in a PRIMOS global variable. MAGNET saves only those options that you declared or modified to a value other than their default. The subcommand line format is as follows:

```
> SAVE object-name EXTERNAL=(global-variable-name)
```

SAVE is analogous to:

```
> DECLARE...EXTERNAL=(...)
```

You must always specify the EXTERNAL option on the SAVE subcommand line. The value of EXTERNAL is a PRIMOS global variable name. (For more information, see the description of EXTERNAL earlier in this section. For additional information on PRIMOS global variables, refer to The CPL User's Guide.) In the following example, all of the options previously specified for JJR are stored, along with their values, in the global variable name .SUE:

```
> SAVE JJR EXTERNAL=(.SUE)
>
```

1.7.3 The LOAD Subcommand

You use this subcommand to load a user-defined translation table. The internal names of these tables are "V", "W", "X", "Y", and "Z". MAGNET access these tables when you specify them as edit tokens in a TRANSLATE subcommand. (For more information, see the description of the TRANSLATE subcommand earlier in this section, and also Appendix A.) Each user-defined translation table is 512 bytes (256 words) long. The first 256 bytes are used when you are reading from a tape; the last 256 bytes are used when writing to a tape.

You must specify three options on the LOAD subcommand line. They are TYPE, LINES, and DISK. You may specify them in any order. The DISK option identifies the PRIMOS filename or pathname that contains your translation table. LINES and TYPE are described later in this section. The LOAD subcommand line format is as follows:

```
                          BIN
                          CHAR
> LOAD table-name TYPE=(OCT), LINES=(n), DISK=(pathname)
                          DEC
                          HEX
```

1.7.3.1 The TYPE Option:

TYPE specifies the interpretation of the characters in the disk file you name as the value for DISK in the LOAD subcommand line. You use TYPE only with the LOAD subcommand, and you must always specify it. There is no default value; you must specify one of the following five values:

- BIN (binary number character strings)
- CHAR (ASCII character strings)
- OCT (octal number character strings)
- DEC (decimal number character strings)
- HEX (hexadecimal number character strings)

All binary digit strings must be eight digits (or characters) long. All octal and decimal digit strings must be three digits (or characters) long. All hexadecimal digit strings must be two digits (or characters) long.

1.7.3.2 The LINES Option:

This option tells MAGNET how many lines to read in the disk file you name as the value for DISK in the LOAD subcommand line. You use LINES only with the LOAD subcommand, and you must always specify it. The value is a number that is a divisor of 512, namely 1, 2, 4, 8, 16, 32, 64, 128, 256, or 512. There is no default value. The number of binary, octal, decimal, or hexadecimal numbers (or characters) that must appear on each line is 512 divided by the value of the LINES option.

Note

For readability, it is not recommended that a small line count (1, 2, 4, 8, or 16) be used, especially if you specify TYPE=(BIN).

1.7.3.3 Creating Translation Tables:

You use a translation table to:

- Convert from one character set to another
- Convert from uppercase to lowercase and vice versa
- Set or reset certain bits in every byte of a character string

An EBCDIC to ASCII conversion table is an example of a tool you can use to convert from one character set to another. (For more information, see Appendix A.) An example of case conversion can be found in the TRANSLATE function of PL/I:

```
STRING = TRANSLATE ( STRING,
                     'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
                     'abcdefghijklmnopqrstuvwxyz');
```

The final function of a translation table, that of setting bits, is found in the industry-compatible ASCII to Prime ASCII translation table. (See Appendix A.)

The syntax you must use for creating your own translation tables is extremely rigid and precise. You have a choice of representing your table in ASCII character strings or binary, octal, decimal, or hexadecimal number character strings.

The ASCII character string representation is useful for case conversion and special symbol translation. With this format, you place the same number (n) of characters on 512 divided by n lines. Every incoming character to be translated actually represents an eight-bit number between 0 and 255. This number is used as an index into the table to find the output character (the translated character). For example, a translation table to convert from lowercase to uppercase would have the letters "A" through "Z" in absolute table positions starting from 0, represented by 225 through 250 decimal.

Each translation table must be 512 bytes long. The first 256 characters form an input table for use when reading from a tape. The second set of 256 characters forms an output table for use when writing to a tape. In the previous case of the lower to uppercase conversion table, it probably makes no sense to have an output table, yet you must include one. A solution is to include just a number of blank lines. Each sequence of characters on every line must begin in column 1.

Note

It is important to keep in mind that blanks are significant. They are treated as characters.

The binary number character string representation is useful when you translate characters that cannot be printed (control characters, for example). Each sequence of binary number character strings must begin in column 1. Each binary number may include the ASCII characters 0 and 1 only. Each number must be exactly eight characters long and must be separated from its neighbor by exactly one space.

The octal, decimal, and hexadecimal number character string representations are exactly the same as the binary representation except:

- For octal numbers, the characters "0" through "7" are allowed
- For decimal numbers, the octal numbers plus "8" and "9" are allowed
- For hexadecimal numbers, the decimal numbers plus "A" through "F" and "a" through "f" are allowed

These representations are also useful when you translate characters that cannot be printed, and provide a more compact notation than binary. As in the binary representation, all numbers must begin in column 1 and must be separated from their neighbors by only one space. As an example, consider the BCD to Prime ASCII and the Prime ASCII to BCD translation tables in Appendix A. The table includes binary, octal, decimal and hexadecimal representations which are not in the form necessary for input to MAGNET. Every two positions have been paired to form 16-bit numbers. In addition, since BCD is a 64-character code, the remaining 192 characters should be written as 0's for the binary, octal, decimal, or hexadecimal representations, or as blanks for the character representation. At the bottom of the BCD to Prime ASCII table, it is noted that the last line occurs 96 times. Similar repetitions are found in the Prime ASCII to BCD table.

Suppose that the BCD-ASCII table was found in a disk file, BCD_BIN.TABLE. Also suppose that this table consists of binary digit strings, eight per line (64 lines). The LOAD subcommand line to load this file (containing the table) into internal table "V" (edit token "V") is:

```
> LOAD_V_DISK=(BCD_BIN.TABLE),_TYPE=(BIN),_LINES=(64)
>
```

1.7.4 The CLOSE Subcommand

You use this subcommand to close a tape, disk or spool object which may have been left open by an abnormal condition or by your depressing the break key. For tape objects, an internal MAGNET switch signifying that the tape is open, is reset. For disk and spool objects, the respective files are truncated (if open for write) and are then physicaly closed. In the following example, a MOVE subcommand was issued to move data between a tape file, TFILE, and two other objects, DFILE, a disk file and SFILE, a spool file. Assuming that an abnormal end to the MOVE subcommand occurs, the CLOSE subcommand can be issued three times to close each of the three objects.

```
> MOVE TFILE  SFILE  DFILE
                .
                .
                .
> LIST ITEMS=(OPEN)
          object-names          type     open status
TFILE                           Tape     Open / Read
DFILE                           Disk     Open / Write
SFILE                           Spool    Open / Write
> CLOSE TFILE
> CLOSE SFILE
> CLOSE DFILE
> LIST
          object-names          type     open status
TFILE                           Tape     Closed
DFILE                           Disk     Closed
SFILE                           Spool    Closed
```

1.7.5 The SILENT and NOISY Subcommands

The two subcommands, SILENT and NOISY, control the printing of severity 1 messages at your console. Issuing the SILENT subcommand is equivalent to specifying "-SILENT" on the Primos command line used to invoke MAGNET (see Appendix B - COMMAND LINE OPTIONS AND BATCH JOBS). The NOISY subcommand enables the printing of severity 1 messages; this is the default mode for MAGNET if "-SILENT" is not specified on the Primos command line.

1.8 THE BREAK KEY

MAGNET contains an internal break key (CONTROL-P) handler. If you type the break key at any time you are within MAGNET, control returns to MAGNET subcommand level where you can enter more commands. To exit from MAGNET, use the QUIT subcommand, described earlier.

1.9 SAMPLE MAGNET SESSION

The following is a sample MAGNET terminal session. Each step of the session is explained in procedural format following the example.

```
OK, MAGNET
[MAGNET, Rev. 19.0]
> DECLARE_SYSIN_DISK=(INTERIOR>NATL-PK>YOSEMITE),LRECL=(128)
> MODIFY_SYSIN_FORMAT=(FIXED)
> /*
> DECLARE_SYSOUT1_TAPE=(3),DENSITY=(1600),FILENO=(4)
> MODIFY_SYSOUT1_LRECL=(141), BFACTOR=(11)
> MODIFY_SYSOUT1_FORMAT=(FIXED),BUFFERS=(3)
> MODIFY_SYSOUT1_CHARACTERS=(1), EXCHANGE=(YES)
> MODIFY_SYSOUT1_POSTACTION=(UNLOAD)
> TRANSLATE_SYSOUT1_(A120,10(F*),04,3(F),A4)
> /*
> DECLARE_SYSOUT2_LIKE=(SYSOUT1), FILENO=(1)
> MODIFY_SYSOUT2_VISUAL=(NPCA0001), BUFFERS=(0)
> /*
> MOVE_SYSIN_SYSOUT1
> /*
> DECLARE_AUDIT_TAPE_1_TAPE=(0),FILENO=(1),RECORDNO=(1)
> DECLARE_AUDIT_TAPE_2_LIKE=(AUDIT_TAPE_1), TAPE=(1)
> /*
> COPY_AUDIT_TAPE_1__AUDIT_TAPE_2_AMOUNT=(274)
> /*
> POSITION_AUDIT_TAPE_1_MODE=(ABSOLUTE)
> POSITION_AUDIT_TAPE_2_MODE=(ABSOLUTE)
> /*
> QUIT
OK,
```

1.9.1 Steps of the Session

1. First, declare SYSIN to be a disk object. The pathname of SYSIN is DC>INTERIOR>NATL_PARK>YOSEMITE. By default, each logical record in this file is variable-length (VAR/PRIME). The maximum length of each logical record is 128 characters.

2. With the MODIFY subcommand, you change the format of this disk file to fixed-length records. Each logical record is now defined to be exactly 128 characters.

3. A comment subcommand (/*) is provided for readability.

4. You declare SYSOUT1 to be a tape object, residing on logical device number 3. It has a density of 1600 bpi. Each logical record contains 141 characters. This tape file is the fourth file on the tape.

5. The next two MODIFY subcommands set the blocking factor to 12 and the record format to FIXED. Three I/O buffers are being used. After use, the tape will be unloaded (dismounted). Because 11*141 equals 1551, an odd number, an extra null character would ordinarily be inserted by Prime's hardware. To circumvent this problem, you specify CHARACTERS=(1) for true odd-length records. Also, this tape file will be transferred to a computer which swaps high- and low-order bytes. Thus, you specify EXCHANGE=(YES).

6. The desired translation is 120 industry-compatible ASCII characters, 10 asterisks, 4 binary characters, 3 spaces, and 4 industry-compatible ASCII characters.

7. You declare SYSOUT2 to be exactly the same as SYSOUT1 except for the BUFFERS, VISUAL, FILENO and NEXTCHAIN options. Since SYSOUT1 and SYSOUT2 form a chain of tapes, the NEXTCHAIN option of SYSOUT1 points to SYSOUT2. If SYSOUT1 should reach end-of-tape, what was the fourth file of SYSOUT1 will become file 1 on SYSOUT2. In addition, to alert the operator that SYSOUT2 is an "extension" to SYSOUT1, a visual identification is provided so that the second tape can be ready if the first tape becomes full. Finally, only the first object in a chain should have any I/O buffers because these buffers are "transferred" to SYSOUT2 if SYSOUT1 reaches EOT.

8. The information in the disk object SYSIN is moved to the destination object SYSOUT1. Should end-of-tape occur on SYSOUT1, messages are sent to the operator requesting that the first tape be dismounted and the second tape

(with external visual identification NPCA0001) be mounted. The operator receives these messages because the default communications mode is -OPERATOR on the PRIMOS command line.

9. You declare two new objects, AUDIT_TAPE_1 and AUDIT_TAPE_2, as tape objects on logical devices 0 and 1, respectively. You have specified both FILENO and RECORDNO values for these two objects.

10. The first 274 files on AUDIT_TAPE_1 are copied to AUDIT_TAPE_2.

11. Both AUDIT_TAPE_1 and AUDIT_TAPE_2 are positioned according to the values you specified for each object's FILENO and RECORDNO options. They are positioned absolutely, which in this particular example causes the two tapes to be rewound.

12. A QUIT subcommand ends your MAGNET session and returns you to PRIMOS command level. QUIT causes all dynamic space occupied by objects and I/O buffers to be returned to the free memory pool.

1.10 MAGNET MESSAGES

There are four types of messages generated by MAGNET that you may see when you work with the subsystem. They are:

- o Severity 0 - This identifies a message. No error condition has occurred. These messages generally signal an operation complete or the detection of the break key typed.

- o Severity 1 - This identifies a warning. Whatever operation is in progress at the time you receive the warning continues. This means that you remain in MAGNET and sometimes, depending on what happened, you remain within the subcommand.

- o Severity 2 - This identifies an uncorrectable error that occurred within a subcommand. You are returned to MAGNET at subcommand level.

- o Severity 3 - This identifies a fatal error. You are returned to PRIMOS.

All MAGNET messages are in this format (nnn is the message identification number):

```
** Message nnn, Severity = n  [, Object-name = name ]
   message to be printed
```

The following is a listing of all MAGNET messages. A suggested user response has been provided for each message. In addition, a description has been provided for each message that is not self-explanatory.

** Message 001, Severity = 3:
Invalid PRIMOS command line.

Suggested User Response: Retype a correct PRIMOS command line.

** Message 002, Severity = 2:
Error while reading subcommand.

Description: This indicates a possible terminal I/O error.

Suggested User Response: Retype the subcommand line. If the problem persists, contact your System Administrator.

** Message 003, Severity = 2:

Invalid subcommand (ignored).

Description: You typed an invalid subcommand line.

Suggested User Response: Retype the subcommand line. If the problem persists, contact your System Administrator.

** Message 004, Severity = 3:
A subcommand has aborted.

Description: A MAGNET subcommand was unable to perform the operation you requested. Normally, you receive a message before this one that indicates the specific type of problem.

Suggested User Response: You must reenter MAGNET and try your MAGNET operation(s) again. If the problem persists, contact your System Administrator.

** Message 005, Severity = 1:
Warning: additional errors may occur.

** Message 006, Severity = 2:
Invalid character(s) in object-name.

Suggested User Response: Retype the subcommand line using valid characters in your object-name(s).

** Message 007, Severity = 2:
An object-name is too long or too short.

Suggested User Response: Retype the subcommand line, making sure that any object-name(s) specified are no longer than 32 characters.

** Message 008, Severity = 2:
EXTERNAL option within global variable is invalid.

Description: The PRIMOS global variable named in the EXTERNAL option contains the word EXTERNAL. Only one level of EXTERNAL is allowed.

Suggested User Response: Correct your global variable so that the word EXTERNAL does not appear.

** Message 009, Severity = 2:
EXTERNAL variable name not found.

Description: The PRIMOS global variable named in the EXTERNAL option was not found.

Suggested User Response: Make sure that you have the correct global variable file activated. Next, make sure that the particular global variable exists in the file.

** Message 010, Severity = 3:
Abnormal halt. Contact your system administrator.

Description: An extremely unusual and uncorrectable condition occurred within MAGNET.

Suggested User Response: Save all pertinent information, such as output from COMOUTPUT, DMSTK, and PM commands. Give all of this information to your System Administrator.

** Message 011, Severity = 3:
Abnormal halt. Some files may be open.

Description: An extremely unusual and uncorrectable condition occurred within MAGNET.

Suggested User Response: Save all pertinent information such as output from COMOUTPUT, DMSTK, and PM commands. Give all of this information to your System Administrator. In addition, check for open disk files.

** Message 012, Severity = 2:
A tape is hardware write - protected.

Suggested User Response: Put a write-enable ring on your tape(s).

** Message 013, Severity = 2:
A tape is software write - protected.

Suggested User Response: Set the value of the PROTECT option for your tape(s) to NO.

** Message 014, Severity = 2:
A disk file is write - protected.

Suggested User Response: If you own the directory where the disk file(s) are located, change the protection attributes to enable write (W) access. If you are a non-owner, contact the owner or the System Administrator to have the protection attributes changed.

** Message 015, Severity = 2:
A disk file is read - protected.

Suggested User Response: If you own the directory where the disk file(s) are located, change the protection attributes to enable read (R) access. If you are a non-owner, contact the owner or the System Administrator to have the protection attributes changed.

** Message 016, Severity = 3:
A run-away tape condition exists.

Description: A read, forward-space file, or forward-space record operation is executing indefinitely, which eventually causes the tape to be pulled from the supply reel. This only occurs when your tape is blank (read, forward-space record), or when no more filemarkers are found on the tape (forward-space file).

Suggested User Response: Press the terminal BREAK key immediately. Unassign the tape drive(s) and immediately reassign them. This suddenly causes the tape drive(s) to halt, but you still need to reenter MAGNET to continue your magnetic tape operations.

** Message 017, Severity = 2:
A tape dismount (unload) operation has aborted.

Description: Either you or the operator typed REPLY -TAPE ABORT in response to a dismount request.

Suggested User Response: Depending on the circumstances, this may be an acceptable response.

** Message 018, Severity = 2:
A tape mount operation has aborted.

Description: Either you or the operator typed REPLY -TAPE ABORT in response to a mount request.

Suggested User Response: Depending on the circumstances, this may be an acceptable response.

** Message 019, Severity = 2:
An object was not previously declared.

Suggested User Response: Declare the object(s).

** Message 020, Severity = 2:
An object was already declared.

Suggested User Response: You cannot declare an object more than once. Use the MODIFY subcommand to change option values.

** Message 021, Severity = 2:
Memory allocation error: delete some objects.

Description: You have used all available memory for objects.

Suggested User Response: Either delete some objects or reduce the value of the BUFFERS option for declared tape objects.

** Message 022, Severity = 2:
Error while freeing dynamic memory.

Description: An error occurred when either some buffers (BUFFERS option), or the space occupied by a tape, spool, or disk object, was returned to the free storage area.

Suggested User Response: If this occurred during a DELETE operation, give a LIST subcommand to make sure that the object was deleted. If the object was not deleted, reissue the DELETE subcommand. If this error occurred during some other operation, try the operation again. If the problem persists, contact your System Administrator.

** Message 023, Severity = 2:
Uncorrectable tape read error.

Description: A tape READ operation was unsuccessful after 10 attempts.

Suggested User Response: Check for a correct density setting on the tape drive. If the density setting is correct, your tape is either bad or improperly formatted. There may also be a problem with either the tape drive or controller. If the problem persists, contact your System Administrator.

** Message 024, Severity = 2:
Uncorrectable tape write error.

Description: A tape WRITE operation was unsuccessful after 10 attempts.

Suggested User Response: Either your tape is bad or you are attempting to write records greater than the maximum I/O window size allowed on your system. There may also be a problem with either the tape drive or controller. If the problem persists, contact your System Administrator.

** Message 025, Severity = 2:
Uncorrectable tape control error.

Description: A tape control operation was unsuccessful.

Suggested User Response: Check for a correct density setting on the tape drive. If the density setting is correct, your tape is either bad or improperly formatted. There may also be a problem with either the tape drive or controller. If the problem persists, contact your System Administrator.

** Message 026, Severity = 2:
No room is available in the object list.

Description: You are attempting to declare more than 100 objects (the maximum).

Suggested User Response: Delete any objects you no longer need.

** Message 027, Severity = 2:
Error while setting up default options.

Suggested User Response: Reissue your DECLARE subcommand. If the problem persists, contact your System Administrator.

** Message 028, Severity = 2:
Error while parsing the option list.

Suggested User Response: Retype your DECLARE or MODIFY subcommand, correcting any errors that exist.

** Message 029, Severity = 1:
Option conflict (DENSITY & TRACKS?).

Description: A conflict exists among various options.

Suggested User Response: Check your options for compatibility and change option values with the MODIFY subcommand.

** Message 030, Severity = 3:
Subsystem aborting- Check for open files.

Description: A subcommand just issued has aborted. Normally, you should have received another message prior to this one indicating the specific error. MAGNET cannot recover from this error, and consequently has returned you to PRIMOS command level.

Suggested User Response: Check for open files. Attempt to correct any reported problems. Reenter MAGNET and issue your subcommand again. If the problem persists, contact your System Administrator.

** Message 031, Severity = 2:
The COPY subcommand will abort.

Suggested User Response: You should have received an message prior to this one indicating the specific error. Correct the problem and reissue the subcommand.

** Message 032, Severity = 2:
The DECLARE subcommand will abort.

See Message 031.

** Message 033, Severity = 2:
The DELETE subcommand will abort.

See Message 031.

** Message 034, Severity = 2:
Tape drive not assigned.

Suggested User Response: Assign all tape drive(s) at PRIMOS command level with the ASSIGN command. Then, reenter MAGNET and reissue the command(s).

** Message 035, Severity = 2:
The LOAD subcommand will abort.

See Message 031.

** Message 036, Severity = 2:
The MODIFY subcommand will abort.

See Message 031.

** Message 037, Severity = 2:
The MOVE subcommand will abort.

See Message 031.

** Message 038, Severity = 2:
The POSITION subcommand will abort.

See Message 031.

** Message 039, Severity = 3:
The QUIT subcommand will abort.

See Message 031.

** Message 040, Severity = 2:
The READ subcommand will abort.

See Message 031.

** Message 041, Severity = 2:
The SAVE subcommand will abort.

See Message 031.

** Message 042, Severity = 2:
The TRANSLATE subcommand will abort.

See Message 031.

** Message 043, Severity = 2:
The WRITE subcommand will abort.

See Message 031.

** Message 044, Severity = 2:
Error while analyzing the ACCESS option.

Description: The READ or MODIFY subcommand found an error when it analyzed this option.

Suggested User Response: Retype the subcommand line with this option corrected.

** Message 045, Severity = 2:
Error while analyzing the BFACTOR option.

See Message 044.

** Message 046, Severity = 2:
Error while analyzing the BUFFERS option.

See Message 044.

** Message 047, Severity = 2:
Error while analyzing the BYPASS option.

See Message 044.

** Message 048, Severity = 2:
Error while analyzing the CHARACTERS option.

See Message 044.

** Message 049, Severity = 2:
Error while analyzing the CREATE option.

See Message 044.

** Message 050, Severity = 2:
Error while analyzing the DENSITY option.

See Message 044.

** Message 051, Severity = 2:
Error while analyzing the DISK option.

See Message 044.

** Message 052, Severity = 2:
Error while analyzing the EXPIRE option.

See Message 044.

** Message 053, Severity = 2:
Error while analyzing the EXTERNAL option.

See Message 044.

** Message 054, Severity = 2:
Error while analyzing the FILEID option.

See Message 044.

** Message 055, Severity = 2:
Error while analyzing the FORMAT option.

See Message 044.

** Message 056 has not yet been assigned.

** Message 057, Severity = 2:
Error while analyzing the GENERATION option.

See Message 044.

** Message 058, Severity = 2:
Error while analyzing the LABELS option.

See Message 044.

** Message 059, Severity = 2:
Error while analyzing the LEVEL option.

See Message 044.

** Message 060, Severity = 2:
Error while analyzing the LRECL option.

See Message 044.

** Message 061, Severity = 2:
Error while analyzing the MAXIO option.

See Message 044.

** Message 062, Severity = 2:
Error while analyzing the NEXTCHAIN option.

See Message 044.

** Message 063, Severity = 2:
Error while analyzing the OWNER option.

See Message 044.

** Message 064, Severity = 2:
Error while analyzing the PARITY option.

See Message 044.

** Message 065, Severity = 2:
Error while analyzing the POSTACTION option.

See Message 044.

** Message 066, Severity = 2:
Error while analyzing the PREACTION option.

See Message 044.

** Message 067, Severity = 2:
Error while analyzing the PREVCHAIN option.

See Message 044.

** Message 068, Severity = 2:
Error while analyzing the PROTECT option.

See Message 044.

** Message 069, Severity = 2:
Error while analyzing the TAPE option.

See Message 044.

** Message 070, Severity = 2:

Error while analyzing the TRACKS option.

See Message 044.

** Message 071, Severity = 2:
Error while analyzing TRANSLATE token(s).

Description: The list of edit tokens in your TRANSLATE subcommand contains one or more errors. For example, you may have typed A$ instead of A4.

Suggested User Response: Retype the TRANSLATE subcommand with all errors corrected.

** Message 072, Severity = 2:
Error while analyzing the VERSION option.

See Message 044.

** Message 073, Severity = 2:
Error while analyzing the VISUAL option.

See Message 044.

** Message 074, Severity = 2:
Error while analyzing the VOLSER option.

See Message 044.

** Message 075, Severity = 2:
External names must begin with periods.

Suggested User Response: Retype the SAVE, DECLARE, or MODIFY subcommand with a correct PRIMOS global variable name as the value for the EXTERNAL option.

** Message 076, Severity = 2:
Invalid subcommand or characters.

Suggested User Response: Retype the subcommand line.

** Message 077, Severity = 2:
An error occurred while saving into a variable.

Description: An error occurred when the SAVE subcommand tried to save an object's options and values.

Suggested User Response: Make sure that your global variable file is activated and not full. Try the SAVE subcommand again. If the problem persists, contact your System Administrator.

** Message 078, Severity = 2:
No more room exists to declare this object.

Description: No more room is available in the free storage area to declare this object.

Suggested User Response: Delete some objects or modify the values of the BUFFERS option of any previously declared tape objects.

** Message 079, Severity = 3:
Subsystem aborting- no files are open.

Description: A subcommand just issued has aborted. Normally, you should have received another error message prior to this one indicating the specific error. MAGNET cannot recover from this error, and consequently has returned you to PRIMOS command level.

Suggested User Response: Attempt to correct any reported problems. Reenter MAGNET and try your subcommand again. If the problem persists, contact your System Administrator.

** Message 080, Severity = 1:
An invalid command line option will be ignored.

Suggested User Response: No corrective action is necessary. However, if you incorrectly typed -SILENT or -USER you should give the QUIT subcommand and reenter MAGNET with the correct MAGNET command line options.

** Message 081, Severity = 1:
Execution will continue.

** Message 082, Severity = 2:
Error while analyzing the FILENO option.

See Message 044.

** Message 083, Severity = 2:
The RENAME subcommand will abort.

See Message 031.

** Message 084, Severity = 1:
Circular chaining is not permitted.

Description: The values you specified for the PREVCHAIN and/or NEXTCHAIN options are the same as the object-name you are currently specifying in a DECLARE or MODIFY subcommand.

Suggested User Response: Execution continues, but the NEXTCHAIN and/or PREVCHAIN options are not set. (You have to modify them.)

** Message 085, Severity = 2:
The wrong tape is mounted.

Suggested User Response: Try the entire subcommand operation again with the correct tape mounted.

** Message 086, Severity = 1:
The ACCESS option is not initialized.

Suggested User Response: Give this option a value if you meant to. Otherwise, ignore this message.

** Message 087, Severity = 1:
The BFACTOR option is not initialized.

See Message 086.

** Message 088, Severity = 1:
The BUFFERS option is not initialized.

See Message 086.

** Message 089, Severity = 1:
The BYPASS option is not initialized.

See Message 086.

** Message 090, Severity = 1:
The CHARACTERS option is not initialized.

See Message 086.

** Message 091, Severity = 1:
The CREATE option is not initialized.

See Message 086.

** Message 092, Severity = 1:
The DENSITY option is not initialized.

See Message 086.

** Message 093, Severity = 1:
The DISK option is not initialized.

See Message 086.

** Message 094, Severity = 1:
The EXPIRE option is not initialized.

See Message 086.

** Message 095, Severity = 1:
The EXTERNAL option is not initialized.

See Message 086.

** Message 096, Severity = 1:
The FILEID option is not initialized.

See Message 086.

** Message 097, Severity = 1:
The FORMAT option is not initialized.

See Message 086.

** Message 098 has not yet been assigned.

** Message 099, Severity = 1:
The GENERATION option is not initialized.

See Message 086.

** Message 100, Severity = 1:
The LABELS option is not initialized.

See Message 086.

** Message 101, Severity = 1:
The LEVEL option is not initialized.

See Message 086.

** Message 102, Severity = 1:
The LRECL option is not initialized.

See Message 086.

** Message 103, Severity = 1:
The MAXIO option is not initialized.

See Message 086.

** Message 104, Severity = 1:
The NEXTCHAIN option is not initialized.

See Message 086.

** Message 105, Severity = 1:

The OWNER option is not initialized.

See Message 086.

** Message 106, Severity = 1:
The PARITY option is not initialized.

See Message 086.

** Message 107, Severity = 1:
The POSTACTION option is not initialized.

See Message 086.

** Message 108, Severity = 1:
The PREACTION option is not initialized.

See Message 086.

** Message 109, Severity = 1:
The PREVCHAIN option is not initialized.

See Message 086.

** Message 110, Severity = 1:
The PROTECT option is not initialized.

See Message 086.

** Message 111, Severity = 1:
The TAPE option is not initialized.

See Message 086.

** Message 112, Severity = 1:
The TRACKS option is not initialized.

See Message 086.

** Message 113, Severity = 1:
TRANSLATION has not been specified.

Suggested User Response: Specify a translation (TRANSLATE subcommand) if you meant to. Otherwise, ignore this message.

** Message 114, Severity = 1:
The VERSION option is not initialized.

See Message 086.

** Message 115, Severity = 1:
The VISUAL option is not initialized.

See Message 086.

** Message 116, Severity = 1:
The VOLSER option is not initialized.

See Message 086.

** Message 117, Severity = 1:
One or more options will be ignored.

Suggested User Response: Use the DISPLAY subcommand to check that all option values are correct. Then, use the MODIFY subcommand to modify option values if necessary.

** Message 118, Severity = 2:
Error while analyzing the RECORDNO option.

See Message 044.

** Message 119, Severity = 2:
Error while analyzing the OFFSET option.

See Message 044.

** Message 120, Severity = 1:
The OFFSET option is not initialized.

See Message 086.

** Message 121, Severity = 2:
First option not DISK/TAPE/SPOOL, EXTERNAL or LIKE.

Suggested User Response: Retype the DECLARE subcommand line with DISK, TAPE, or EXTERNAL as the first option specified.

** Message 122, Severity = 2:
Disk options are invalid for non-disk objects.

Suggested User Response: Retype the subcommand line with valid tape or spool options.

** Message 123, Severity = 2:
Tape options are invalid for non-tape objects.

Suggested User Response: Retype the subcommand line with valid disk or spool options.

** Message 124, Severity = 2:
Error while analyzing the AMOUNT option.

See Message 044.

** Message 125, Severity = 1:
The AMOUNT option is not initialized.

See Message 086.

** Message 126, Severity = 2:
No objects have been declared.

Suggested User Response: You must first declare all objects before using them.

** Message 127, Severity = 1:
All options following EXTERNAL are ignored.

Suggested User Response: Since all options that follow the EXTERNAL option in your DECLARE or MODIFY subcommand have been ignored, use the DISPLAY subcommand to check that all option values are correct. If necessary, use the MODIFY subcommand to change option values.

** Message 128, Severity = 2:
Options must be specified.

Description: You typed a DECLARE or MODIFY subcommand without specifying any options.

Suggested User Response: Retype the subcommand line specifying options.

** Message 129, Severity = 2:
This object has too many options to save.

Description: The object you specified in the SAVE subcommand contains option values which exceed 1,024 characters (the maximum allowed for a PRIMOS global variable).

Suggested User Response: Set a few options back to their default values, and try to issue the SAVE subcommand again.

** Message 130, Severity = 2:
Global variable file bad or uninitialized.

Description: This message usually indicates that you do not have a global variable file activated.

Suggested User Response: Activate your global variable file at PRIMOS level with the DEFINE_GVAR command. If the problem persists, your global variable file contains bad data and must be re-created.

** Message 131, Severity = 2:
Global variable file is too small or full.

Description: There is not enough room left in your global variable file to save an object with its option values.

Suggested User Response: To make room in your global variable file, use the PRIMOS command DELETE_VAR to delete some variables.

** Message 132, Severity = 2:
An invalid translation table was specified.

Suggested User Response: Retype the LOAD subcommand, specifying V, W, X, Y, or Z as the user translation table.

** Message 133, Severity = 2:
Error while analyzing the LINES option.

See Message 044.

** Message 134, Severity = 2:
Error while analyzing the TYPE option.

See Message 044.

** Message 135, Severity = 1:
The LINES option is not initialized.

See Message 086.

** Message 136, Severity = 1:
The TYPE option is not initialized.

See Message 086.

** Message 137, Severity = 2:
Error while opening a disk file.

Suggested User Response: Do one of the following:

- If you are trying to read from a disk file, check that the file exists and that the protection and/or access control for this file is correctly set.
- If you are trying to write to a disk file, check for proper protection and/or access control. Check that the disk is not full.

** Message 138, Severity = 2:
Error while reading from a disk file.

Suggested User Response: Check that the protection and/or access control for this disk file is correctly set.

** Message 139, Severity = 2:
Data conversion error.

Description: Bad character(s) were detected while reading from a disk file into a user translation table.

Suggested User Response: Check your disk file for valid characters. Binary numbers may contain the characters 0 or 1 only. Octal numbers may contain the characters 0 through 7 only. Decimal numbers may contain the characters 0 through 9 only. Hexadecimal numbers may contain the characters 0 through 9 and A through F only. Check for proper spacing of these numbers. Blanks can be considered part of a numeric field and will cause this error to occur.

** Message 140, Severity = 2:
Error while closing a disk file.

Suggested User Response: Use the PRIMOS CLOSE command to close the disk file.

** Message 141, Severity = 2:
The DISPLAY subcommand will abort.

See Message 031.

** Message 142, Severity = 2:
The LIST subcommand will abort.

See Message 031.

** Message 143, Severity = 2:
A tape drive is not ready and/or online.

Suggested User Response: Make sure that the tape drive is powered-on, a tape has been mounted, and that the online button has been depressed.

** Message 144, Severity = 2:
File not found.

Suggested User Response: If you are working with a tape object, try rewinding the tape first. If this is a disk object, use the MODIFY subcommand to change the name of the specified disk file.

** Message 145, Severity = 2:
Specified record not found.

Description: The POSITON subcommand was unable to position to the record number you specified.

Suggested User Response: This may be an acceptable condition. If not, change the value of the RECORDNO option with the MODIFY subcommand and reissue the subcommand.

** Message 146, Severity = 2:
A positioning sub-operation has aborted.

Description: An error occurred while a tape was being positioned to its correct file and/or record location.

Suggested User Response: Check that the correct tape is mounted and that the correct density is set. You may try rewinding the tape first before you reissue the subcommand. If the problem persists, it usually indicates that your tape is bad.

** Message 147, Severity = 2:
One or more objects are not of type TAPE.

Suggested User Response: Reissue the subcommand, making sure that at least one of the objects you specified is a tape object. For the COPY subcommand, all objects must be tape objects. For the POSITION subcommand, the one object you specify must be a tape object. For the READ subcommand, the first object must be a tape object. For the WRITE subcommand, the second object must be a tape object.

** Message 148, Severity = 2:
One or more objects are not of type DISK.

Suggested User Response: Reissue the subcommand, making sure that at least one of the objects you specified is a disk object. For the READ subcommand, the second object you specify must be a disk object. For the WRITE subcommand, the first object you specify must be a disk object.

** Message 149, Severity = 2:
Error while analyzing the EXCHANGE option.

See Message 044.

** Message 150, Severity = 2:
Invalid operation for a disk or spool object.

Suggested User Response: Reissue the COPY or POSITION subcommand, specifying tape objects only.

** Message 151, Severity = 2:
Invalid operation for a tape object.

Suggested User Response: Reissue the READ or WRITE subcommand with disk and tape objects in the correct order.

** Message 152, Severity = 2:
An I/O error has occurred in the source object.

Description: This message may indicate one of the following problems:

- A tape drive is not assigned.
- A tape drive is not connected.
- An invalid command was sent to the tape drive.
- The source object in the READ, WRITE, or MOVE subcommand you just issued was left open due to a previous error.
- An uncorrectable error occurred while reading a record from the source object.

Suggested User Response: Make sure you have assigned all tape drives you plan to use. Make sure that any tape drives you have assigned are valid. (If your system contains only one tape drive and it is connected to the first controller, that drive can only be identified by 0, 1, 2, or 3.) Make sure that the density switches on all drives are set correctly. Make sure that all disk files are closed before you invoke MAGNET. If the error persists, it may indicate that your tape contains bad data.

** Message 153, Severity = 2:
An I/O error has occurred in a destination object.

Description: This message may indicate one of the following problems:

- A tape drive is not assigned.
- A tape drive is not connected.
- An invalid command was sent to the tape drive.
- A destination object in the READ, WRITE, or MOVE subcommand you just issued was left open due to a previous error.
- An uncorrectable error occurred while writing a record to a destination object.

Suggested User Response: Make sure you have assigned all tape drives you plan to use. Make sure that any tape drives you have assigned are valid. (If your system contains only one tape drive and it is connected to the first controller, that drive can only be identified by 0, 1, 2, or 3.) Make sure that the density switches on all drives are set correctly. Make sure that all disk files are closed before you invoke MAGNET. If the error persists, it may indicate that a tape is dirty or creased.

** Message 154, Severity = 2:
Invalid number of files to copy.

Suggested User Response: Reissue the COPY subcommand specifying a valid number of files to copy.

** Message 155, Severity = 1:
A labelled tape is being used in a COPY operation.

Suggested User Response: This is a warning only and may be an acceptable message. Note that it is unacceptable for the VOL1 label to be copied to any location other than the first record on a tape.

** Message 156, Severity = 2:
Multiple objects specify the same tape.

Description: In a COPY or MOVE subcommand two or more objects specify the same tape.

Suggested User Response: Use the MODIFY subcommand to change the

value of the TAPE option. Then, reissue the MOVE or COPY subcommand.

** Message 157, Severity = 2:
Not enough objects have been specified.

Suggested User Response: Reissue the READ, WRITE, MOVE, or COPY subcommand and specify at least two object-names.

** Message 158, Severity = 1:
Up to 8 object-names may be specified.

Suggested User Response: This is a warning only. You may specify no more than eight object-names on the MOVE or COPY subcommand lines. Any other object-names are ignored.

** Message 159, Severity = 2:
Same object-name appears more than once.

Suggested User Response: Reissue the READ, WRITE, MOVE, or COPY subcommand so that object-names on the subcommand line appear only once.

** Message 160, Severity = 2:
Tape objects require at least one buffer.

Description: You have not specified a BUFFERS option for one or more tape objects. Since the default value for this option is 0, the READ, WRITE, or MOVE subcommand could not be executed.

** Message 161, Severity = 2:
Error while analyzing the SEQUENCE option.

See Message 044.

** Message 162, Severity = 2:
Error while analyzing the PRINT option.

See Message 044.

** Message 163, Severity = 2:

Error while chaining tapes.

Description: You had specified a NEXTCHAIN option value for a tape object. However, when it became necessary to change tapes, the next object could not be found.

Suggested User Response: Use the DECLARE subcommand to declare all objects in a chain. Reissue the READ, WRITE, or MOVE subcommand.

** Message 164, Severity = 2:
Object-name specified by LIKE option points to self.

Description: The name specified as the value for the LIKE option is the object-name itself.

Suggested User Response: Retype the DECLARE or MODIFY command using a different name with the LIKE option.

** Message 165, Severity = 2:
Error while analyzing the LIKE option.

See Message 044.

** Message 166, Severity = 2:
Object-name specified by LIKE option not declared.

Description: The name specified as the value for the LIKE option was not previously declared.

Suggested User Response: Re-type the command specifying as a value for the LIKE option the name of another object that was previously declared.

** Message 167, Severity = 2:
The CLOSE subcommand will abort.

See Message 031.

** Message 168, Severity = 2:
Spool options are illegal for non-spool objects.

Description: You specified a spool option while you were attempting to declare or modify a tape or disk object.

Suggested User Response: Re-type the command specifying only disk options for disk objects and tape options for tape objects.

** Message 169, Severity = 2:
Error while analyzing the SPOOL option.

See Message 044.

** Message 170, Severity = 2:
Error while analyzing the FORM option.

See Message 044.

** Message 171, Severity = 2:
Error while analyzing the AT option.

See Message 044.

** Message 172, Severity = 2:
Error while analyzing the COPIES option.

See Message 044.

** Message 173, Severity = 2:
Error while analyzing the CONTROL option.

See Message 044.

** Message 174, Severity = 2:
Error while analyzing the LINENOS option.

See Message 044.

** Message 175, Severity = 2:
Error while analyzing the DEFER option.

See Message 044

** Message 176, Severity = 2:
The SPOOL option value for this object is bad.

Description: The printing of this error message may indicate an internal, severe, MAGNET error condition or a difficulty with the spool subsystem.

Suggested User Response: Attempt to DELETE an objects you were using. Then, try to re-declare these same objects and re-issue any command(s) that you typed prior to receiving this message. If you receive this message again or encounter any difficulties while using the DELETE and/or DECLARE commands, exit MAGNET and report the problem to your system administrator.

** Message 177, Severity = 2:
A spool object cannot be an information source.

Description: You typed a MOVE subcommand and specified the source object as a spool object. Spool objects can only be used for output.

Suggested User Response: Re-type the MOVE subcommand, specifying only a disk or a tape object as the source.

** Message 178, Severity = 0:
QUIT. To exit from MAGNET, use the "QUIT" command.

Description: You typed CONTROL-P (the "BREAK" key). This is not an error condition. If any operation was in progress, for example a MOVE subcommand, it is not restartable. That is, any operation that is interrupted must be re-issued again.

Suggested User Response: Type LIST ITEMS=(OPEN) to obtain a list of any object(s) that have been left open. If you were using the old (pre-rev 18.4) versions of the READ, WRITE or COPY subcommands, you may find "dummy" objects called $DUMMY1 and $DUMMY2. These objects will be closed and deleted automatically whenever you issue another (old version) READ, WRITE, COPY or position subcommand. These objects should be closed, however, (use the CLOSE subcommand) prior to exiting from MAGNET. If you interrupted ANY subcommand, you MUST re-issue the command again from the beginning; there is NO restart capability.

** Message 179, Severity = 2:
Error while analyzing the ITEMS option.

See Message 044.

** Message 180, Severity = 2:
Error while closing a spool file.

Suggested User Response: This is not usually a severe difficulty because, at this point, any information that you wanted to move to the spool subsystem has been moved. You can either attempt to use the CLOSE subcommand within MAGNET or, after typing QUIT and returning to Primos, issue a Primos CLOSE command.

** Message 181, Severity = 2:
Error while opening a tape file.

Description: MAGNET could not open a tape file for any of several reasons: your tape drive may be offline or not ready; an error may have occurred in the tape drive, controller or both; the tape may be bad and MAGNET could not read from or write to the tape even after ten attempts; the end-of-tape marker may have been detected; the specified FILEID option does not match the file identification fields of any HDR1 labels; the file number specified by the FILENO option is larger than the number of files on the tape.

Suggested User Response: Check the FILEID option value for correct letters, symbols and case. Try specifying a PREACTION=(REWIND) option in a MODIFY command. Re-issue the READ, WRITE or MOVE subcommand.

** Message 182, Severity = 2:
Error while opening a spool file.

See Message 137.

** Message 183, Severity = 2:
Error while closing a tape file.

Description: You may receive this message for any one of several reasons: the tape drive may have gone offline or become not ready; the end-of-tape marker may have been detected; an error may have occurred in the tape drive, controller or both; the tape may have a bad spot and MAGNET could not read from or write to the tape even after 10 attempts.

Suggested User Response: This is usually a severe problem. It may be necessary to rewind the tape and re-issue your WRITE or MOVE command. If, after re-writing to the tape you receive the same message and you notice that the tape has not reached the EOT marker, the tape probably contains a bad spot. If you had been reading or moving from the tape, this problem can be rectified by issuing a CLOSE command.

** Message 184, Severity 2:
Error while reading from a tape file.

Description: This message usually indicates a bad spot on the tape.

Suggested User Response: Retry the READ or MOVE operation. If the error message persists, the tape is bad and cannot be read.

** Message 185, Severity = 2:
Error while writing to a tape file.

Description: This message usually indicates a bad spot on the tape.

Suggested User Response: Retry the WRITE or MOVE operation. If the error message persists, the tape is bad and cannot be read.

** Message 186, Severity = 2:
Error while writing to a disk file.

Description: This message may indicate a number of problems: a full disk or a FAM or network difficulty.

Suggested User Response: Exit MAGNET, and check for a full disk. If that is not the problem and you are attempting to access a disk file through the FAM, check that the FAM is running and that all systems on a network are also running.

** Message 187, Severity = 2:
Error while writing to a spool file.

See Message 186.

** Message 188, Severity = 0:
Operation complete.

Description: This message indicates that an I/O operation (READ, WRITE POSITION, COPY or MOVE has successfully completed.

** Message 189, Severity = 0:
n Physical blocks read.

Description: This message should always be preceeded by message 188. "n" indicates the number of physical blocks read from a tape file.

** Message 190, Severity = 0:
n Logical records read.

Description: This message should always be preceeded by message 188. "n" indicates the nuber of logical records read from a tape or disk file.

** Message 191, Severity = 0:
n Physical blocks written.

Description: This message should always be preceeded by message 188. "n" is the number of physical blocks written to a tape.

** Message 192, Severity = 0:
n Logical records written.

Description: This message should always be preceeded by messaged 188. "n" is the number of logical records written to a tape or disk file.

** Message 193, Severity = 2:
Error while analyzing the MODE option.

See Message 044.

2 APPENDIX A

2.1 EDIT TOKENS, CHARACTER SET TABLES and TRANSLATION TABLES

2.1.1 Translation Edit Tokens

Edit tokens are part of the MAGNET TRANSLATE subcommand line. Each edit token specifies how you want particular fields of a logical record translated. All MAGNET edit tokens are described in detail in the following paragraphs. (For more information on the TRANSLATE subcommand and edit tokens, see Section 7.)

2.1.1.1 Edit Token A:

All characters specified with edit token A translate to or from industry-standard ASCII. When you are reading from a tape, this means translating from industry-standard ASCII to Prime ASCII. When you are writing to a tape, this means translating from Prime ASCII to industry-standard ASCII. For example, if you have 80-character industry-standard ASCII logical records on an input tape, you could specify the following:

```
(A80)
```

Note that you may also specify this translation as (80(A1)), but this format processes much more slowly.

Edit Token B

This edit token translates seven-track BCDIC code to Prime ACSII, and vice versa. For example, if you have declared a tape object named BANKST consisting of 120-character logical records, your TRANSLATE subcommand line may be:

```
> TRANSLATE BANKST (B120)
>
```

This causes each of the 120 characters in a logical record to be translated from seven-track BCDIC code to Prime ASCII if you are using BANKST as an input object. Again, note that you can also use the form (120(B1)), but it processes much more slowly.

2.1.1.2 Edit Token C:

This is the column position edit token. You use it to move to a field in a logical record. It causes MAGNET's internal pointer to position to a different location. For example, assuming input from a tape, the string:

(A80,C10,A10)

causes the following to occur:

- o character columns one through 80 are translated from industry-standard ASCII to Prime ASCII;
- o you are repositioned to column 10;
- o character columns 10 through 19 are translated from industry-standard ASCII to Prime ASCII.

In this example, output from the translation process is 90 characters even though the length of the input logical record is only 80 characters. Note that columns 10 through 19 are translated twice and will appear twice in the output. However, in the following example less characters will appear on output:

(A40,C71,A10)

Here, character columns 41 through 70 are not translated and are effectively deleted from the output.

2.1.1.3 Edit Token D:

This is the deletion edit token. It allows you to delete characters. For example, the string:

(A40,D30,A10)

performs the same task as the previous example. In other words, (A40,C71,A10) is equivalent to (A40,D30,A10).

2.1.1.4 Edit Token E:

This edit token translates EBCDIC code to Prime ASCII, and vice versa. For example, the string:

(E100)

causes 100 characters to be translated from EBCDIC to Prime

ASCII or vice versa.

2.1.1.5 Edit Token F:

This edit token allows you to specify fill characters. These characters may be any symbols except the following:

- o comma (,)
- o left and right parentheses ("(" and ")")
- o slash (/)
- o apostrophe (')

Note that alpha characters are always translated to upper case. For example, the string:

```
(A10,10(F*),C20,A20,10(FX))
```

causes the following to occur:

- o 10 characters are translated using A format;
- o 10 asterisks (*) are inserted into the output record;
- o you are positioned to column 20;
- o 20 characters (columns 20 through 39) are translated using A format;
- o 10 X's are inserted into the output record.

The output record contains 50 characters. If you specify this edit token by itself, without any fill character, a blank is inserted The string:

```
(20(F))
```

causes 20 blanks to be inserted.

2.1.1.6 Edit Tokens G through N:

These eight edit tokens specify seven-track packing and unpacking codes. (For more information, see Section 1.)

Edit Token	Packing Code
G	2424
H	4242
I	2466
J	4266
K	6246
L	6426
M	6624
N	6642

The repetition factor you specify for these edit tokens is a byte count. Thus, if you wish to pack, on input, three binary numbers in K format, you specify (K9). You use the number nine because each binary number on input occupies three bytes. On output, each binary number occupies only two bytes or one 16-bit word. Likewise, G and H format binary numbers occupy four bytes on input from a tape and occupy two bytes on output to disk. When you are writing to a tape, binary numbers are expanded from two bytes to four bytes for G and H formats, and to three bytes for I, J, K, L, M, and N formats. When you are determining logical record lengths for tape and disk objects, you must take into account seven-track binary number expansion and compression.

In the following example, 20 binary numbers are translated using H format:

```
(H80)
```

2.1.1.7 Edit Token O:

You use this edit token to specify no translation in your TRANSLATE subcommand line. For example, the subcommand line:

```
> TRANSLATE ACCOUNTS (O100)
>
```

causes each logical record in ACCOUNTS to be transferred

through the translation mechanism without any translation occurring (assuming that each logical record is 100 characters long).

2.1.1.8 Edit Token P:

This edit token enforces Prime ASCII input and output. In other words, the high-order bit of each byte is set on if it is not already. For example, the subcommand line:

```
> TRANSLATE ACCOUNTS (P100)
>
```

causes the high-order bit of each byte of each logical record to be set on.

2.1.1.9 Edit Tokens Q through U:

These edit tokens do not specify anything at this time. They are reserved for future use.

2.1.1.10 Edit Tokens V through Z:

These five edit tokens specify user-defined translation codes. (For more information, see Section 7.) You use these edit tokens the same way you use edit tokens A, B, E, O, and P. One character of input is translated according to the table you loaded with a LOAD subcommand and becomes one character of output.

2.1.1.11 Repetition Factor *:

If you wish to specify translation for the remaining characters of a logical record without specifying an absolute number, you may use the repetiton factor *. For example, the subcommand line:

```
> TRANSLATE WUMPUS (A*)
>
```

causes all the characters in each logical record of WUMPUS to be translated according to A format. The subcommand line:

```
> TRANSLATE GRINCH (E29,010,3(F),E*)
>
```

causes the following to occur:

- o 29 characters are translated according to E format;
- o 10 characters or bytes are not translated;
- o three blanks are inserted;
- o all remaining characters are translated according to E format.

2.1.2 Character Set Tables

2.1.2.1 Industry-Standard ASCII:

^ => Control key depressed

8-Bit Octal Code	Char		Code in Left Byte	8-Bit Octal Code	Char	Code in Left Byte
000	NUL		0000	040	Sp	0200
001	SOH	^A	0004	041	!	0204
002	STX	^B	0010	042	"	0210
003	ETX	^C	0014	043	#	0214
004	EOT	^D	0020	044	$	0220
005	ENQ	^E	0024	045		
006	ACK	^F	0030	046		0230
007	BEL	^G	0034	047	'	0234
010	BS	^H	0040	050	(	0240
011	HT	^I	0044	051	)	0244
012	NL	^J	0050	052	*	0250
013	VT	^K	0054	053	+	0254
014	FF	^L	0060	054	,	0260
015	CR	^M	0064	055	-	0264
016	RRS	^N	0070	056	.	0270
017	BRS	^O	0074	057	/	0274
020	RCP	^P	0100	060	0	0300
021	RHT	^Q	0104	061	1	0304
022	HLF	^R	0110	062	2	0310
023	RVT	^S	0114	063	3	0314
024	HLR	^T	0120	064	4	0320
025	NAK	^U	0124	065	5	0324
026	SYN	^V	0130	066	6	0330
027	ETB	^W	0134	067	7	0334
030	CAN	^X	0140	070	8	0340
031	EM	^Y	0144	071	9	0344
032	SUB	^Z	0150	072	:	0350
033	ESC	^UP-K	0154	073	;	0354
034	FS	^UP-L	0160	074	<	0360
035	GS	^UP-M	0164	075	=	0364
036	RS	^UP-N	0170	076	>	0370
037	US	^UP-O	0174	077	?	0374

8-Bit Octal Code	Char	Code in Left Byte	8-Bit Octal Code	Char	Code in Left Byte
100		0400	140	`	0600
101	A	0404	141	a	0604
102	B	0410	142	b	0610
103	C	0414	143	c	0614
104	D	0420	144	d	0620
105	E	0424	145	e	0624
106	F	0430	146	f	0630
107	G	0434	147	g	0634
110	H	0440	150	h	0640
111	I	0444	151	i	0644
112	J	0450	152	j	0650
113	K	0454	153	k	0654
114	L	0460	154	l	0660
115	M	0464	155	m	0664
116	N	0470	156	n	0670
117	O	0474	157	o	0674
120	P	0500	160	p	0700
121	Q	0504	161	q	0704
122	R	0510	162	r	0710
123	S	0504	163	s	0714
124	T	0520	164	t	0720
125	U	0524	165	u	0724
126	V	0530	166	v	0730
127	W	0534	167	w	0734
130	X	0540	170	x	0740
131	Y	0544	171	y	0744
132	Z	0550	172	z	0750
133	[	0554	173	{	0754
334	#	0560	174	|	0760
135	]	0564	175	}	0764
136	^	0570	176	~	0770
137	_	0574	177	DEL	0774

2.1.2.2 Prime ASCII:

F => Valid file name character
R => Reserved command line character
^ => Control key depressed

8-Bit Octal Code	Char		Code in Left Byte	8-Bit Octal Code	Char	Code in Left Byte	
200	NUL		1000	240	Sp	1200	
201	SOH	^A	1004	241	!	1204	R
202	STX	^B	1010	242	"	1210	R
203	ETX	^C	1014	243	#	1214	F
204	EOT	^D	1020	244	$	1220	F
205	ENQ	^E	1024	245			
206	ACK	^F	1030	246		1230	FR
207	BEL	^G	1034	247	'	1234	R
210	BS	^H	1040	250	(	1240	R
211	HT	^I	1044	251	)	1244	R
212	NL	^J	1050	252	*	1250	F
213	VT	^K	1054	253	+	1254	
214	FF	^L	1060	254	,	1260	R
215	CR	^M	1064	255	-	1264	FR
216	RRS	^N	1070	256	.	1270	F
217	BRS	^O	1074	257	/	1274	F
220	RCP	^P	1100	260	0	1300	F
221	RHT	^Q	1104	261	1	1304	F
222	HLF	^R	1110	262	2	1310	F
223	RVT	^S	1114	263	3	1314	F
224	PLR	^T	1120	264	4	1320	F
225	NAK	^U	1124	265	5	1324	F
226	SYN	^V	1130	266	6	1330	F
227	ETB	^W	1134	267	7	1334	F
230	CAN	^X	1140	270	8	1340	F
231	EM	^Y	1144	271	9	1344	F
232	SUB	^Z	1150	272	:	1350	R
233	ESC	^UP-K	1154	273	;	1354	R
234	FS	^UP-L	1160	274	<	1360	
235	GS	^UP-M	1164	275	=	1364	R
236	RS	^UP-N	1170	276	>	1370	
237	US	^UP-O	1174	277	?	1374	

8-Bit Octal Code	Char	Code in Left Byte		8-Bit Octal Code	Char	Code in Left Byte	
300		1400	R	340	`	1600	R
301	A	1404	F	341	a	1604	
302	B	1410	F	342	b	1610	
303	C	1414	F	343	c	1614	
304	D	1420	F	344	d	1620	
305	E	1424	F	345	e	1624	
306	F	1430	F	346	f	1630	
307	G	1434	F	347	g	1634	
310	H	1440	F	350	h	1640	
311	I	1444	F	351	i	1644	
312	J	1450	F	352	j	1650	
313	K	1454	F	353	k	1654	
314	L	1460	F	354	l	1660	
315	M	1464	F	355	m	1664	
316	N	1470	F	356	n	1670	
317	O	1474	F	357	o	1674	
320	P	1500	F	360	p	1700	
321	Q	1504	F	361	q	1704	
322	R	1510	F	362	r	1710	
323	S	1514	F	363	s	1714	
324	T	1520	F	364	t	1720	
325	U	1524	F	365	u	1724	
326	V	1530	F	366	v	1730	
327	W	1534	F	367	w	1734	
330	X	1540	F	370	x	1740	
331	Y	1544	F	371	y	1744	
332	Z	1550	F	372	z	1750	
333	[	1554	R	373	{	1754	R
334	\	1560	R	374	|	1760	
335	]	1564	R	375	}	1764	R
336	^	1570	R	376	~	1770	R
337	_	1574	F	377	DEL	1774	

2.1.2.3 EBCDIC:

(bank) = bank character

Decimal	Octal	Hex.	Char.	Decimal	Octal	Hex.	Char.
000	000	00	NUL	032	040	20	DS
001	001	01	SOH	033	041	21	SOS
002	002	02	STX	034	042	22	FS
003	003	03	ETX	035	043	23	
004	004	04	PF	036	044	24	BYP
005	005	05	HT	037	045	25	LF
006	006	06	LC	038	046	26	ETB
007	007	07	DEL	039	047	27	ESC
008	010	08		040	050	28	
009	011	09		041	051	29	
010	012	0A	SMM	042	052	2A	SM
011	013	0B	VT	043	053	2B	CU2
012	014	0C	FF	044	054	2C	
013	015	0D	CR	045	055	2D	ENQ
014	016	0F	SO	046	056	2E	ACK
015	017	0F	SI	047	057	2F	BEL
016	020	10	DLE	048	060	30	
017	021	11	DC1	049	061	31	
018	022	12	DC2	050	062	32	SYN
019	023	13	TM	051	063	33	
020	024	14	RES	052	064	34	PN
021	025	15	NL	053	065	35	RS
022	026	16	BS	054	066	36	UC
023	027	17	IL	055	067	37	EOT
024	030	18	CAN	056	070	38	
025	031	19	EM	057	071	39	
026	032	1A	CC	058	072	3A	
027	033	1B	CU1	059	073	3B	CU3
028	034	1C	IFS	060	074	3C	DC4
029	035	1D	IGS	061	075	3D	NAK
030	036	1E	IRS	062	076	3E	
031	037	1F	IUS	063	077	3F	SUB

Decimal	Octal	Hex.	Char.	Decimal	Octal	Hex.	Char.
064	100	40	SP	096	140	60	-
065	101	41		097	141	61	/
066	102	42		098	142	62	
067	103	43		099	143	63	
068	104	44		100	144	64	
069	105	45		101	145	65	
070	106	46		102	146	66	
071	107	47		103	147	67	
072	110	48		104	150	68	
073	111	49		105	151	69	
074	112	4A	(cents)	106	152	6A	|
075	113	4B	.	107	153	6B	,
076	114	4C	<	108	154	6C	%
077	115	4D	(	109	155	6D	_
078	116	4F	+	110	156	6E	>
079	117	4F	|	111	157	6F	?
080	120	50		112	160	70	
081	121	51		113	161	71	
082	122	52		114	162	72	
083	123	53		115	163	73	
084	124	54		116	164	74	
085	125	55		117	165	75	
086	126	56		118	166	76	
087	127	57		119	167	77	
088	130	58		120	170	78	
089	131	59		121	171	79	`
090	132	5A	!	122	172	7A	:
091	133	5B	$	123	173	7B	#
092	134	5C	*	124	174	7C	@
093	135	5D	)	125	175	7D	'
094	136	5E	;	126	176	7E	=
095	137	5F	(not)	127	177	7F	"

Decimal	Octal	Hex.	Char.	Decimal	Octal	Hex.	Char.
128	200	80		160	240	A0	
129	201	81	a	161	241	A1	~
130	202	82	b	162	242	A2	s
131	203	83	c	163	243	A3	t
132	204	84	d	164	244	A4	u
133	205	85	e	165	245	A5	v
134	206	86	f	166	246	A6	w
135	207	87	g	167	247	A7	x
136	210	88	h	168	250	A8	y
137	211	89	i	169	251	A9	z
138	212	8A		170	252	AA	
139	213	8B		171	253	AB	
140	214	8C		172	254	AC	
141	215	8D		173	255	AD	
142	216	8E		174	256	AE	
143	217	8F		175	257	AF	
144	220	90		176	260	B0	
145	221	91	j	177	261	B1	
146	222	92	k	178	262	B2	
147	223	93	l	179	263	B3	
148	224	94	m	180	264	B4	
149	225	95	n	181	265	B5	
150	226	96	o	182	266	B6	
151	227	97	p	183	267	B7	
152	230	98	q	184	270	B8	
153	231	99	r	185	271	B9	
154	232	9A		186	272	BA	
155	233	9B		187	273	BB	
156	234	9C		188	274	BC	
157	235	9D		189	275	BD	
158	236	9E		190	276	BE	
159	237	9F		191	277	BF	

Decimal	Octal	Hex.	Char.	Decimal	Octal	Hex.	Char.
192	300	C0	{	224	340	E0	ß
193	301	C1	A	225	341	E1	
194	302	C2	B	226	342	E2	S
195	303	C3	C	227	343	E3	T
196	304	C4	D	228	344	E4	U
197	305	C5	E	229	345	E5	V
198	306	C6	F	230	346	E6	W
199	307	C7	G	231	347	E7	X
200	310	C8	H	232	350	E8	Y
201	311	C9	I	233	351	E9	Z
202	312	CA		234	352	EA	
203	313	CB		235	353	EB	
204	314	CC	(bank)	236	354	EC	(bank)
205	315	CD		237	355	ED	
206	316	CE	(bank)	238	356	EE	
207	317	CF		239	357	EF	
208	320	D0	}	240	360	F0	0
209	321	D1	J	241	361	F1	1
210	322	D2	K	242	362	F2	2
211	323	D3	L	243	363	F3	3
212	324	D4	M	244	364	F4	4
213	325	D5	N	245	365	F5	5
214	326	D6	O	246	366	F6	6
215	327	D7	P	247	367	F7	7
216	330	D8	Q	248	370	F8	8
217	331	D9	R	249	371	F9	9
218	332	DA		250	372	FA	|
219	333	DB		251	373	FB	
220	334	DC		252	374	FC	
221	335	DD		253	375	FD	
222	336	DE		254	376	FE	
223	337	DF		255	377	FF	

2.1.2.4 BCD:

Decimal	Octal	Hex.	Char.	Decimal	Octal	Hex.	Char.
000	000	00	?	032	040	20	-
001	001	01	A	033	041	21	/
002	002	02	B	034	042	22	S
003	003	03	C	035	043	23	T
004	004	04	D	036	044	24	U
005	005	05	E	037	045	25	V
006	006	06	F	038	046	26	W
007	007	07	G	039	047	27	X
008	010	08	H	040	050	28	Y
009	011	09	I	041	051	29	Z
010	012	0A		042	052	2A	
011	013	0B	.	043	053	2B	,
012	014	0C	)	044	054	2C	
013	015	0D	[	045	055	2D	not
014	016	0E	<	046	056	2E	¤
015	017	0F		047	057	2F	
016	020	10	+ !	048	060	30	0
017	021	11	J	049	061	31	1
018	022	12	K	050	062	32	2
019	023	13	L	051	063	33	3
020	024	14	M	052	064	34	4
021	025	15	N	053	065	35	5
022	026	16	O	054	066	36	6
023	027	17	P	055	067	37	7
024	030	18	Q	056	070	38	8
025	031	19	R	057	071	39	9
026	032	1A		058	072	3A	blank
027	033	1B	$	059	073	3B	# =
028	034	1C	*	060	074	3C	@ '
029	035	1D	]	061	075	3D	:
030	036	1E	;	062	076	3E	>
031	037	1F		063	077	3F	check

2.1.3 Translation Tables

2.1.4 BCD to Prime ASCII:

Binary	Octal	Decimal	Hex.	Characters
1011000010110001	130261	45233	B0B1	0 1
1011001010110011	131263	45747	B2B3	2 3
1011010010110101	132265	46261	B4B5	4 5
1011011010110111	133267	46775	B6B7	6 7
1011100010111001	134271	47289	B8B9	8 9
1011111110111101	137675	49085	BFBD	? =
1010011110111010	123672	42938	A7BA	' :
1010000110111110	120676	41406	A1BE	! >
1010000010101111	120257	41135	A0AF	spc /
1101001111010100	151724	54228	D3D4	S T
1101010111010110	152726	54742	D5D6	U V
1101011111011000	153730	55256	D7D8	W X
1101100111011010	154732	55770	D9DA	Y Z
1011111110101100	137654	49068	BFAC	? ,
1010100010111111	124277	43199	A8BF	(?
1101110110100010	156642	56738	DDA2	] "
1010110111001010	126712	44490	ADCA	- J
1100101111001100	145714	52172	CBCC	K L
1100110111001110	146716	52686	CDCE	M N
1100111111010000	147720	53200	CFD0	O P
1101000111010010	150722	53714	D1D2	Q R
1011101110100100	135644	48036	BBA4	; $
1010101011011011	125333	43739	AADB	* [
1011111110111100	137674	49084	BFBC	? <
1010101111000001	125701	43969	ABC1	+ A
1100001011000011	141303	49859	C2C3	B C
1100010011000101	142305	50373	C4C5	D E
1100011011000111	143307	50887	C6C7	F G
1100100011001001	144311	51401	C8C9	H I
1011111110101110	137656	49070	BFAE	? .
1010100110100101	124645	43429	A9A5	) %
1101110011011111	156337	56543	DCDF	^ _
0000000000000000	000000	00000	0000	(No lowercase or control characters. This line repeats 95 additional times.)

2.1.5 Prime ASCII to BCD:

Binary	Octal	Decimal	Hex.	Characters
0000000000000000	000000	00000	0000	(No lowercase or control characters. This line repeats 15 additional times.)
0001000000001110	010016	04110	100E	spc !
0001111100001011	017413	07947	1F0B	" #
0010101100111101	025475	11069	2B3D	$ %
0000000000001100	000014	00012	000C	? '
0001110000111100	016074	07228	1C3C	( )
0010110000110000	026060	11312	2C30	* +
0001101100100000	015440	06944	1B20	, -
0011101100010001	035421	15121	3B11	. /
0000000000000001	000001	00001	0001	0 1
0000001000000011	001003	00515	0203	2 3
0000010000000101	002005	01029	0405	4 5
0000011000000111	003007	01543	0607	6 7
0000100000001001	004011	02057	0809	8 9
0000110100101010	006452	03370	0D2A	: ;
0010111100001011	027413	12043	2F0B	< =
0000111100111010	007472	03898	0F3A	> ?
0000000000110001	000061	00049	0031	? A
0011001000110011	031063	12851	3233	B C
0011010000110101	032065	13365	3435	D E
0011011000110111	033067	13879	3637	F G
0011100000111001	034071	14393	3839	H I
0010000100100010	020442	08482	2122	J K
0010001100100100	021444	08996	2324	L M
0010010100100110	022446	09510	2526	N O
0010011100101000	023450	10024	2728	P Q
0010100100010010	024422	10514	2912	R S
0001001100010100	011424	04884	1314	T U
0001010100010110	012426	05398	1516	V W
0001011100011000	013430	05912	1718	X Y
0001100100101101	014455	06445	192D	Z]
0011111000011110	037036	15902	3E1E	↑ [
0011101000111111	035077	14911	3A3F	^ _
0000000000000000	000000	00000	0000	(No lowercase or control characters. This line repeats 31 additional times.)

```
  0001000000001110  010016  04110   100E    spc !
  0001111100001011  017413  07947   1F0B    "   #
  0010101100111101  025475  11069   2B3D    $   %
  0000000000001100  000014  00012   000C    ?   '
  0001110000111100  016074  07228   1C3C    (   )
  0010110000110000  026060  11312   2C30    *   +
  0001101100100000  015440  06944   1B20    ,   -
  0011101100010001  035421  15121   3B11    .   /
  0000000000000001  000001  00001   0001    0   1
  0000001000000011  001003  00515   0203    2   3
  0000010000000101  002005  01029   0405    4   5
  0000011000000111  003007  01543   0607    6   7
  0000100000001001  004011  02057   0809    8   9
  0001101001001010  006452  03370   0D2A    :   ;
  0010111100001011  027413  12043   2F0B    <   =
  0000111100111010  007472  03898   0F3A    >   ?
  0000000000110001  000061  00049   0031    ?   A
  0011001000110011  031063  12851   3233    B   C
  0011010000110101  032065  13365   3435    D   E
  0011011000110111  033067  13879   3637    F   G
  0011100001110001  034071  14393   3839    H   I
  0010000100100010  020442  08482   2122    J   K
  0010001100100100  021444  08996   2324    L   M
  0010010100100110  022446  09510   2526    N   O
  0010011100101000  023450  10024   2728    P   Q
  0010100100010010  024422  10514   2912    R   S
  0001001100010100  011424  04884   1314    T   U
  0001010100010110  012426  05398   1516    V   W
  0001011100011000  013430  05912   1718    X   Y
  0001100100101101  014455  06445   192D    Z   ]
  0011111000011110  037036  15902   3E1E    \   [
  0011101000111111  035077  14911   3A3F    ^   _
  0000000000000000  000000  00000   0000    (No lowercase or contr
ol
                                            characters.  This lin
e
                                            repeats 15 additional
                                            times.)
```

2.1.6 Prime ASCII to EBCDIC:

Binary	Octal	Decimal	Hex.	Characters	
0000000000000001	000001	00001	0001	NUL	SOH
0000001000000011	001003	00515	0203	STX	ETX
0011011100101101	033455	14125	372D	EOT	ENQ
0111110000101111	076057	31791	7C2F	@	BEL
0001011000000101	013005	05637	1605	BS	HT
0010010100001011	022413	09483	250B	LF	VT
0000110000001101	006015	03085	0C0D	FF	CR
0000111000001111	007017	03599	0E0F	SO	SI
0001000000010001	010021	04113	1011	DLE	DC1
0001001000010011	011023	04627	1213	DC2	TM
0011110000111101	036075	15421	3C3D	DC4	NAK
0011001000100110	031046	12838	3226	SYN	ETB
0001100000011001	014031	06169	1819	CAN	EM
0011111100100111	037447	16167	3F27	SUB	ESC
0010000100111100	020174	08828	227C	FS	@
0011010101111100	032574	13692	357C	RS	@
0100000001011010	040132	16474	405A	SP	!
0111111101111011	077573	32635	7F7B	"	#
0101101101101100	055554	23404	5B6C	$	%
0101000001111101	050175	20605	507D		'
0100110101011101	046535	19805	4D5D	(	)
0101110001001110	056116	23630	5C4E	*	+
0110101101100000	065540	27488	6B60	,	-
0100101101100001	045541	19297	4B61	.	/
1111000011110001	170361	61681	F0F1	0	1
1111001011110011	171363	62195	F2F3	2	3
1111010011110101	172365	62709	F4F5	4	5
1111011011110111	173367	63223	F6F7	6	7
1111100011111001	174371	63737	F8F9	8	9
0111101001011110	075136	31326	7A5E	:	;
0100110001111110	046176	19582	4C7E	<	=
0110111001101111	067157	28271	6E6F	>	?
0111110011000001	076301	31937	7CC1	@	A
1100001011000011	141303	49859	C2C3	B	C
1100010011000101	142305	50373	C4C5	D	E
1100011011000111	143307	50887	C6C7	F	G
1100100011001001	144311	51401	C8C9	H	I
1101000111010010	150722	53714	D1D2	J	K
1101001111010100	151724	54228	D3D4	L	M
1101010111010110	152726	54742	D5D6	N	O
1101011111011000	153730	55256	D7D8	P	Q
1101100111100010	154742	55778	D9E2	R	S
1110001111100100	161744	58340	E3E4	T	U
1110010111100110	162746	58854	E5E6	V	W
1110011111101000	163750	59368	E7E8	X	Y
1110100101111100	164574	59772	E97C	Z	@
1110000001111100	160174	57468	E07C	■	@
0101111101101101	057555	24429	5F6D	not	_

0111100110000001	074601	31105	7981	`	a
1000001010000011	101203	33411	8283	b	c
1000010010000101	102205	33925	8485	d	e
1000011010000111	103207	34439	8687	f	g
1000100010001001	104211	34953	8889	h	i
1001000110010010	110622	37266	9192	j	k
1001001110010100	111624	37780	9394	l	m
1001010110010110	112626	38294	9596	n	o
1001011110011000	113630	38808	9798	p	q
1001100110100010	114642	39330	99A2	r	s
1010001110100100	121644	41892	A3A4	t	u
1010010110100110	122646	42406	A5A6	v	w
1010011110101000	123650	42920	A7A8	x	y
1010100111000000	124700	43456	A9C0	z	{
0110101011010000	065320	27344	6AD0	|	}
1010000100000111	120407	41223	A107	~	del
0000000000000001	000001	00001	0001	NUL	SOH
0000001000000011	001003	00515	0203	STX	ETX
0011011100101101	033455	14125	372D	EOT	ENQ
0111110000101111	076057	31791	7C2F	@	BEL
0001011000000101	013005	05637	1605	BS	HT
0010010100001011	022413	09483	250B	LF	VT
0000110000001101	006015	03085	0C0D	FF	CR
0000111000001111	007017	03599	0E0F	SO	SI
0001000000010001	010021	04113	1011	DLE	DC1
0001001000010011	011023	04627	1213	DC2	TM
0011110000111101	036075	15421	3C3D	DC4	NAK
0011001000100110	031046	12838	3226	SYN	ETB
0001100000011001	014031	06169	1819	CAN	EM
0011111100100111	037447	16167	3F27	SUB	ESC
0010001001111100	021174	08828	227C	FS	@
0011010101111100	032574	13692	357C	RS	@
0100000001011010	040132	16474	405A	SP	!
0111111101111011	077573	32635	7F7B	"	#
0101101101101100	055554	23404	5B6C	$	%
0101000001111101	050175	20605	507D		'
0100110101011101	046535	19805	4D5D	(	)
0101110001001110	056116	23630	5C4E	*	+
0110101101100000	065540	27488	6B60	,	-
0100101101100001	045541	19297	4B61	.	/
1111000011110001	170361	61681	F0F1	0	1
1111001011110011	171363	62195	F2F3	2	3
1111010011110101	172365	62709	F4F5	4	5
1111011011110111	173367	63223	F6F7	6	7
1111100011111001	174371	63737	F8F9	8	9
0111101001011110	075136	31326	7A5E	:	;
0100110001111110	046176	19582	4C7E	<	=
0110111001101111	067157	28271	6E6F	>	?
0111110011000001	076301	31937	7CC1	@	A
1100001011000011	141303	49859	C2C3	B	C
1100010011000101	142305	50373	C4C5	D	E
1100011011000111	143307	50887	C6C7	F	G
1100100011001001	144311	51401	C8C9	H	I
1101000111010010	150722	53714	D1D2	J	K

1101001111010100	151724	54228	D3D4	L	M
1101010111010110	152726	54742	D5D6	N	O
1101011111011000	153730	55256	D7D8	P	Q
1101100111100010	154742	55778	D9E2	R	S
1110001111100100	161744	58340	E3E4	T	U
1110010111100110	162746	58854	E5E6	V	W
1110011111101000	163750	59368	E7E8	X	Y
1110100101111100	164574	59772	E97C	Z	@
1110000001111100	160174	57468	E07C	~	@
0101111101101101	057555	24429	5F6D	not	_
0111100110000001	074601	31105	7981	`	a
1000001010000011	101203	33411	8283	b	c
1000010010000101	102205	33925	8485	d	e
1000011010000111	103207	34439	8687	f	g
1000100010001001	104211	34953	8889	h	i
1001000110010010	110622	37266	9192	j	k
1001001110010100	111624	37780	9394	l	m
1001010110010110	112626	38294	9596	n	o
1001011110011000	113630	38808	9798	o	q
1001100110100010	114642	39330	99A2	r	s
1010001110100100	121644	41892	A3A4	t	u
1010010110100110	122646	42406	A5A6	v	w
1010011110101000	123650	42920	A7A8	x	y
1010100111000000	124700	43456	A9C0	z	{
0110101011010000	065320	27344	6AD0	|	}
1010000100000111	120407	41223	A107	~	del

2.1.7 EBCDIC to Prime ASCII:

Binary	Octal	Decimal	Hex.	Characters	
1010000010000001	120201	41089	A081	space	SOH
1000001010000011	101203	33411	8283	STX	ETX
1011111110001001	137611	49033	BF89	@	HT
1011111110010000	137620	49040	BF90	@	RCP
1011111110111111	137677	49087	BFBF	@	@
1011111110001011	137613	49035	BF8B	@	VT
1000110010001101	106215	35981	8C8D	FF	CR
1000111010001111	107217	36495	8E8F	RRS	BRS
1001000010010001	110221	37009	9091	RCP	RHT
1001001010010011	111223	37523	9293	HLF	RVT
1011111110111111	137677	49087	BFBF	@	@
1000100010111111	104277	35007	88BF	BS	@
1001100010011001	114231	39065	9899	CAN	EM
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1001110010111111	116277	40127	9CBF	FS	@
1011111110001010	137612	49034	BF8A	@	NL
1001011110011011	113633	38811	979B	ETB	ESC
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110000101	137605	49029	BF85	@	ENQ
1000011010000111	103207	34439	8687	ACK	BEL
1000000010000000	100200	32896	8080	NUL	NUL
1001011010000000	113200	38528	9680	SYN	NUL
1011111110011110	137636	49054	BF9E	@	RS
1011111110000100	137604	49028	BF84	@	EOT
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1001010010010101	112225	38037	9495	HLR	NAK
1000000010011010	100232	32922	809A	NUL	SUB
1010000010111111	120277	41151	A0BF	SP	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110101110	137656	49070	BFAE	@	.
1011110010101000	136250	48296	BCA8	<	(
1010101111111100	125774	44028	ABFC	+	|
1010011010111111	123277	42687	A6BF		@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1010000110100100	120644	41380	A1A4	!	$
1010101010101001	125251	43689	AAA9	*	)
1011101110111111	135677	48063	BBBF	;	@

1010110110101111	126657	44463	ADAF	-	/
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110101100	137654	49068	BFAC	@	,
1010010111011111	122737	42463	A5DF		
1011111010111111	137277	48831	BEBF	>	?
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111111100000	137740	49120	BFE0	@	SP
1011101010100011	135243	47779	BAA3	:	#
1100000010100111	140247	49319	C0A7	'@'	'
1011110110100010	136642	48546	BDA2	=	"
1011111111100001	137741	49121	BFE1	@	a
1110001011100011	161343	58083	E2E3	b	c
1110010011100101	162345	58597	E4E5	d	e
1110011011100111	163347	59111	E6E7	f	g
1110100011101001	164351	59625	E8E9	h	i
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111111101010	137752	49130	BFEA	@	j
1110101111101100	165754	60396	EBEC	k	l
1110110111101110	166756	60910	EDEE	m	n
1110111111110000	167760	61424	EFF0	o	p
1111000111110010	170762	61938	F1F2	q	r
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111111111110	137776	49150	BFFE	@	~
1111001111110100	171764	62452	F3F4	s	t
1111010111110110	172766	62966	F5F6	u	v
1111011111111000	173770	63480	F7F8	w	x
1111100111111010	174772	63994	F9FA	y	z
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1111101111000001	175701	64449	FBC1	{	A
1100001011000011	141303	49859	C2C3	B	C
1100010011000101	142305	50373	C4C5	D	E
1100011011000111	143307	50887	C6C7	F	G
1100100011001001	144311	51401	C8C9	H	I
1011111110111111	137677	49087	BFBF	@	@

1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1111101111001010	175712	64458	FBCA	{	J
1100101111001100	145714	52172	CBCC	K	L
1100110111001110	146716	52686	CDCE	M	N
1100111111010000	147720	53200	CFD0	O	P
1101000111010010	150722	53714	D1D2	Q	R
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1101110010111111	156277	56511	DCBF	~	@
1101001111010100	151724	54228	D3D4	S	T
1101010111010110	152726	54742	D5D6	U	V
1101011111011000	153730	55256	D7D8	W	X
1101100111011010	154732	55770	D9DA	Y	Z
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@
1011000010110001	130261	45233	B0B1	0	1
1011001010110011	131263	45747	B2B3	2	3
1011010010110101	132265	46261	B4B5	4	5
1011011010110111	133267	46775	B6B7	6	7
1011100010111001	134271	47289	B8B9	8	9
1111110010111111	176277	64703	FCBF	|	@
1011111110111111	137677	49087	BFBF	@	@
1011111110111111	137677	49087	BFBF	@	@

2.1.8 Industry-Standard ASCII to Prime ASCII:

Binary	Octal	Decimal	Hex.	Characters	
1000000010000001	100201	32897	8081	NUL	SOH
1000001010000011	101203	33411	8283	STX	ETX
1000010010000101	102205	33925	8485	EOT	ENQ
1000011010000111	103207	34439	8687	ACK	BEL
1000100010001001	104211	34953	8889	BS	HT
1000101010001011	105213	35467	8A8B	NL	VT
1000110010001101	106215	35981	8C8D	FF	CR
1000111010001111	107217	36495	8E8F	RRS	BRS
1001000010010001	110221	37009	9091	RCP	RHT
1001001010010011	111223	37523	9293	HLF	RVT
1001010010010101	112225	38037	9495	HLR	NAK
1001011010010111	113227	38551	9697	SYN	ETB
1001100010011001	114231	39065	9899	CAN	EM
1001101010011011	115233	39579	9A9B	SUB	ESC
1001110010011101	116235	40093	9C9D	FS	GS
1001111010011111	117237	40607	9E9F	RS	US
1010000010100001	120241	41121	A0A1	sp	!
1010001010100011	121243	41635	A2A3	"	#
1010010010100101	122245	42149	A4A5	$	%
1010011010100111	123247	42663	A6A7		'
1010100010101001	124251	43177	A8A9	(	)
1010101010101011	125253	43691	AAAB	*	+
1010110010101101	126255	44205	ACAD	,	-
1010111010101111	127257	44719	AEAF	.	/
1011000010110001	130261	45233	B0B1	0	1
1011001010110011	131263	45747	B2B3	2	3
1011010010110101	132265	46261	B4B5	4	5
1011011010110111	133267	46775	B6B7	6	7
1011100010111001	134271	47289	B8B9	8	9
1011101010111011	135273	47803	BABB	:	;
1011110010111101	136275	48317	BCBD	<	=
1011111010111111	137277	48831	BEBF	>	?
1100000011000001	140301	49345	C0C1	nul	A
1100001011000011	141303	49859	C2C3	B	C
1100010011000101	142305	50373	C4C5	D	E
1100011011000111	143307	50887	C6C7	F	G
1100100011001001	144311	51401	C8C9	H	I
1100101011001011	145313	51915	CACB	J	K
1100110011001101	146315	52429	CCCD	L	M
1100111011001111	147317	52943	CECF	N	O
1101000011010001	150321	53457	D0D1	P	Q
1101001011010011	151323	53971	D2D3	R	S
1101010011010101	152325	54485	D4D5	T	U
1101011011010111	153327	54999	D6D7	V	W
1101100011011001	154331	55513	D8D9	X	Y
1101101011011011	155333	56027	DADB	Z	[
1101110011011101	156335	56541	DCDD	\	]
1101111011011111	157337	57055	DEDF	^	_

1110000011100001	160341	57569	E0E1	nul	a
1110001011100011	161343	58083	E2E3	b	c
1110010011100101	162345	58597	E4E5	d	e
1110011011100111	163347	59111	E6E7	f	g
1110100011101001	164351	59625	E8E9	h	i
1110101011101011	165353	60139	EAEB	j	k
1110110011101101	166355	60653	ECED	l	m
1110111011101111	167357	61167	EEEF	n	o
1111000011110001	170361	61681	F0F1	p	q
1111001011110011	171363	62195	F2F3	r	s
1111010011110101	172365	62709	F4F5	t	u
1111011011110111	173367	63223	F6F7	v	w
1111100011111001	174371	63737	F8F9	x	y
1111101011111011	175373	64251	FAFB	z	{
1111110011111101	176375	64765	FCFD	|	}
1111111011111111	177377	65279	FEFF	~	DEL
1000000010000001	100201	32897	8081	NUL	SOH
1000001010000011	101203	33411	8283	STX	FTX
1000010010000101	102205	33925	8485	EOT	ENQ
1000011010000111	103207	34439	8687	ACK	BEL
1000100010001001	104211	34953	8889	BS	HT
1000101010001011	105213	35467	8A8B	NL	VT
1000110010001101	106215	35981	8C8D	FF	CR
1000111010001111	107217	36495	8E8F	RRS	BRS
1001000010010001	110221	37009	9091	RCP	RHT
1001001010010011	111223	37523	9293	HLF	RVT
1001010010010101	112225	38037	9495	HLR	NAK
1001011010010111	113227	38551	9697	SYN	ETB
1001100010011001	114231	39065	9899	CAN	EM
1001101010011011	115233	39579	9A9B	SUB	ESC
1001110010011101	116235	40093	9C9D	FS	GS
1001111010011111	117237	40607	9E9F	RS	US
1010000010100001	120241	41121	A0A1	sp	!
1010001010100011	121243	41635	A2A3	"	#
1010010010100101	122245	42149	A4A5	$	%
1010011010100111	123247	42663	A6A7		•
1010100010101001	124251	43177	A8A9	(	)
1010101010101011	125253	43691	AAAB	*	+
1010110010101101	126255	44205	ACAD	,	-
1010111010101111	127257	44719	AEAF	.	/
1011000010110001	130261	45233	B0B1	0	1
1011001010110011	131263	45747	B2B3	2	3
1011010010110101	132265	46261	B4B5	4	5
1011011010110111	133267	46775	B6B7	6	7
1011100010111001	134271	47289	B8B9	8	9
1011101010111011	135273	47803	BABB	:	;
1011110010111101	136275	48317	BCBD	<	=
1011111010111111	137277	48831	BEBF	>	?
1100000011000001	140301	49345	C0C1	nul	A
1100001011000011	141303	49859	C2C3	B	C
1100010011000101	142305	50373	C4C5	D	E
1100011011000111	143307	50887	C6C7	F	G
1100100011001001	144311	51401	C8C9	H	I
1100101011001011	145313	51915	CACB	J	K

1100110011001101	146315	52429	CCCD	L	M
1100111011001111	147317	52943	CECF	N	O
1101000011010001	150321	53457	D0D1	P	Q
1101001011010011	151323	53971	D2D3	R	S
1101010011010101	152325	54485	D4D5	T	U
1101011011010111	153327	54999	D6D7	V	W
1101100011011001	154331	55513	D8D9	X	Y
1101101011011011	155333	56027	DADB	Z	[
1101110011011101	156335	56541	DCDD	\	]
1101111011011111	157337	57055	DEDF	^	_
1110000011100001	160341	57569	E0E1	nul	a
1110001011100011	161343	58083	E2E3	b	c
1110010011100101	162345	58597	E4E5	d	e
1110011011100111	163347	59111	E6E7	f	g
1110100011101001	164351	59625	E8E9	h	i
1110101011101011	165353	60139	EAEB	j	k
1110110011101101	166355	60653	ECED	l	m
1110111011101111	167357	61167	EEEF	n	o
1111000011110001	170361	61681	F0F1	p	q
1111001011110011	171363	62195	F2F3	r	s
1111010011110101	172365	62709	F4F5	t	u
1111011011110111	173367	63223	F6F7	v	w
1111100011111001	174371	63737	F8F9	x	y
1111101011111011	175373	64251	FAFB	z	{
1111110011111101	176375	64765	FCFD	|	}
1111111011111111	177377	65279	FEFF	~	DEL

2.1.9 Prime ASCII to Industry-Standard ASCII:

Binary	Octal	Decimal	Hex.	Characters	
0000000000000001	000001	00001	0001	NUL	SOH
0000001000000011	001003	00515	0203	STX	ETX
0000010000000101	002005	01029	0405	EOT	ENQ
0000011000000111	003007	01543	0607	ACK	BEL
0000100000001001	004011	02057	0809	BS	HT
0000101000001011	005013	02571	0A0B	NL	VT
0000110000001101	006015	03085	0C0D	FF	CR
0000111000001111	007017	03599	0E0F	RRS	BRS
0001000000010001	010021	04113	1011	RCP	RHT
0001001000010011	011023	04627	1213	HLF	RVT
0001010000010101	012025	05141	1415	HLR	NAK
0001011000010111	013027	05655	1617	SYN	ETB
0001100000011001	014031	06169	1819	CAN	EM
0001101000011011	015033	06683	1A1B	SUB	ESC
0001110000011101	016035	07197	1C1D	FS	GS
0001111000011111	017037	07711	1E1F	RS	US
0010000000100001	020041	08225	2021	sp	!
0010001000100011	021043	08739	2223	"	#
0010010000100101	022045	09253	2425	$	%
0010011000100111	023047	09767	2627		'
0010100000101001	024051	10281	2829	(	)
0010101000101011	025053	10795	2A2B	*	+
0010110000101101	026055	11309	2C2D	,	-
0010111000101111	027057	11823	2E2F	.	/
0011000000110001	030061	12337	3031	0	1
0011001000110011	031063	12851	3233	2	3
0011010000110101	032065	13365	3435	4	5
0011011000110111	033067	13879	3637	6	7
0011100000111001	034071	14393	3839	8	9
0011101000111011	035073	14907	3A3B	:	;
0011110000111101	036075	15421	3C3D	<	=
0011111000111111	037077	15935	3E3F	>	?
0100000001000001	040101	16449	4041	nul	A
0100001001000011	041103	16963	4243	B	C
0100010001000101	042105	17477	4445	D	E
0100011001000111	043107	17991	4647	F	G
0100100001001001	044111	18505	4849	H	I
0100101001001011	045113	19019	4A4B	J	K
0100110001001101	046115	19533	4C4D	L	M
0100111001001111	047117	20047	4E4F	N	O
0101000001010001	050121	20561	5051	P	Q
0101001001010011	051123	21075	5253	R	S
0101010001010101	052125	21589	5455	T	U
0101011001010111	053127	22103	5657	V	W
0101100001011001	054131	22617	5859	X	Y
0101101001011011	055133	23131	5A5B	Z	[
0101110001011101	056135	23645	5C5D	\	]
0101111001011111	057137	24159	5E5F	^	_

0110000001100001	060141	24673	6061	nul	a
0110001001100011	061143	25187	6263	b	c
0110010001100101	062145	25701	6465	d	e
0110011001100111	063147	26215	6667	f	g
0110100001101001	064151	26729	6869	h	i
0110101001101011	065153	27243	6A6B	j	k
0110110001101101	066155	27757	6C6D	l	m
0110111001101111	067157	28271	6E6F	n	o
0111000001110001	070161	28785	7071	p	q
0111001001110011	071163	29299	7273	r	s
0111010001110101	072165	29813	7475	t	u
0111011001110111	073167	30327	7677	v	w
0111100001111001	074171	30841	7879	x	y
0111101001111011	075173	31355	7A7B	z	{
0111110001111101	076175	31869	7C7D	|	}
0111111001111111	077177	32383	7E7F	~	DEL
0000000000000001	000001	00001	0001	NUL	SOH
0000001000000011	001003	00515	0203	STX	ETX
0000010000000101	002005	01029	0405	EOT	ENQ
0000011000000111	003007	01543	0607	ACK	BEL
0000100000001001	004011	02057	0809	BS	HT
0000101000001011	005013	02571	0A0B	NL	VT
0000110000001101	006015	03085	0C0D	FF	CR
0000111000001111	007017	03599	0E0F	RRS	BRS
0001000000010001	010021	04113	1011	RCP	RHT
0001001000010011	011023	04627	1213	HLF	RVT
0001010000010101	012025	05141	1415	HLR	NAK
0001011000010111	013027	05655	1617	SYN	ETB
0001100000011001	014031	06169	1819	CAN	EM
0001101000011011	015033	06683	1A1B	SUB	ESC
0001110000011101	016035	07197	1C1D	FS	GS
0001111000011111	017037	07711	1E1F	RS	US
0010000000100001	020041	08225	2021	sp	!
0010001000100011	021043	08739	2223	"	#
0010010000100101	022045	09253	2425	$	%
0010011000100111	023047	09767	2627		'
0010100000101001	024051	10281	2829	(	)
0010101000101011	025053	10795	2A2B	*	+
0010110000101101	026055	11309	2C2D	,	-
0010111000101111	027057	11823	2E2F	.	/
0011000000110001	030061	12337	3031	0	1
0011001000110011	031063	12851	3233	2	3
0011010000110101	032065	13365	3435	4	5
0011011000110111	033067	13879	3637	6	7
0011100000111001	034071	14393	3839	8	9
0011101000111011	035073	14907	3A3B	:	;
0011110000111101	036075	15421	3C3D	<	=
0011111000111111	037077	15935	3E3F	>	?
0100000001000001	040101	16449	4041	nul	A
0100001001000011	041103	16963	4243	B	C
0100010001000101	042105	17477	4445	D	E
0100011001000111	043107	17991	4647	F	G
0100100001001001	044111	18505	4849	H	I
0100101001001011	045113	19019	4A4B	J	K

0100110001001101	046115	19533	4C4D	L	M
0100111001001111	047117	20047	4E4F	N	O
0101000001010001	050121	20561	5051	P	Q
0101001001010011	051123	21075	5253	R	S
0101010001010101	052125	21589	5455	T	U
0101011001010111	053127	22103	5657	V	W
0101100001011001	054131	22617	5859	X	Y
0101101001011011	055133	23131	5A5B	Z	[
0101110001011101	056135	23645	5C5D	\	]
0101111001011111	057137	24159	5E5F	^	_
0110000001100001	060141	24673	6061	nul	a
0110001001100011	061143	25187	6263	b	c
0110010001100101	062145	25701	6465	d	e
0110011001100111	063147	26215	6667	f	g
0110100001101001	064151	26729	6869	h	i
0110101001101011	065153	27243	6A6B	j	k
0110110001101101	066155	27757	6C6D	l	m
0110111001101111	067157	28271	6E6F	n	o
0111000001110001	070161	28785	7071	p	q
0111001001110011	071163	29299	7273	r	s
0111010001110101	072165	29813	7475	t	u
0111011001110111	073167	30327	7677	v	w
0111100001111001	074171	30841	7879	x	y
0111101001111011	075173	31355	7A7B	z	{
0111110001111101	076175	31869	7C7D	|	}
0111111001111111	077177	32383	7E7F	~	DEL

3 APPENDIX B

3.1 COMMAND LINE OPTIONS AND BATCH JOBS

While invoking MAGNET, there are several command line options you can specify. The -SILENT option disables the printing of Severity 1 messages. (See Section 7 for detailed information on MAGNET messages.) The -USER and -OPERATOR (or -OPR) options specify where MOUNT and DISMOUNT messages are printed. -USER causes these messages to be printed on your terminal. -OPERATOR (or -OPR) causes them to be printed on the operator's console.

All MOUNT and DISMOUNT messages require a reply from either you or the operator. At the operator's console, the reply takes the form of the REPLY command (see Section 5). The operator, however, should not use the following form of the REPLY command:

```
OK, REPLY -usrnum -TAPE pdn
```

Acceptable forms for the operator are:

```
OK, REPLY -usrnum -TAPE GO

          or

OK, REPLY -usrnum -TAPE ABORT
```

After you receive a MOUNT or DISMOUNT message and a prompt (>) at your terminal, your reply should be one of the following:

```
> REPLY -TAPE GO

          or

> REPLY -TAPE ABORT
```

Any other response causes the MOUNT or DISMOUNT message to be repeated.

The default communication mode is -OPERATOR. Directing messages and replies to the operator's console rather than your input command stream allows you to build pre-packaged magnetic tape jobs that you can run under BATCH. These jobs can be run at any time, with tape mount and dismount instructions submitted to an operator prior to execution.

4 APPENDIX C

4.1 MAGNET ADDITIONAL FEATURES AND NOTES

4.1.1 Pre-rev 18.4 MAGNET

The pre-Rev 18.4 functionality of MAGNET is still supported. The four subcommands that still accept interactive dialogue are:

Subcommand	Function
POSITION	Positions the tape to a file/record
READ	Reads a file from tape to disk
WRITE	Writes a file to tape from disk
COPY	Copies a file from one tape to another

4.1.2 The POSITION Subcommand

The POSITION subcommand positions the magnetic tape to a specific file and record number. POSITION has two modes, absolute and relative. An absolute position causes the tape to be rewound before any spacing occurs. A relative position allows the tape to be moved forward or backward from the current position.

When you give the POSITION subcommand, MAGNET requests a magnetic tape unit number. The response is an integer in the range 0 to 7, optionally followed by a /7 or /9 to indicate a seven- or nine-track transport. The default is nine-track. MAGNET then asks whether this is an absolute or relative position. Finally, the file and record numbers are requested. If you specify an absolute position, the file and record numbers are absolute (starting with 1), and must be positive. In relative mode, the file number represents the number of files to space forward or backward and may be positive or negative. The record number is the number of records to space forward or backward and must be greater than or equal to zero. An example of a typical POSITION operation is as follows:

```
OK, MAGNET
[MAGNET, Rev. 19.0]
> POSITION
MTU # = 0
Relative or absolute? ABSOLUTE
File # = 23
Record # = 271
>
```

4.1.3 The READ Subcommand

The READ subcommand allows a file to be read from a magnetic tape and written to disk. It also provides optional unblocking and EBCDIC or BCD translation.

The READ subcommand requests the unit number, which you must enter in the format described under the POSITION subcommand. You then enter the magnetic tape file number, which must be a positive integer. If this number is greater than zero, the tape is rewound and positioned to the specified file number. If the file number is zero, the tape is not rewound.

The next series of questions in the READ subcommand dialogue relates to the format of the data on the magnetic tape file to be read. The logical record size is the number of bytes in each logical record. The blocking factor is the number of logical records (line images) contained in one physical tape record. MAGNET then asks you for the type of translation, and you provide one of the following responses:

ASCII — Specifies no translation between tape and disk. MAGNET writes the data to the disk file in Prime ASCII.

EBCDIC — Instructs MAGNET to translate the data on the tape from EBCDIC to Prime ASCII before writing it to the disk file.

BCD — Instructs MAGNET to translate the data from BCD (six-bit) to Prime ASCII before writing it to the disk file. This option is only useful for seven-track tapes.

BINARY — Instructs MAGNET to write the data verbatim to a binary disk file. The record size is the specified logical record size. Packing or unpacking only occurs if you are using a seven-track tape.

If you specify either EBCDIC or BCD translation, MAGNET asks if the entire record is to be translated or only selected fields. This permits bypassing translation of binary fields in EBCDIC records (output, for example, by COBOL). For partial record translation, MAGNET requests starting and ending column numbers of the data to be translated. This input is terminated with a null line (CR only). Finally, MAGNET requests the disk output file name. The following is an example of a typical READ procedure:

```
OK, MAGNET
[MAGNET, Rev. 19.0]
> READ
MTU # = 0
File # = 1
Logical record length = 80
Blocking factor = 10
ASCII, EBCDIC, BCD or BINARY? EBCDIC
Full or partial record translation? PARTIAL
Enter pairs of starting/ending column numbers, one pair
per line. Separate each column numbe with a comma.
Terminate entry with a null line (carriage-return)
only.
10,25
35,50
      (this line is a carriage-return)
Disk file: FILE.INPUT
>
```

4.1.4 The WRITE Subcommand

The WRITE subcommand is similar to the READ subcommand, except that the file on disk is written to the magnetic tape. WRITE also provides facilities for blocking and character translation.

The WRITE subcommand first asks for the magnetic tape unit and file numbers, and then for information on how the data format is to appear on the tape. As for the READ subcommand, you must specify the logical record size and the blocking factor. You must also provide the type of translation, if any. You specify one of the following options:

ASCII	Specifies no translation.
EBCDIC	Specifies EBCDIC translation onto tape.
BCD	Specifies BCD translation onto tape.
BINARY	Specifies a binary file.

If you specify EBCDIC or BCD translation, MAGNET asks whether the entire record or just certain fields are to be translated. (See the READ subcommand description for more information.) Finally, MAGNET requests the name of the input disk file. An example of a typical WRITE procedure is as follows:

```
OK, MAGNET
[MAGNET, Rev. 19.0]
> READ
MTU # = 0
File # = 1
Logical record length = 60
Blocking factor = 20
ASCII, EBCDIC, BCD or BINARY? ASCII
Disk file: SUEFILE
>
```

4.1.5 The COPY Subcommand

The COPY subcommand copies a file (or files) from one magnetic tape to another. No character translation is provided. The magnetic tape transports may be either seven- or nine-track.

MAGNET requests the FROM and TO tape units, starting file numbers, and the number of files to copy. To copy an entire tape, you must enter a large number (EOT is detected and causes the copy operation to halt). The followng is an example of a typical COPY procedure:

```
OK, MAGNET
[MAGNET, Rev. 19.0]
> COPY

*** "FROM" tape information: ***

MTU # = 2
Starting file # = 5

*** "TO" tape information: ***

MTU # = 6
Starting file # = 9

 # of files to copy = 27
Print record sizes? NO
>
```

4.1.6 Notes on Pre-Rev 18.4 Features

- All interactive commands now return to MAGNET subcommand level, NOT TO PRIMOS LEVEL as in the pre-Rev 18.4 software. You must use the QUIT subcommand to return to PRIMOS command level.

- Pre-Rev 18.4 MAGNET printed messages indicating completion of READ, WRITE, COPY, or POSITION operations. New MAGNET does not print these messages.

- When EOT is detected, the new MAGNET REPLY mechanism is now used. In addition, all MOUNT and DISMOUNT messages are directed to the operator's console unless you explicitly use the -USER command line option.

- MAGNET now retries all I/O operations up to ten times in case of error. Prior to Rev 18.4, MAGNET indicated that an error had occurred and whether or not it was a recoverable error. New MAGNET does not supply you with this information. In addition, all error messages have changed (see Section 7).

- It is recommended that you modify any existing command or CPL files to take advantage of the new MAGNET subcommands. The old (pre-Rev 18.4) subcommand forms are provided for compatibility only.

4.2 ADDITIONAL NOTES

- User labels are not yet supported.

- In the new forms of the READ, WRITE, and MOVE subcommands, the default translation code used is Prime ASCII to Industry Standard ASCII for output, and Industry Standard ASCII to Prime ASCII for input. The default translation is Prime ASCII to Prime ASCII for both input and output when you use the old forms of the READ and WRITE subcommands.

- Physical record lengths (LRECL * BFACTOR) should be at least 20 characters.

- ACCESS codes are recognized, but are not processed or verified.

- Pre-Rev 18.4 MAGNET did not write a filemarker after EOT was detected. This functionality was non-standard. New MAGNET does write this filemarker on output, and expects to read it on input. Therefore, some tapes may be incompatible between the old and new versions of MAGNET.

- To read information from a tape verbatim, specify:

  FORMAT=(VAP/PRIME) for your source tape;
  FORMAT=(FIXED) for your destination disk file;
  (0*) translation for your source tape;
  LRECL=(the-same-large-number) for both source and destination;
  and use the MOVE subcommand.

- Use ASSIGN commands with -TPID options to get the first tapes mounted prior to invoking MAGNET.

- DISMOUNT messages repeat until a tape drive is offline and not ready. MOUNT messages repeat until a tape drive is both online and ready.

- Use the LABEL command to initialize a tape (see Section 6) prior to using labelled tape features within MAGNET.

- For MAGNET messages 152 and 153 (see Section 7), check for tape drives not assigned, powered-down, not ready, or offline.

- The best value to specify for the BUFFERS option is 3. A larger number will probably not produce any increase in speed. In fact, speed may decrease due to the larger amounts of storage accessed.

- Do not specify a BUFFERS option for the second or later objects in a chain. (For more information, see the descriptions of the PREVCHAIN and NEXTCHAIN options in Section 7.) Buffers that belong to the first object are reused for later objects.

- Tape objects should be reused for multiple tape files. In addition to opening files on tapes, MAGNET requires that the tapes themselves be opened. If you use several objects specifying different files on the same tape, the tape is rewound each time you specify a different object within a READ, WRITE, or MOVE subcommand. It is better, therefore, to use the MODIFY subcommand to change filenames or numbers prior to using different data transferral subcommands.

- All subcommand names may be abbreviated to their first few letters. The table below illustrates these abbreviations:

CLOSE	CL
COPY	C
DECLARE	DEC (alternatively, DCL)

DELETE	D
DISPLAY	DI
LIST	L
LOAD	LO
MODIFY	MOD
MOVE	M
NOISY	N
POSITION	P
QUIT	Q
READ	R
RENAME	REN
SAVE	S
SILENT	SI
TRANSLATE	T
WRITE	W

You cannot abbreviate options or option values. You can abbreviate responses to some of the questions asked by old forms of the READ, WRITE and POSITION subcommands. For example, to the "ABSOLUTE OR RELATIVE?" question within the old form (pre-Rev 18.4) of the POSITION subcommand you may respond with an 'A' or 'P'.

o FORMAT OPTION

The FORMAT option for disk objects has been expanded. A new option value now accepted is UNCOMPRESSED (i.e. FORMAT=(UNCOMPRESSED)). UNCOMPRESSED format, valid when writing to disk files only, is similar to FIXED format except that new-line characters are written at the end of every record. As in FIXED format, no blanks are compressed. UNCOMPRESSED format is the default format used when old-style (pre-rev. 18.4) READ subcommands are issued and partial translation is specified for EBCDIC or BCD files. Uncompressed files are read the same way as VAR/PRIME disk files.

o PAD OPTION

A new option, PAD, has been included for tape objects. This option is used when writing files to tape and specifies the padding character to use to fill a logical record.

The default padding character in new-style operations is an industry-standard blank character (octal 040). In old-style operations, the padding character selected is either a PRIME ASCII blank character (octal 240), an EBCDIC blank (octal 100), a BCD blank (octal 000) or, for BINARY files, zero (octal 000), depending on the user's reply given to the "ASCII, EBCDIC, BCD or BINARY" question asked by MAGNET.

The PAD option is specified in the following manner:

```
PAD=('ccc'T)
```

where the "c"s specify one or more characters and T is the type indicator. Note that the apostrophes MUST be specified.

There are eight padding types that can be selected. These include industry-standard ASCII, PRIME ASCII, BCD and EBCDIC as well as BINARY, DECIMAL, OCTAL and HEXADECIMAL numbers. These types are indicated by the single letters:

A	industry-standard ASCII
B	BCD
D	decimal
E	EBCDIC
H	hexadecimal
I	binary
O	octal
P	PRIME ASCII

Thus, to specify an EBCDIC blank as a padding character, one could use any of the following:

```
PAD=(' 'E)            (for EBCDIC)
PAD=('01000000'I)     (for binary)
PAD=('100'O)          (for octal)
PAD=('064'D)          (for decimal)
PAD=('40'H)           (for hexadecimal)
```

The type indicators (A, B, D, E, H, I, O or P) may be in either upper- or lower-case. If 'A', 'B', 'E' or 'P' is specified as a type, then only one character must appear between the quotes. If 'I' is specified, eight digits must appear between the quotes. If 'D' or 'O' is specified, three digits must appear between the quotes. If 'H' is specified, only two digits must appear between the quotes.

Note that if 'A', 'B', 'E' or 'P' is selected, upper- and lower-case characters are significant within the quotes. That is,

```
PAD=('K'e)
```

is not equivalent to

```
PAD=('k'e)
```

The first results in a padding value of octal 322, while the second results in a padding value of octal 222.

Because BCD is only a 64-bit code, lower-case characters specified map to upper-case pad characters.

Subject: MAGSAV,MAGRST

Release: 19.00

Date: August 19, 1982

MAGSAV and MAGRST for Rev19.00

Problems fixed:

1) The question "Do you want to rewind tape" (asked by MAGSAV if the second or subsequent physical reel of a save is not at load point) now takes its answer from the TTY stream if the command line option -TTY has been given.

2) This question is also asked by MAGRST if the second or subsequent reels of a restore are not at load point.

3) The question "Continue with this reel" (asked by MAGRST after loading a continuation reel of the wrong sequence number) takes its answer from the TTY stream if the command line option -TTY is used.

4) Erase and Kill characters can be used when inputting a tape unit number from the Terminal.

5) If the Tape unit is not assigned, the message

"Device not assigned. Type "S" to continue"

is output, followed by an error exit to Primos. If the unit entered was in error, typing "S" will repeat the "Unit number" prompt instead of repeating the Error message, thus allowing the save/restore to continue.

6) If a multi-reel tape was searched for a logical tape not on the first reel, when mounting the second reel, the message "Wrong tape" was produced This is because MAGSKP called VALIDA to validate the Reel label, without having the first label to validate against! MAGSKP now doesn't validate the reels loaded, it just stops at the appropriate tape.

7) POLER # 40596
When MAGRST opened the current UFD to see if it was the MFD, it didn't close it again afterwards.

8) POLER # 41766
An extra option has been added to Magsav, to enable it to save UFD entries on an incremental save. This is to enable a single incremental save to be restored into an empty UFD without having to restore the

original save. Note that UFDs will be saved even if none of the files in them have changed, and this will cause empty UFDs to be created on restore.

9) ***IMPORTANT***

If an unrecoverable write error occurs in MAGSAV, the error recovery action will be printed.

Error recovery in Magsav is to re-start the save at the last "checkpoint". There are 3 checkpoint types:-

1) at the beginning of the Logical tape.

2) at the beginning of the Continuation Reel.

3) at the last prompt "Name or Command"

Magsav can only recover the save to the CLOSEST checkpoint, which means that in the case where more than one answer has been given to the "Name or Command" prompt since the start of the current Reel/Logical tape then MAGSAV CANNOT RECOVER THE WHOLE SAVE, but can only restart using the last answer to "Name or Command".

This limitation has always been present, but has never been documented, and no warnings were produced. Magsav now recognises the recovery action possible and prints a message after an unrecoverable error. There are 3 messages to go with the 3 checkpoint types:-

1) If the reel number is 1 and there is only one "Name or Command" answer SINCE THE START OF THE LOGICAL TAPE then the message is "Restarting current logical tape".

2) If the reel number is > 1 and there is only one "Name or Command" answer SINCE THE BEGINNING OF THE CURRENT REEL then the message is "Restarting current reel (reel n)" where n is the current reel number.

3) If there has been more than one answer to "Name or Commard" during the current reel, then the message is

WARNING*
Unable to recover to beginning.
Restarting at name <answer to last prompt>
If you continue, you will have to keep the Reel which failed.

In this case, the failed reel will contain some of the save, and thus must be kept even though is is not complete. An alternative is to RESTART THE ENTIRE SAVE AGAIN.

10) Magrst now sets the DTM on files restored from tape correctly. It also sets DTM and protection on UFDs if they didn't already exist.

11) POLER 45426
The tape error handling in Magsav has been re-worked. Versions 2 & 3

of the tape controller (Integrated formatter, and GCR formatter) now use the ERASE command to recover from tape write errors. Versions 0 & 1 (Kennedy formatters) will continue to use File Marks for recovery.

12) The error recovery on labels has been improved. Error recovery will now be applied on every label, not just labels on the first logical tape. The number of retries when writing the label has been increased from 5 to 20. If the label is successfully written AFTER recovery has taken place, the number of retries will be printed. If the label cannot be written, the reason for failure will be printed.

New error messages for this scheme are:-

Unable to write File Mark.

Unable to Backspace.

Unable to find File Mark.

Unable to Erase tape.

Recovery retries exceeded.

Hardware status: <status> (except retries exceeded)

13) POLER #48252

The magtape boot has been changed as follows:-

The default speed of the bootstrap is now 300 baud instead of 110 baud as before.
If the file being restored is a DOS runfile (i.e. normally *DOS64) then setting switch 13 (add '10 on a VCP) will cause the boot to restore the file, relocate DOS, patch the speed settings from itself, and start DOS at the correct place.

Example:

```
CP> BOOT_15
TREENAME= *DOS64

PRIMOS II REV 19.0  01/22/82 (AT 170000)

OK:
```

14) POLER # 37054

The default for the $A command is now to search all disks, not just logical device zero.

15) abbreviation for -NO_ACL changed from -NA to -NOA

Mocifications_for_Rev_19.00

1) Quotas are saved by MAGSAV and restored cnto NEWLY CREATED ufd's by MAGRST. Existing UFDs do not have their Quota overwritten.

2) Default is now 2048 word variable size blocks (-VAR). To write 512 word fixed recoras use the option "-P300". To write 1024 word fixed size blocks, use -LONG.

Note that the option -P300 also supresses ACLS.

3) New option "-NO_ACL" (abbreviation "-NOA") suppresses the saving of ACLS and ACL references.

4) MACSAV will no longer run on an OLD partition.

5) MAGSAV will no longer produce Pre-rev12 tapes, the $OLD command has been removed.

6) The $VALID command has been removed.

7) Magsav now saves category acls, and acl references on objects and Magrst will restore Acls and references if they do not already exist.

8) Under Primos II, neither ACLS nor QUOTAS are saved OR restored.

9) Command line options in MAGRST and MAGSAV may not be abbreviated using the old "leftmost substring" algorithm.

10) Tapes made by Magsav have a new tape class which enables Magrst to identify the tape. Tapes made by a pre-rev19 Magsav will produce the message

(Pre-Rev19. Tape)

Tapes made by rev19 magsav using the -NO_ACL option (or under Primos II) produce the message:

(Rev19 tape without ACLS)

IMPORTANT

If either a Rev 19 tape without acls or a Pre-rev 19 tape is restored into an ACL protected tree, the password UFDs on the tape will be restored as Default_protected_ACL_UFDs.

When an acl'ed ufd with a password is MAGSAVed the password is ignored. This is a file system problem which is known about.

SUBJECT: MAKF

RELEASE: 19.0

DATE: January 22, 1982

NEW FUNCTIONALITY:

Make at rev. 19 will support the 34MB, 68MB, 160MB, and the 600MB in addition to the 80 and 300MB'S, CMD'S and floppies. The new drives have a variable number of cylinders and a new mechanism has been implemented to take this into account.

In order for MAKE to determine the number of cylinders used on a particular drive, it must ask through a series of questions for the drive type. The number of cylinders is stored in word 7 of the DSKRAT header for future use.

To provide diagnostic capabilities, the last 2 cylinders of a Winchester disk will be reserved. This is done transparently to the software. A file system partition is set up using 2 less cylinders. The RAT size and records in partition field are set up appropriately. Note: These tracks are not mapped out using the badspot mechanism.

Badspots will now be mapped out on a record basis in order to reduce the amount of lost storage due to mapping out a complete track. This will happen automatically when MAKF finds a bad record in its write or verify phase. If badspots are being entered by hand, MAKE will prompt for record based badspots.

For support of the 40 head 600MB drive, MAKE will now use the full 5 bits in the number of heads field of the PDEV. This allows a maximum of 31 heads in a partition.

A worst case test pattern is now used in the data portion of a record to more carefully screen for badspots.

In the case of a split paging partition, the whole partition will now be initialized, not just the file system portion. This will generate a useful badspot file for use by the preloader.

A new command argument (-AUTO) has been added in order to generate a badspot file without the need for user intervention. This ability is needed because of the increased number of possible badspots, possibly in the hundreds.

The total number of records lost to badspots will be reported at the end of the make.

PROBLEMS FIXED:

The PACK NAME will now be converted to upper case before being written to the pack.

MAKE will continue if BADSPOT not entered into BADSPT file.

The number of badspots will be reset when PARAMETERS OK? question is NO.

OUTSTANDING PROBLEMS:

none.

ENVIRONMENT:

Normally used on a REV 19 system , must use OLD option in order to generate an old style badspot file.

BUILD and INSTALL PROCEDURE:

standard

FIXRAT

1. Modified to ignore Rev. 19 ACL's and QUOTA's.

2. Modified to conform to Rev. 19 Master Disk Standards.

* NOTE: All products have been modified to conform to master disk standards. For a description of these modifications, please read INF019>STANDARDS.RUNO.

MAKE

Clarified questions asked about physical device type.
"80 OR 300MB STORAGE MOD" -> "STORAGE MODULE OR CMD"
Inform user these are YES or NO questions.

Subject: PHYSAV. PHYRST

Release: 19.0

Date: August 19, 1982

1 New Functionality

1.1 Badspot Handling

A new format BADSPT file has been defined in which an individual bad record can be flagged, rather than the whole track containing the bad record. PHYSAV has been modified to use this new format.

Badspot handling has been added to PHYRST so that records are not written to badspots but are mapped to the first available free record on the target partition.

In order to ensure that badspots are handled correctly, the following guidelines should be followed:
1) The Record Availability Table for each source partition should be correct. To ensure this is so, FIX_DISK can be run.
3) There should be enough free records on a source partition being saved for records falling on badspots an the target partition to be relocated to.
3) To be safe, it would be wise to keep copies elsewhere in filestore of all BADSPT files in case of accidental loss.

To clean up a disk after badspot handling has taken place, the FIXRAT replacement FIX_DISK must be used. If FIX_DISK is not available then badspot handling must not be performed. The new command line option -NO_BADS has been provided in PHYRST to turn off badspot handling.

Example: PHYRST -NO_BADS

1.2 New Option to PHYSAV

A new option, -LOWEND, has been added to PHYSAV at rev 19.0 to enable users to gain maximum performance. This option should be specified when running PHYSAV on any machine below a P750.
N.B. If -LOWEND is not specified for a machine below a P750 then a performance degredation will occur.

2 Fixes

POLFR 32189 has been fixed at rev 19.C. PHYSAV now works when a user uses REN after having broken-in when asked the question 'WRITE NEXT LOG. TAPE (YES/NO)?'.

PHYSAV can now write more than one logical tape when used with an Integrated Formatter.

Physav will now save partitions of up to 40 heads (600 support).

Physav will now cope with EOT when performing a GAP operation during error recovery and continue correctly onto the next tape with the same logical tape number.

Physav will now work when user types REN after having broken in when given the message 'END OF REEL. MOUNT REEL' and then 'UNIT NO:'

Phyrst will not now tell user to run fix_disk when in verify mode.

3 Error Messages.

'EOT DETECTED WHILE DOING ERROR RECOVERY WILL CONTINUE WITH SAVE'

IF EOT is detected while Physav is recovering from an unrecoverable error then the current record will be saved and written on to the next reel. Physav will write the trailer labels on the current tape and ask for the new tape and continue.

'IF YOU DO NOT WISH TO CONTINUE WITHOUT BADSPOT
HANDLING YOU WILL NEED TO RE-MAKE PARTITION xxxxxx'

This message is sent whenever the target partiton cannot accommodate the source partition (usually occurs when the source was full and the target has more badspots than the source). This message will appear in conjunction with the message -

'NO FREE RECORDS ON PARTITION xxxxxx'

ABSTRACT

This document describes the Changes made to PMA for PRIMOS release 19.0

1 PMA

There is one bug fix to PMA for rev 19.0. PMA will now correctly handle Source file names with passwords in the directory portion of the treename.

2 P850 Support

The following mnemonic Opcodes are now implemented:
ENBL, ENBM, and ENBP,
INHL, INHM, and INHP.

3 P300 Support

The FRAC instruction has been removed from PMA at rev 19.0. The instruction
only existed on the P300 with special hardware and was never implemented on
the P400,P500, or the 50 series.
Support for the P300 was dropped at rev 15.0

This document contains all new functionality and POLER bug fixes for Revision 19.0 of PRIMOS. Certain portions of this document must be read in order to upgrade from Rev 18 to Rev 19 (eg. section 1.3, "Conversion to revision 19 of PRIMOS").

3 Switch 4 up, Switch 5 up. Fully automatic.

Physical device is automatically started up from front panel switch setting as above.

Primos is then automatically brought up from the pathname saved in the PRIMOS command.

1.1.4 Example of Coldstart using Device 460

Assume Primos runfiles are in a directory called OPSYS.

- o Power on.
- o Turn rotary selector to Stop/Step
- o Master clear.
- o Turn Address/Data switch to Address.
- o Set %10114 in the sense switches (switches 4, 10, 13, 14 up).
- o Turn selector to load.
- o Press Start.
- o Turr selector to Run.
- o Type PRIMOS OPSYS on the system console.

To reboot the system:

- o Turn rotary selector to Stop/Step
- o Master Clear
- o Turn Address/Data switch to Address.
- o Set %14114 in the sense switches (switches 4, 5, 10, 13, 14 up).
- o Turn selector to Load.
- o Press Start.
- o Turn selector to Run.

1.1.5 Paging Space Requirements

Prior to Revision 19, space on the main and alternate paging disks was allocated in segment-sized (128KB) blocks, except for Operating System Kernel Segments which were truncated to the 16KB boundary nearest their actual size.

This resulted in wasted space due to the fact that most segments are much smaller than 128KB.

At Revision 19, kernel segments are allocated paging space in the same way (truncated to 16KB boundary nearest their actual size).

All other segments are allocated space on demand in units of 16KB. This means that if one never accesses beyond the first 8 pages of a segment, for example, only 16KB of paging space is ever used up by the segment, a savings of 112KB over Rev. 18.

This change should make it possible to support many more segments within a given paging disk size than under Rev. 18, provided most of the segments are smaller than full size.

This change does make it much more difficult to compute the amount of paging space required to support a given user load, because it may not be known how large the segments will be even if it is known how many segments there will be.

It is possible to compute worst-case requirements, as follows:

space for 24 kernel segments = 1536 records max.

space for (nusr) ring 0 stacks = (nusr)*16 records max.

space for (nusr) ring 3 stacks = (nusr)*64 records max.

space for N user segments = N*64 records max.

total space required = (1536+(nusr)*80 + N*64) records max.

Note that the total number of segments per system remains as 1022 (including all kernel segments). The maximum number of segments per user remains configurable (via NUSEG) up to 240.

1.2 PRIMOS SOURCE DIRECTORY

This section describes the structure of the Primos source directory. Currently it is:

```
PRIMOS  (ufd)

    (sub_directories)
     KS      kernel operating system source
     FS      file system source
     R3S     ring 3 command environment source
     CPLS    command procedure language source
     NS      networking source
     NPXS    ring 3 networking source
     CS      communication source
     ES      emulator source
     PJES    RJE emulator source
     PS      Primos source for pl1 functions.
     INSERT  insert files for Primos source modules

    (utilities)
     VPSD      Primos vpsd command source
     PRMLD     Primos pre-loader source
     MAPGEN    Primos mapgen utility source
     FIND_OBJ  Primos utility for build
     USAGE     Primos Usage command source

    (files)
     (Segment run-file images)
```

```
PR0000, PR0004, PR0005, PR0006
PR0010, PR0011, PR0012, PR0013
PR0014, PR0015, PR0022, PR0032
PR0033, PR0034, PR0037, PR0040,
PP0067, PR0070, PR0141, PR0142.
PR6000, PR6003

*COLDS       Cold start run-file image.
PRIMOS       Pre-loader run-file.
RING0.MAP    Seg map for Ring 0 Primos
RING3.MAP    Seg map for Ring 3 Primos
RING0.LOAD   Template for Ring 0 Primos load
RING3.LOAD   Template for Ring 3 Primos load
VERSION_STAMP.CPL  prints version number of Primos build.
(build tools descibed below)
```

1.2.1 Build_Tools

The basic procedures for building PRIMOS have been converted to CPL build tools. All tools are designed to be executed while attached at the UFD level.

1.2.1.1 PRIMOS.BUILD.CPL

Usage: R PRIMOS.BUILD {version_number} {-load}

This tool will re-build all of Primos operating system and associated utilites. New cold start run-files and maps are also created. The default version number is 19.0; and the output file PRIMOS.COM0 is created. The '-load' option will perform only the load procedure for both ring 0 and ring 3.

1.2.1.2 COMPILE.CPL

Usage: R COMPILE {source_tree} {-no_como}

This utility is a compiling tool tailored for a PRIMOS directory. If source_tree is omitted then all the standard Primos source sub_directories described above are compiled. If source_tree is solely a filename all sub-directories are searched for the file. If the language suffix is omitted on the filename then a search is performed for file.(PMA FTN PLP). Also, any unclaimed arguments are used as compiler options overriding the default of ' -b *>obj>file.BIN -l no '.

```
E.g.  r compile ks
      r compile ks>ainit.ftn
      r compile ainit
      r compile ks>ainit    -b no -l ainit.list
```

1.2.1.3 LOAD.CPL

Usage: R LOAD <datafile> -ring {0, 3} {-version version_number}

This tools performs the load for a Primos build.

The datafile is a template used for the Primos load, defaults are RING0.LOAD and RING3.LOAD.

1.2.1.4 MOVSYS.CPL

Usage: R MOVSYS <source_pathname> <destination_pathname>
{-all} {-opsys}

This tool is tailored to move parts of the PRIMOS ufd. The default is '-opsys' which moves all run-files (PRxxxx), PRIMOS, *COLDS, VERSION_STAMP.CPL, RING0.MAP, and RING3.MAP. Both source and destination pathnames must be specified.

1.3 CONVERSION TO REVISION 19 OF PRIMOS

1.3.1 Introduction

At revision 19 of Primos, many new and powerful features are provided to the system administrator which allow him or her to control use of the system's resources more stringently and accurately than ever before. Furthermore, many user convenience features have been added to Primos, allowing both novice and experienced users to utilize more of the power of the Prime processor. Unfortunately, as with any changes of this nature, some conversion effort on the part of the system administrator and site operators is required. This section describes the use of the conversion tools provided by PRIME at rev 19. The tools, along with the procedures described below, should be followed as closely as possible for best results.

1.3.2 Master Disk Installation Instructions

For general information on installation of a new Master Disk Release please reference INF019>INSTALL.RUNO. Note that the conversion tools will perform many of the tasks described in this document automatically.

1.3.3 What is required

Before you begin the conversion, you should have the following things immediately available to you:

- A system currently running revision 18.3 or 18.4 of PRIMOS.

- A disk partition onto which revision 19 software can be loaded.

- MAGSAV tapes containing (1) the TOOLS directory (see section 3.2), (2) the U1 partition of the Master Disk (see section 3.4), and (3) those portions of the C1 partition which are in use at your site.

- A blank magnetic tape or spare disk partition for backup purposes.

- A file containing a list of top-level UFD names which you do NOT want to be "loginable" (the "exception file").

- This document, or the Installation/Conversion Guide (DOC-xxx-190).

- The System Administrator's Guide (DOC-5037-190), specifically those portions dealing with EDIT_PROFILE, FIX_DISK, and MAGSAV/MAGRST.

In addition, before starting you should be familiar with:

- This section of this document.

- EDIT_PROFILE, the revision 19 utility which supports User Profiles.

- FIX_DISK, the revision 19 utility which supersedes FIXRAT.

- MAGSAV and MAGRST, the PRIMOS logical backup utilities; PHYSAV and PHYRST, the PRIMOS physical backup utilities; or COPY_DISK, the PRIMOS physical disk copy utility.

1.3.4 Conversion Procedure

This section describes the actual conversion procedure. It is extremely important that it be followed in the order given, and that no steps not specifically indicated as optional be omitted.
Most of this procedure applies to initial installation of revision 19 as well as to conversion from rev 18 to rev 19. Portions of the procedure which may be omitted during initial installation are so marked.

1.3.4.1 Step 1 -- Back up existing partition

If you are going to use the existing command partition to run rev 19, you should make a backup copy of the current command partition in case problems are encountered during the conversion. You may do this using any of the PRIMOS backup utilities: MAGSAV, PHYSAV, or COPY_DISK.
This step may be omitted if either (a) you are using a different partition for rev 19 than your existing command partition, or (b) this is an initial installation.

1.3.4.2 Step 2 -- Load the TOOLS directory

The TOOLS directory contains the tools which make converting from rev 18 to rev 19 easier. These tools include:

- o CONVERT_PROFILE to generate the User Profiles database
- o LOAD_19 to load revision 19 Master Disk software
- o CONVERT_ACLS to protect PRIMOS directories with ACLs

You should load this directory from the first logical tape, which should have the name TOOLS, onto the command partition using MAGRST.

1.3.4.3 Step 3 -- Run CONVERT_PROFILE

CONVERT_PROFILE is contained in the TOOLS directory. It is used to generate input for EDIT_PROFILE, the PRIMOS Profile Editor. This input tells EDIT_PROFILE who the legal users of your system are, and what their initial attach points (login UFDs) and login passwords are. CONVERT_PROFILE is designed to do this for you as automatically as possible. It is invoked with the following command line:

```
R TOOLS>CONVERT_PROFILE [<packname_list>] [-No_Query] [-DeBug]
                        [-Mfd_PassWd <password>]
```

If no <packname_list> is given, only logical device zero (the command partition) is converted. Otherwise, all partitions listed will be used to create the list of legal users.
If neither password on the MFD of logical device zero is "XXXXXX", you must supply the owner password for the MFD. This is done with the -MFD_PASSWD option. If at least one of the passwords is "XXXXXX", you need not supply this parameter, but CONVERT_PROFILE will prompt you for the correct owner password. CONVERT_PROFILE will always prompt you for the MFD owner passwords of any other partitions which you are converting.
The -NO_QUERY option prevents CONVERT_PROFILE from asking you to supply alternate user IDs for any illegal IDs which are encountered during the run. If -NO_QUERY is specified, all illegal IDs will be skipped, but warning messages will be printed for each one found. Note that user IDs may contain only letters, digits, and the special characters ".", "$", and "_", and must begin with a letter.
The -DEBUG option causes CONVERT_PROFILE to execute in a test mode, writing files into the current directory rather than TOOLS. Also, a detailed version number which may be of use to your analyst will be displayed. Normally, you should not use this option.
CONVERT_PROFILE will ask you various questions in order to properly set up the rev 19 User Profiles database. The questions which will be asked are:

MFD owner passwords for each partition to be converted, plus that for the command partition regardless of whether it is being converted. These are required so that passwords may be read.

The System Administrator name. At rev 19, a special user called the System Administrator (SA) exists. This user is the most powerful user on the system aside from user 1. Responsibilities of the SA include administration of the User Profiles databases and allocation of system resources, as described in the System Administrator's Guide.

The name of an "exception file." This is a file, created in the editor, which contains a list of all top-level UFDs which you to not want to be legal user IDs on your rev 19 system. These UFD names should be entered one per line. Note that PRIME-supplied directories such as CMDNCO and MFD are automatically skipped by CONVERT_PROFILE, so if you do not have any directories of your own which you do not want included, you need not create an exception file. (SYSTEM, however, is normally included in the list of legal users, so if you do not want it to be loginable you should put it into an exception file.)

When CONVERT_PROFILE first starts up, you may see some error messages referring to files not being found by DELETE. These errors are a normal part of CONVERT_PROFILE's cleanup processing, and should be ignored.

As CONVERT_PROFILE progresses, it may encounter illegal or duplicate user IDs (top-level directory names). Duplicate names are always skipped; that is, the first UFD with a given name is always the one chosen. When an illegal name (one containing characters other than letters, digits, ".", "$", or "_") is found, you are given the option of supplying an alternative name or skipping the illegal ID, unless you specified the -NO_QUERY option, in which case the illegal ID is automatically skipped.

When CONVERT_PROFILE has finished, a CPL program called PROFILE_CONVERSION will be placed into the TOOLS directory. This program contains input for EDIT_PROFILE, and must be executed by you under rev 19 before any users will be allowed to login.

You may execute CONVERT_PROFILE at any time before bringing up rev 19. You may execute it more than once for testing purposes if you so desire. It is recommended, however, that your final run take place immediately before the conversion so that the most up-to-date information is passed to EDIT_PROFILE.

This step will be omitted during initial installation.

1.3.4.4 Step 4 -- Load rev 19 Master Disk software

Prior to rev 19, you may have loaded PRIME software piecemeal--that is, PRIMOS at one point, compilers at another, external commands at yet another, etc. Because of the complexity of features at rev 19, this is no longer permissible. During this step, you MUST load the following software:

- o PRIMOS (PRIRUN directory)
- o DOS (PRIMOS II, DOS directory)
- o All external commands (CMDNCO, SEGRUN* directories)
- o Shared segments (SYSTEM directory)
- o Libraries (LIB, SYSCOM, SYSOVL directories)

- o HELP files (HELP* directory)
- o BATCH (BATCHQ directory)
- o SPOOLER (SPOOLQ directory)
- o RJE (RJSPLQ*)
- o PRIMENET software, if you have it (PRINET/PRIMENET* dirs)

The LOAD_19 tool loaded in Step (2) will correctly load all the non-chargeable software listed above. You use LOAD_19 by typing:

```
R TOOLS>LOAD_19 [-MFD_path <pathname>] [-DRIVE <drive_num>]
                [-DeBug]
```

If neither of the MFD passwords is "XXXXXX", or if you want to load your software onto a partition other than logical device zero, you must specify the correct pathname of the MFD, including partition name and owner password, with the -MFD_PATH option.
The -DRIVE option indicates the magnetic tape drive number on which the tape containing the U1 partition is loaded. If this option is not given, LOAD_19 will ask you for the drive number.
The -DEBUG option provides analysts with a detailed version number which may prove useful for debugging purposes. It is not normally used.
LOAD_19 serves two purposes: First, it loads the Master Disk software required to run revision 19. Secondly, it makes sure the names of all software in non-chargeable directories complies with the new standard for filenames (suffixes) which is supported at rev 19. In particular, all R-mode run files (except external commands which run under PRIMOS II) must end in ".SAVE", and all library files must end in ".BIN". Note that since this is done before the rev 19 software is loaded, your rev 18 software will have the .SAVE and .BIN suffixes. This should not affect operation of rev 18.
The tape loaded in this step should be named M190U1.

1.3.4.5 Step 5 -- Load PRIMENET software

If you use PRIMENET, you MUST load its revision 19 software before attempting to bring up rev 19. You should do this by reading the tape containing the PRIMENET software with "MAGRST.18". NOTE that since MAGRST is now a rev 19 MAGRST, and since rev 19 MAGRST will not run under rev 18, your old MAGRST will be renamed MAGRST.18 so that you can restore additional tapes after the Master Disk software has been loaded. (If you are loading rev 19 software onto a spare partition, this will not be true; you may continue to use MAGRST.)
If you are converting from rev 18 to rev 19 and already have a PRIMENET* directory on your command disk, you should attach to PRINET and type

```
COMI PRINET.INSTALL.COMI
```

If you are installing rev 19 initially, or do not yet have a PRIMENET* directory, attach to PRINET and type

```
COMI PRINET.INITINSTALL.COMI
```

1.3.4.6 Step 6 -- Boot Revision 19

At this point all software has been loaded, and you should be ready to go. Bring down your rev 18 system in the usual fashion, and boot rev 19. In order to use the new automatic boot facilities of rev 19 to go directly to PRIMOS, type

```
BOOT 14114
```

instead of BOOT 114. This will boot PRIMOS from the PRIRUN directory. You need not enter the physical device number or type "R PRIMOS" under PRIMOS II. Note, however, that BOOT 14114 assumes that you boot your system from the top partition of drive zero, controller zero. If you normally boot from another device, or have been using a partition other than the top partition of a device for booting PRIMOS, you should refer to the System Operator's Guide for information on how to boot from other devices.

1.3.4.7 Step 7 -- Execute PROFILE_CONVERSION

You are now running rev 19 PRIMOS. Before users will be allowed to login, however, the User Profiles database must be set up. Recall that you generated input to accomplish this back in Step (2) with CONVERT_PROFILE. Now you complete the process by typing

```
R TOOLS>PROFILE_CONVERSION
```

After you have done this, your rev 19 system is up and running normally. Users will see virtually no difference in terms of logging in and using the system. All directories are still password directories, and disk Quotas are not enabled. In order to enable use of ACLs and Quotas, you must continue to Steps (8) and (9).

1.3.4.8 Step 8 -- Use FIX_DISK to convert partitions to Rev 19

Before you can use ACLs and/or Quotas, you must use FIX_DISK to format your disks for rev 19. You do this by invoking FIX_DISK with the -FIX and -CONVERT_19 options. (We recommend that you also use the -DUFE and -CMPR options, but you need not.) You must do this for each partition which you want to support ACLs and Quotas. If you are converting the command partition, you must use the -COMDEV option as well.

NOTE

Under NO circumstances should you use FIXRAT under rev 19. FIX_DISK will work correctly on rev 18 partitions; we recommend that it be used all the time, regardless of whether or not partitions have been converted to rev 19 and regardless of whether or not they have ACLs or Quotas on them.

1.3.4.9 Step 9 -- Convert to ACLS with CONVERT_ACLS

The last tool in the TOOLS directory which you will need is CONVERT_ACLS. This routine converts the MFD and all non-chargeable directories to standard ACL protection. The SAD (User Profiles database) and BATCHQ are converted by EDIT_PROFILE and BATCH's INIT tool, respectively. Both these utilities are invoked automatically by CONVERT_ACLS.
CONVERT_ACLS is invoked as follows:

```
R TOOLS>CONVERT_ACLS [<partition_names>] [-Mfd_PassWd <pw>]
                     [-DeBug]
```

If a list of <partition_names> is given, the MFDs of each of the named partitions is converted in addition to the command device. If no list is given, only the command disk is converted.
The MFD password need be supplied only if neither the owner nor the non-owner password of the MFD on the command device is "XXXXXX".
The -DEBUG option prints detailed version information for your analyst. It should not normally be used.
CONVERT_ACLS will ask only for the System Administrator name. The SA name must be the name given to CONVERT_PROFILE.
CONVERT_ACLS may be run only from the system console or by the registered System Administrator. Note that the System Administrator is not registered by PRIMOS when PROFILE_CONVERSION is first run. In order to register the System Administrator the first time, PRIMOS must be rebooted. Once the initial System Administrator is registered, however, subsequent changes to the System Administrator name (made through EDIT_PROFILE) will be recognized immediately.

2 NEW FEATURES

2.1 ACCESS CONTROL LISTS

A completely new method of protecting files in the file system has been implemented. Called "Access Control Lists" (ACLs), the method allows a list of users and access rights to be associated with any file, specifying who may use that file and in what ways.

We believe this to be a far superior method of protection compared to the directory password scheme, and that it will be to most users' advantage to convert as soon as possible.

Passworded directories continue to be supported, and may be converted to ACL directories on an individual basis using tools we have provided.

Refer to Chapter 5 for complete details.

2.2 CHANGES TO ATTACH SCANS

This section describes the changes in the way attach scans work at rev 19. To review, an attach scan takes place when a call is made (1) to AT$ANY (see above), (2) to AT$ with a pathname which begins with a directory name, or (3) to ATCH$$ with an LDEV of '100000 and the K$IMFD key. Attach scans also occur when the ATTACH command is given with a pathname which begins with a directory name.

2.2.1 Rev 18 attach scans

On rev 18 systems, attach scans stopped whenever an error was encountered or an object with the given name was found in one of the MFDs. Remote portions of the scan, once initiated on a given system, did not stop unless the directory was found (that is, all errors were ignored). Thus, rev 18 attach scans were inconsistent: if an error was encountered locally, the scan was halted; if a network error occurred during the remote part of a scan the scan was halted; however, if an error occurred on a remote system as part of the scan, it was ignored.
Thus, at rev 18, attach scans could stop with a variety of errors, including E$FNTF (not found), E$NTUD (not a UFD), E$NETE (network error), and E$BPAS (bad password).

2.2.2 Changes at Rev 19

In order to make attach scans more logical and consistent, at rev 19 the following changes were made:

1) Both local and remote scans stop on only two types of error: E$NPIT (Insufficient access rights) and F$BPAS (Bad password). All other errors are ignored.

2) MFDs to which the user does not have Use rights are skipped.

3) If the attach does not succeed for any reason other than the two mentioned above, error code E$NFAS (Top-level directory not found or inaccessible) is returned.

2.2.3 How an attach scan works at rev 19

As a result of the above changes, at rev 19 attach scans work as follows. Local disks are always searched first, in logical device number order. If the directory is not found on any local partition, remote disks are searched, also in logical device number order. The search terminates when the directory is found and (a) can be accessed, (b) cannot be accessed because of insufficient access rights (E$NRIT), or (c) cannot be accessed because of a bad password (E$BPAS). This means that it is possible for an object with the given name to be found and the scan not halted. Such action will occur if (1) the object is not a directory, (2) the user does not have List access on the MFD containing the object (resulting in error code E$NINF), (3) one of the MFDs cannot be accessed for some reason, or (4) a network error of any kind occurs during the remote part of a scan. If all MFDs are scanned and the directory was not found or an error of type (1)-(4) occurs, error code E$NFAS (Top-level directory not found or inaccessible) is returned.

Note that changing the order of the disks in the logical device list may alter the results of an attach scan, and further that the error codes returned by AT$ABS (which searches only a single MFD) will in some cases not be the same as those returned by AT$ANY for the "same" directory. The reason these differences exist is that when an attach scan is initiated there is no way of knowing what partition the user wishes to search, and therefore all errors must be non-specific.

Network errors occurring during the remote portion of an attach scan are ignored. The search continues with the first disk in the list which is on a system different from that on which the error occurred.

MFDs for which the user does not have Use rights are skipped during the attach scan.

2.2.4 Impact

The most significant changes at rev 19 involve (1) MFDs on which the user is missing Use rights and (2) UFDs on which the user is missing Use in MFDs on which the user does not have List. In the case (1) the MFD will not be searched at all, while in case (2) the scan will not stop because it would have resulted in an E$NINF (No information) error, which is not one of the two errors which terminates a scan.
Some examples may help illustrate these problems. In both cases, assume that ldev 0 is called CMD and contains a UFD named MYUFD, while ldev 3 is called HISDSK and also contains a MYUFD. Depending on the accesses available on <CMD>MFD, <HISDSK>MFD, and the two MYUFDs, the results of the scan will be as follows (rights listed are <u>minimum</u> rights):

	<CMD>	<HISDSK>	<CMD>MYUFD	<HISDSK>MYUFD
Rights:	U	---	U	---
Attached to:			(here)	
Rights:	U/NONE	U	NONE	U
Attached to:				(here)
Rights:	U/NONE	U/NONE	NONE	NONE (Error E$NFAS)
Rights:	LU	---	NONE (Error E$NRIT)	---

2.3 ASSIGNABLE AMLC LINE IMPROVEMENTS

2.3.1 Introduction

Primos operating system has long supported the concept of "assignable amlc lines". This functionality allows a user to own and control the I/O for a particular amlc line. In many installations, peripheral devices such as printers, plotter etc. are used via assignable amlc lines.

This document describes improvements to the internal mechanism within Primos for assigned lines. This document is written for readers already familiar with the assigned amlc line functionality of PRIMOS. Please refer to Primos command manual and the System Administrator's Guide for additional information.

2.3.2 Enhancements:

A) Currently the only method to assign a line is via the Primos ASSIGN command. Some applications required the user to type the ASSIGN command for the desired amlc line, then execute the application. Currently user programs achieve the same result, internally, by calling CP$('ASSIGN AMLC...').

A new direct entrance call, ASNLN$, has been added thus allowing user programs to request the assignment of a line directly.

B) The meaning of the internal table LBT (line to buffer) has changed. Currently the LBT entry contains the line number and current owner of the line. The LBT now contains the state of a line regarding it's assignability. The table is indexed by line number.

The table entry is defined as follows:

-1 => line is not assignable. (Normal Primos user at command level)
 0 => line is assignable by any user.
>0 => line has been assigned, LBT entry contains owner.

1 CONFIGURATION AND OPERATIONAL MODIFICATIONS

1.1 BOOTSTRAP PROCEDURE

At REV 19, Primos may be coldstarted using a procedure that takes the system from depressing the start switch to Primos in one step. This procedure uses additional front panel switch settings (switches 4 and 5) and a new command in CMDNC0 (PRIMOS).

1.1.1 Introduction

At REV 18, three software systems are used during coldstart. They are Boot, Primos II and Primos. At REV 19 a fourth system has been introduced, the PRIMOS command. It is installed in CMDNC0 and is instrumental in simplifying the coldstart procedure. Subsequent sections of this document specify in detail the software required and procedures to be followed to perform a simplified coldstart at REV 19.

1.1.2 Software Required

1 Boot - must be from a REV 19 Master Disk or created by a REV 19 MAKE.

2 Primos II - must be REV 19. Must be installed in DOS>*DOS64.

3 PRIMOS command installed in CMDNC0.

4 Primos runfiles installed in a directory on the partition to be coldstarted.

1.1.3 Use of Front Panel Switches 4 and 5

1 Switch 4 down, switch 5 down. No change from REV 18 procedure.

2 Switch 4 up, switch 5 down. Do not prompt for 'Physical Device ='.

2.1 Front panel switches are interrogated by software and the device is automatically started up. For example, if coldstarting from physical device 60, switch setting 10114 will startup disk 460, 1060, etc. e.g. you cannot start up disk 20060 this way. Switch setting 10134 will start up disk 660, 1260, etc. Note this may be used only with the top (head 0) partition on a disk.

2.2 When the system prompts 'OK:', it is running Primos II. At this point the PRIMOS command is used to bring up Primos. The command is issued as PRIMOS <pathname>, where <pathname> is the pathname of the directory containing the run files for Primos. The Primos command remembers the pathname so the next time typing just PRIMOS is sufficient. Initially the pathname defaults to PRIRUN.

The current method of making a normal user line an assignable line will continue to work properly, i.e., the system administrator manually sets the buffer number in the LWORD entry to zero. The action will now also change that line's LBT entry to zero thus making it assignable by any user.

2.3.3 Interface: ASNLN$

Name: ASNLN$ (Assign amlc line)

Usage:

```
call asnln$ (key, line, protocol, config, lword, status)

dcl asnln$  (fixed bin, fixed bin, char(6), fixed bin, fixed bin, fixed
bin);
```

Description

This routine is a new direct entrance call available to users. It performs the assignment and unassignment of amlc lines for a respective caller. A user may own more than one assigned line. The caller may also set line characteristics, protocol, etc. This routine will only allow a caller to assign a line that has a corresponding LBT entry of zero, i.e. the line is assignable. The buffer used for the assigned line is dynamically chosen within ASNLN$.

(refer to System Administrator's Guide for protocol, config, and lword values)

status
: error status returned to caller.

key
: 1 => assign amlc line.
: 0 => unassign amlc line.
: 2 => unassign all amlc lines owned by caller.

line
: Desired line number.

protocol
: Desired protocol. Blanks indicate no change desired.

config
: Desired config setting. A zero indicates no change desired.

lword
Desired line characteristics. The buffer number used for the line cannot be changed by a ring 3 user program using this interface.

2.4 ASSIGNing Magtapes

At rev's 18.4 and 19.0 the format of the Primos ASSIGN command for magtapes has changed:

```
FROM:     ASSIGN MTn [-ALIAS MTm] [<options>]
          ASSIGN MTX  -ALIAS MTm  [<options>]
           options: [ -WAIT ]
                    [ -MOUNT ]
                    [ -TPID <id> ]
                    [ -7TRK | -9TRK ]
                    [ -RINGON | -RINGOFF ]
                    [ -800BPI | -1600BPI | -6250BPI ]

TO:       ASSIGN MTn [-ALIAS MTm] [<options>]
          ASSIGN MTX  -ALIAS MTn  [<options>]
           options: [ -WAIT ]
                    [ -MOUNT ]
                    [ -TPID <id> ]
                    [ -7TRK | -9TRK ]
                    [ -SPEED {25 | 100} ]
                    [ -RINGON | -RINGOFF ]
                    [ -DENSITY {800 | 1600 | 3200 | 6250} ]
```

That is, the options -800, -800BPI, -1600, -1600BPI, -6250, and -6250BPI are no longer recognized. Attempts to use them will result in an error message which lists the above new format. They have been replaced with the new -DENSITY option, which requires a tape density in bpi (bits per inch) as an option argument. Currently only 800, 1600, 3200, and 6250 are legal option arguments to the -DENSITY option. This change was made to comply with the new Prime Command Arguments standard as well as to add 3200 bpi density support for future drives in a compatible manner. Note that the ASSIGN command does not actually check that the drive being assigned is capable of the desired density.

The option -SPEED, which requires a tape speed in ips (inches per second) as an option argument has also been added to support future drives. Currently only 25 and 100 are supported as legal option arguments to the -SPEED option.

2.4.1 Model 3 1600/6250 BPI Capable Magtape Drive Operational Note:

The tape density on a model 3 1600bpi/6250bpi capable magtape drive can be set via software or via the density select switch on the front panel of drive, but not both at the same time. When the density is set via a software call to T$MT, the front panel switch is disabled from having any further effect on the tape density. The front panel switch can only be re-enabled by another call to T$MT that enables the switch (see T$MT changes below).

Under PRIMOS2 (DOS), these drives are initialized to enable the front panel density select switch. Under PRIMOS4, density setting is handled at magtape at ASSIGN time as follows: If no density setting option is given in the magtape assign command line, a call is made by the ASSIGN command to T$MT to enable the front panel switch. If a density setting is given in the magtape assign command line, a call is made to T$MT to set the drive to the desired density and the front panel density switch is therefore disabled. In both cases operator intervention is not needed and is therefore not requested at the system console as required with other magtape drive models. If PRIMOS crashes and a tape dump is to be taken, the act of hitting the "MASTER CLEAR" switch causes these drives to revert to their initial state of 1600 bpi with the front panel switch disabled and the dump is therefore taken at this density.

It is important to remember given that the front panel density select switch is in an enabled or disabled state after giving an ASSIGN command, running a user program which sets the density thru a call to T$MT may change its state as outlined in the beginning of this section.

2.4.2 I$GS

The maximum I/O transfer size has now been increased from 4096 words to 8192 words.

2.4.3 I$MI

The following instructions have been added:

Octal	Hexadecimal	Meaning
100140	8060	Enable front panel density select switch (Version 3 controller only). Set density to 3200 BPI (Version 4 controller only).
100160	8070	Set speed to 25 IPS (Version 4 controller only).
100200	8080	Set speed to 100 IPS (Version 4 controller only).

2.5 BADSPOT HANDLING

The scheme for handling "badspots", or unusable records, on disks has been extensively modified. See Chapter 6 for details.

2.6 COMMAND PROCESSOR EXTENSIONS

The command language and its processor have been extended to recognize several new constructs. These are intended to make it easy to repeatedly execute a command over a set of files or with a specific list of arguments.

A detailed description can be found in Chapter 7.

IMPORTANT: Several punctuation characters, which have been reserved for Prime use on command lines since Revision 14, have now been brought into use.

These are: " ", "+", "=", "%", ".", "," and "~".

Any user programs that employed these characters unquoted in the command line, in violation of the documented restrictions, will cease to work correctly.

2.7 CPL PHANTOMS

The ring 0 gate PHNTM$ allows a process to start up a phantom using either a cominput file or a CPL program; a flag CPLFLG specifies which case holds.

If a pl1g user uses PHNTM$ to start up a non-CPL phantom, then he should specify dummy arguments for ARGS and ARGSL.

The old gate PHANT$ is being retained for compatibility; it may be used only to start up non-CPL phantoms, and its calling sequence is unchanged.

2.7.1 PHNTM$: Start Up Phantom

Usage

```
dcl phntm$ entry (bit(16) aligned, char(32), fixed bin, fixed bin,
                  fixed bin, fixed bin, char(128), fixed bin)

call phntm$ (cplflg, file_name, name_len, unit, phant_user,
             code, args, argsl)
```

cplflg - if true ('1'b), then a CPL program is being started as a phantom; if false ('0'b), then a cominput file is being started as a phantom.

file_name - The name of the file to be started as a phantom.

name_len - The number of characters in file_name.

unit - The unit on which to open the phantom file.

user - The user number of the phantom (output).

code - An error code; zero means no error (output).

args - The arguments for a CPL phantom; a dummy argument must be given for non-CPL phantoms.

argsl - The number of characters in args; a dummy argument must be given for non-CPL phantoms.

2.8 DATE

The DATE command has been changed to output todays date in the same form as the message printed during login or logout. This form is also used by LD to output dates. When it is quoted, this form is accepted by and of the date command line options like -BEFORE, or to any CPL file that has an &args directive that contains the DATE data type.

2.9 DISK_QUOTAS

Primos now provides the ability for the system administrator or users to place limits on the amount of disk storage that is used by any given file system directory.

See Chapter 8 for details.

2.10 EVENT_LOGGING

For information on the new event loggin mechanism, see Chapter 12 of this document.

2.11 FILE_SYSTEM_UTILITY

At Revision 19, we have replaced the old FUTIL subsystem and LISTF command with a set of PRIMOS commands jointly called the File System Utility. Together with the new Command Processor extensions described elsewhere in this document, these commands provide a more powerful and easier to use set of file system functions.

FUTIL and LISTF remain in the system and are unchanged, but they do not support all of the new command environment features.

For details turn to Chapter 9.

2.12 FIX_DISK

FIX_DISK is a replacement for the FIXRAT utility: it checks and corrects damaged file system disks. See Chapter 10 for details.

2.13 ENHANCED_FORCEW_PRIMITIVE

The FORCEW primitive has been enhanced giving the user the option of obtaining the status of disk write operations to a file.

When a disk write error occurs, all units open on the file are specially marked. When FORCEW is called with the error code parameter given, if an error condition exists E$DISK is returned and the error mark is reset. If CODE is not supplied no action is taken, the error mark is not reset and may be sensed at a later time.

NOTE: The error mark is set in all units associated with the file regardless of which one of them caused the actual error.

The new calling sequence of FORCEW is:

```
dcl forcew entry (fixed, fixed, fixed);

call forcew (key, unit, code);
```

key is as before.

unit is as before.

code is a standard error code that is E$DISK when a disk error occurred on the file referenced by UNIT.

If code is not supplied as an argument then disk errors will not be reported.

2.14 THE_HELP_COMMAND

At revision 19 an elementary help facility has been provided as part of Primos. It may be used by invoking the HELP command, which has a very simple syntax:

```
HELP [<subject>]
```

If no <subject> is given, a list of subjects will be printed on the terminal. Otherwise, a top-level directory called HFLP* is searched for a file with the name specified by <subject>. If one is found, it is printed. If one is not found, then a file in HELP* called HELP.SEARCH_LIST is searched for the <subject>. Entries in the search

list point to either a file in HELP* or list of conflicting subjects. Whichever is found will be printed. If the <subject> cannot be found in the search list, an error message is printed.
In order to prevent CRT users from losing information off their screens, the HELP system will stop after every 23 lines and type "--More--". If any of "q", "quit", "n", or "no" (followed by a return) is entered, the HELP system will be exited. Anything else (followed by a return) will cause the next 23 lines of information to be displayed.

2.15 Named Semaphores

Before a named semaphore may be used, it must be opened by either SEM$OP or SEM$OU, which will assign semaphores from a global pool for the user to use. Both of these routines return a set of numbers which can be used instead of numbered semaphore numbers in all other semaphore routine calls. Only valid 'numbered semaphore' numbers and semaphore numbers that have been assigned to a process by either sem$op or sem$ou can be used in subroutine calls that manipulate semaphores. An attempt to use any other numbers will result in an error return from the routine.

2.15.1 Opening Named Semaphores

To open a set of named semaphores, they must be associated with a file system object. SEM$OP will open a set of named semaphores associated with the name of a file in the current ufd of the process performing the open operation. If the process has at least read access rights to the file, it will be assigned the semaphores. Each semaphore will be initialized to zero. SEM$OU will open a set of named semaphores, associating with them a file open on a particular file unit. As before, if the process has at least read access rights to the file, it will be assigned the semaphores. Unlike SEM$OP, the each semaphore within the set may be initialized to the same non-positive value, not less than -32767 decimal.

All calls to either SEM$OP or SEM$OU which indicate the same file system object will result in the same semaphore numbers being returned. The operating system can tell when different processes wish to use the same set of semaphores by examining the parameters that they include in the call to the open routine.

The first process to open a named semaphore may initialize the counter for the semaphore to any non-positive value.

2.15.1.1 Open semaphores

```
call sem$op (fname, namlen, snbr, ids, code)

call sem$ou (funit, snbr, ids, init_val, code)
```

'Fname' (char(32)) is a file name, discussed below.

'Namlen' (fixed bin) is the number of characters in 'fname'.

'funit' (fixed bin) is a file unit number on which a file is already open.

'Snbr' (fixed bin) is a number that specifies how many semaphores are to be opened by this call.

'Ids' ((snbr)fixed bin) is an array of semaphore numbers; one number is returned for each semaphore that was successfully opened.

'init_val' (fixed bin) is the non-positive value with which to initialize each named semaphore which is being opened for the first time. This value is restricted to being non-positive and greater than -32767.

'Code' (fixed bin) is a success/failure code. A value of '0' indicates success. 'E$BPAR' indicates that an invalid value was supplied for 'snbr'. 'E$UNOP' means that a file is no longer open on the specified file unit. 'E$BUNT' indicates that the file unit specified was invalid. 'E$IREM' means that a file that is on a remote disk was specified in the 'Fname' parameter -- remote files can not be used as parameters to this call. 'E$FUIU' means that either the user has all available file units opened, or that there are no available named semaphores. It is also possible that 'code' will be set to any error code that can be returned by the 'SRCH$$' routine.

If access is granted to the semaphores, then the call will return an array of semaphore numbers in the 'ids' parameter. One number will be returned for each semaphore requested in 'snbr', assuming enough semaphores exist in the system pool. A semaphore number of zero will be returned if a semaphore could not be assigned. In addition, 'code' will be non-zero if one or more semaphore numbers could not be assigned. The values returned in 'ids' should be examined to determine which semaphores were opened (non-zero value returned), and which were not (zero value returned).

The semaphore numbers returned should be used in all other semaphore calls as the semaphore number parameter.

If different processes call either of the named semaphore open routines, and specify the same file, the same semaphore numbers will be returned to each process. This allows multiple processes of a subsystem to reference common semaphores.

If a call to the open routine specifies the same file as a previous call to open, and a larger number of semaphores is requested, then new semaphores are acquired from the system pool to make up the difference between the number currently open (with that file name) and the number requested in the call. Other processes can not use these newly assigned semaphores unless they explicitly open them via a call to the open routine.

When the first process opens a named semaphore, the operating system will set the value of the semaphore counter to an initial value. If the named semaphore is being opened via SEM$OP, the counter will be set to zero. Alternatively, SEM$OU will initialize the semaphore counter to the caller supplied value, which is restricted to be non-positive and not less than -32767 decimal. Subsequent opens of the semaphore do not alter the value of the counter.

A named semaphore to be opened may only be associated with a file which resides on the local computer system. Attempts to associate a remote file with a named semaphore during the semaphore open operation will result in failure; no semaphore numbers will be assigned and 'code' will be set to F$IREM.

2.16 CHANGES IN THE STATUS COMMAND

STAT DISK does not display the physical device number for remote disks that are accessed via FAM II.

STAT USERS display all the slaves that are working on behalf of remote users as user type slave, with the name of the user for whom they are executing.

2.17 New user primitive PAR$RV

A new user primitive is being provided to obtain the revision number of a disk partition via the name of the partition.

```
dcl par$rv entry (char (32) var, fixed bin) returns (fixed bin);

par_rev = par$rv (part_name, code);

part_name    32 character varying string containing the pack name

code         error return code
             e$fntf     partition name not found in disk tables
             e$bnam     illegal disk partition name

par_rev      partition revision number
                         0 = pre-acls and quotas
                         1 = converted to allow acls and quotas
                        -1 = error - see error return code for details
```

2.18 New user primitive PRI$RV

A new user primitive is being provided to obtain the revision number of the currently running Primos operating system.

```
dcl pri$rv entry (char(32) var);
```

```
call pri$rv (primos_rev);
```

primos_rev 32 character varying string containing the Primos revision number

2.19 USER_PROFILES

In order to reliably use Access Control Lists, it is necessary to validate a user's login name to be reasonably certain that the user is not an imposter.

At Revision 19, we provide in the standard system a way for system administrators to assign login names and passwords to users. The system will check the legality of the login name and the password at login time.

This new subsystem provides other features as well, such a project administration and an initial set of user "attributes" than can be set by the administrator.

Many sites implement subsets of these features in their own external login programs. We believe most will find it advantageous to begin using this Prime-supplied subsystem.

For full details, see Chapter 12, "USER PROFILES".

3 NEW ERROR CODES

The following paragraphs describe the new standard PRIMOS error codes, broken
down by the part of PRIMOS which returns them (if possible):

3.1 General error codes

3.1.1 E$BVER

Bad version occurs whenever a bad version argument is passed to a gate which requires a structure with a user-supplied version number.

3.1.2 E$DNSK

A PIO Instruction Did Not Skip occurs whenever the direct execution of an
io instruction is requested and the io instruction didn't skip.

3.1.3 E$DTNS

Date and time not set occurs whenever a command is issued that must be preceded by the system's date and time being set.

3.1.4 E$IEDI

I/O error or device interrupt occurs whenever an i/o error or a device interrupt is detected.

3.1.5 E$IMFD

Operation illegal on MFD occurs whenever a user tries to execute some file system function which is illegal for MFDs, e.g. setting a quota with
SET_QUOTA or reverting to default protection with SET_ACCESS.

3.1.6 E$MISA

Missing_argument_to_command occurs whenever a command is not passed enough arguments.

3.1.7 E$NFAS

Top-level_directory_not_found_or_inaccessible is returned when an attach scan takes place and any error other than E$NPIT (Insufficient access rights) or E$BPAS (Bad password) occurs. See the discussion of changes to ATTACH, below.

3.1.8 E$NINF

No_Information occurs when an attempt is made to access an object in a directory on which the user does not have List access, but the user cannot access the object for some (unspecified) reason.

3.1.9 E$SCCM

System_console_command_only occurs whenever a command that may only be issued from the system console is entered anywhere else.

3.1.10 E$ST19

Disk_format_does_not_support_this_revision_of_PRIMOS is returned when an operation such as SET_ACCESS is attempted on a pre-rev 19 disk or system.

3.1.11 E$WMST

Warm_start_occurred occurs whenever a PRIMOS warmstart has occured.

3.2 Disk Quota error codes

3.2.1 E$MXQB

Maximum quota exceeded occurs whenever a user tries to use more disk space than has been allowed for the user.

3.2.2 E$NFQB

No free quota blocks occurs whenever an attach operation finds that there is not another free quota block entry available.

3.2.3 E$NOQD

Not a quota disk occurs when an attempt is made to set a quota on a disk that has not been formatted for disk quotas, or on a pre-rev 19 system.
If the disk is running on a rev 19 system, FIX_DISK should be run to format the disk so that quotas may be set.

3.2.4 E$QEXC

Quota set below current usage occurs whenever a user's quota limit is reduced to below the current amount of space used. This error is only a warning: the quota is set anyway.

3.3 ACL error codes

3.3.1 E$ACBG

ACL too big occurs when an AC$SET or AC$CHG call would cause the ACL being operated on to have more than 32 entries or be more than 255 words long.

3.3.2 E$ACNF

Access category not found is returned by AC$CAT when the category cannot be found.

3.3.3 E$ADRF

Directory still contains ACL subdirectories occurs whenever an AC$RVT call
is made on a directory which contains one or more ACL-protected subdirectories.

3.3.4 E$BACL

Incorrect access control list format is returned by AC$SET or AC$CHG when one of the access pairs is formatted incorrectly, e.g. no colon separates the <id> from the <access>.

3.3.5 E$BID

Illegal identifier is returned by AC$SET and AC$CHG when an
identifier in one of the passed access pairs is illegal.

3.3.6 E$BMOD

Illegal access mode occurs whenever a user specifies an ACL access mnemonic which does not exist.

3.3.7 E$CATF

Directory still contains access categories occurs whenever an AC$RVT call
is made on a directory which contains one or more access categories.

3.3.8 E$CPMF

Category protects MFD results when a CAT$DL call is attempted on the access category protecting the MFD.

3.3.9 E$CTPR

Object is category-protected is returned by AC$CHG when it is called on an object which is currently protected by an access category.

3.3.10 E$DFPR

Object is default-protected is returned by AC$CHG when it is called on an object which is currently protected by default access.

3.3.11 E$DLPR

File is delete-protected occurs when an attempt is made to delete a file whose delete-protect switch is set.

3.3.12 E$IACL

Entry is an access category occurs whenever a file operation is attempted on an access category.

3.3.13 E$LRNA

Cannot access like reference results when the reference in an AC$LIK call cannot be read for any reason.

3.3.14 E$LRNF

Like reference not found is returned by AC$LIK when the reference object cannot be found.

3.3.15 E$NACL

Not an ACL directory occurs whenever a user tries to perform an ACL operation on a password directory. Compare E$PNAC, below.

3.3.16 E$NCAT

Not an access category occurs when an object other than an access category is used as the reference in the -CATEGORY option of SET_ACCESS (or in an AC$CAT call).

3.3.17 E$NTFD

Not a file or directory results from attempting to place an access category
into an access category with AC$CAT.

3.3.18 E$PANF

Priority ACL not found is returned by PA$LST and PA$DEL when the
partition specified is not currently protected by a Priority ACL.

3.3.19 E$PNAC

Parent not an ACL directory occurs whenever an attempt is made to perform
an ACL operation within a password directory. Compare E$NACL, above.

3.4 User Profiles error codes

3.4.1 E$LOGO

Is a special error code which is used by FATAL$ in the logout sequence. It
is never returned to users.

3.4.2 E$NFUT

No unit table available occurs whenever an attempt is made to allocate a unit
table and there are no more unit tables left in the free pool.

3.4.3 E$NUTP

No unit table for phantom occurs whenever a phantom is trying to login and
no unit table can be acquired.

3.4.4 E$NVAL

Validation Error occurs whenever an NPX slave fails in its attempt to log into an untrusted node.

3.4.5 E$UAHU

User already has unit table occurs whenever a try is made to allocate a unit table that a user has already been accuired.

3.4.6 E$UNIU

Unit table not in use occurs whenever an attempt is made to return a unit table to the free pool when the unit table is not in use.

3.4.7 E$UTAR

Unit table already returned occurs whenever a try is made to deallocate a unit table that has already been deallocated.

4 CORRECTED REVISION 18 POLEPS

4.1 MAGTAPE

If buffer cross segment boundry, unpredictable results including crashes and hangs may occur.

4.2 MAGTAPE

The current limit for magtape reads and writes is six pages. If a user attempted to transfer more than 32K words with a buffer in the low part of a segment, the error was not detected and the I/O transfer would crash the system.

4.3 MAGTAPE

When a user assigns a magtape with an ASSIGN command with options that require operator help, the user is set to waiting for a reply command from the system console that says that option has been carried out. When the ASSIGN command originated from the console, the console was hung wait for a reply from itself! This has been fixed so that if the ASSIGN comes from the console, no wait takes place. Note that the 'ASSIGN MTX -ALIAS MTn ...' form of the ASSIGN command will produce an error because it requires a reply which cannot be sent in this situtation.

4.4 MAGTAPE (POLEP # 29261, 31851, 35688)

It was impossible to set the density to 6250 bpi on model 3 controller/magtape drive from the front panel. This has been fixed - see the note in the under the ASSIGN command and under the usage note for this magtape drive, both in the enhancements section.

4.5 UNASSIGN (POLER # 33448)

An error in using the magtape UNASSICN command gave an error that demonstrated the proper use of the magtape ASSIGN command.

4.6 FORCEW (POLER #10520)

Make sure a force write to an NP/NW file happens properly.

4.7 AINIT (POLER #23089)

Make error message print if Ainit can't attach to CMDNC0 because of bad password.

4.8 CINIT (POLER #82632)

Fixed default value for second argement in CINIT.FTN which was incorrectly
calculated.

4.9 USER_1 (POLER #29087)

Check user 1's msg buffer rather than the output buffer for consistency.

4.10 SLABRT (POLER 29196)

Fix bug in slabrt.ftn when unassiging a line >3.

4.11 PMSG$ (POLER #32411)

Corrected bug in PMSG$ which caused system halt if called with a bad argument type.

4.12 COMO$$ (POLER 34341)

Correct incorrect comment in como$$.ftn regarding :200 key.

4.13 COMO_FILES

Comoutput files may now be closed either by the "COMO -E" command, or the
OPEN or CLOSE commands.

4.14 ATTACH

TA$.FTN now attempts to attach to an MFD when the target attach point is
'<'device name'>'filename. User commands effected are 'LISTING' and 'BINARY'.

4.15 PRINTRONIX PRINTER (POLERS # 20655,32622,31068)

When the Printronix printer is nearly done printing the last file in the
spool queue, it stops before completely printing out the file. The remaining lines are eventually printed but at the rate of about one line per minute.

4.16 CPL (POLER #41506)

The following additions have been made to CPL for Rev. 19.0:

1. It is no longer required that the &tty directive be the last directive in a &data block. The directive is always invoked when the commands in the &data temporary file are exhausted, as before. This feature allows the use of a conditional statement with &tty, such as:

   ```
   &if blat &then &tty
    &else stuff
   ```

2. When echoing is enabled, comments and null lines will NOT be echoed. Only null lines will be inserted into &data temp files.

3. The defgv command now accepts an "-off" option, which turns off the current global variable file.

4. The wild function now accepts the singular forms of all the options (e.g. -file, -segdir, -dir). This makes it work just like the other file system command functions. The plural forms are obsolete.

5. There is a new command function "gvpath" which returns the pathname of the current global variable file, if one is defined, or "-off" if no file is active. It is invoked by [gvpath].

6. The get_var function now returns $UNDEFINED$ if the value of a global variable is requested and no global variable file is active.

7. The &args directive now explicitly rejects numeric option arguments, i.e., those of the form "-123".

4.17 CPL (POLER 41507)

A CPL nonlocal goto from a CPL on-unit to a "start <address>" command misthreaded the stack, causing a fatal$ error. The stack is now correctly threaded.

4.18 T$CMPC (POLER #27073)

Receipt of any status codes from T$CMPC with a read status instuction (:100000) is unreliable. This routine has been modified to add a new instruction to T$CMPC to return the actual hardware status (:100001). The status returned by the existing read status instruction will continue to reflect only the status of the input buffer in Primos.

4.19 T$CMPC (POLER #27971)

Reading cards in binary takes twice as long as reading then in ASCII.

4.20 T$LMPC (POLER #33480)

The read status instruction of T$LMPC returns a status indicating the line printer is online regardless of its state. This routine now returns the hardware online status bit from the controller.

4.21 PHANTOMS (POLER # 41521 41622)

1. PHNTM$ checks the existence of files using k$exst key. This causes an error to be reported if the file is inaccessible.
2. Phantoms which explode while logging in will now clean up (close files, etc.) before dying.

4.22 WTLIN$ DISK FULL CONDITION (POLER #45589)

Code is checked afer call to PRWF$$ to check for a disk full condition. If condition exists WTLIN$ exits file.

5 ACCESS CONTROL LISTS

5.1 Introduction

5.1.1 What are Access Control Lists?

Access Control Lists (ACLs) are a way of protecting file system objects (files and directories) from unauthorized access. They provide a passive (requiring no intervention by the accessing user) mechanism for effecting this protection, as opposed to the active mechanism currently provided by passwords. ACLs are simply lists of ordered pairs (<identifier>, <access rights>) which determine what users and groups of users are accorded rights to files and directories.

5.1.2 Why ACLs?

The current password protection mechanism is cumbersome to use (because passwords must be typed as part of the pathname), inflexible (because it is not hierarchical and provides only very coarse access control), and not really very secure (since passwords must be hidden in programs). ACLs provide an easy to use, flexible, and secure method of protecting file system objects.

5.1.3 Problems

This section describes some of the problems which the ACL system is attempting to resolve.

5.1.3.1 Flexibility

Password protection is inflexible for a variety of reasons. First, it is not hierarchical; that is, access on the root of a subtree may not be inherited by the branches. Secondly, passwords may be applied only to directories; files may not be protected for access only by a specific group of users (those who know the password) without placing them into a separate directory. Thirdly, the group of users granted access to a directory is only controllable by word-of-mouth: the group is defined by to whom the password is given. This makes changing the group extremely difficult.

5.1.3.2 Ease of use

Passwords are difficult to use because they must be embedded in pathnames and remembered by both users and programs. The fact that they are separated from the rest of the pathname by a space further complicates the matter because they must be quoted in order to form a single token for the command processor.

5.1.3.3 Security

Because eliminating a user from the group of those granted access requires changing the password and informing all other users of the change, passwords often remain unchanged for long periods of time, resulting in breaches of security. Furthermore, there is no way of allowing a user to use files in a directory without being able to examine the contents of the entire directory.

5.1.4 Solutions

The new ACL system addresses these problems in the following ways:

5.1.4.1 Flexibility

ACLs have the ability to protect objects singly or collectively. The provide a passive means of protection, both because there is nothing extra to be typed or remembered, and because default protection can be made very restrictive. Adding and removing single users or groups of users from ACLs is very easy, and thus changes in access may be made regularly and without difficulty by a single individual.

5.1.4.2 Ease of use

The set of commands required to use the ACL system has intentionally been kept very small. In all respects it attempts to do "DWIM" -- "Do What I Mean" -- as much as possible, in order to make the user's life both easier and safer. Only three commands are needed to fully utilize the ACL system: SET_ACCESS to protect an object or create an access category, EDIT_ACCESS to change the access in an existing ACL, and LIST_ACCESS to examine the ACL for any object; furthermore, users could get along quite well without ever using EDIT_ACCESS.

5.1.4.3 Security

The ACL system, when coupled with user profiles, provides a measure of security that the password system could never hope to approach. Access control may be as fine or coarse as necessary. Users may be granted a variety of rights on an individual or group basis. Certain users may easily be excepted from a group without affecting the rest of the group.

5.1.4.4 Default protection

In the ACL system, default protection is strictly hierarchical: the default protection on a subtree is provided by the ACL which protects the root of the tree.

5.1.4.5 Directory clutter

The problems of clutter and extra work brought about by requiring all ACLs to have names of their own has been solved by providing a "file-centric" view of ACLs. File system objects may be protected by a specific ACL which has no name of its own and is consequently referenced only through the object itself. If a group of objects requires common protection, named ACLs called access categories may be created, and the objects "added" to the category. Default protection, of course, requires no additional ACLs.

5.1.4.6 Ability to change access on one's own directory

In the ACL system, an ACL may be edited by either the person having Protect access on its parent directory, or any person who has Protect access in the ACL itself. This means that a user may change the ACL protecting his top-level directory himself (assuming, of course, that he has Protect access on it), and also that a user may edit an access category in which she has been granted Protect rights.

5.2 Definitions

In order to assist the reader, the following definitions are provided.

5.2.1 Access Category

A named ACL (see below) which is used to protect a number of file system objects in a common way. Access category names always end with the suffix ".ACAT".

5.2.2 Access Control List (ACL)

A list of ordered pairs (<identifier>:<access>) which, when combined, define the collection of users who either have or are denied access to some file system object.

5.2.3 ACL system

The software in PRIMOS which provides support for use of the Access Control Lists mechanism of protecting file system objects.

5.2.4 Access Rights

A list of mnemonic privileges which define the manner in which some user or group of users is allowed to use a file system object.

5.2.5 Default Protection

Security provided for file system objects without any intervention by the user.

5.2.6 Directory

Either a UFD or MFD; not a segment directory.

5.2.7 Explicit Protection

Non-default protection; that is, protection by specific ACL or access category (which see).

5.2.8 File

Specifically, a file system object which contains user-accessible data (as opposed to a directory, which basically does not). For purposes of access control, "file" also refers to segment directories. In general, the term "file" may be used to designate any file system object.

5.2.9 File System

The software in PRIMOS which provides support for the file and directory hierarchy.

5.2.10 File System Object

Anything that exists in the File System and has a name.

5.2.11 Identifier

A name which identifies a user (user id) or group of users (group id).

5.2.12 Priority ACL

An ACL which is specified for an entire disk partition in order to allow overriding of the access control on the disk. Priority ACLs exist so that special functions such as system backup may be performed. Priority ACLs may be set only by user 1 or the system administrator.

5.2.13 Specific Protection

Protection of a file system object provided by an ACL which is directly associated with the object itself, almost as an attribute of the object. Specific ACLs may be referenced only through the object which they protect.

5.3 Using ACLs

This section describes the use of the ACL system by first describing in detail the components of an ACL, and then going on to discuss the commands used and their interaction with the file system.

5.3.1 The contents of an ACL

As mentioned above, an ACL is simply a set of ordered pairs of identifiers and access rights. The identifier is separated from the access rights by a colon (":").

5.3.1.1 Identifiers

Identifiers in ACL pairs identify either a single user, a group of users, or all users who do not fall into the above two categories. Individual users are identified by their user id (for a description of user id's, see the section of this document describing User Profiles). Groups of users are identified by group names, which always begin with a dot ("."). Groups are assigned by system and project administrators and set up at login time; they are also discussed fully in the section on Profiles. Any user not listed either by name or in a group may be covered by the special identifier "$REST," which is essentially a "catch-all" group.

5.3.1.2 The ACL access rights

ACLs provide access control by associating identifiers with lists of access rights. The rights available and their meanings are as follows:

Right	Applies to	Primary Meaning
Protect	Directories	Accesses and attributes may be changed.
Delete	Directories	Entries may be deleted from the dir.
Add	Directories	Entries may be added to the dir.
List	Directories	The contents of the dir may be read.
Use	Directories	The dir may be attached to.
Read	Files	The contents of the file may be read.
Write	Files	The file contents may be changed.
ALL	Both	Grant all access rights.
NONE	Both	Explicitly deny all access.

5.3.1.3 What rights do I need?

The variety of access rights supported by the ACL system was chosen to allow greater flexibility of controlling what users can and cannot do to a given file or directory. A more detailed description of what each access provides is given here.

5.3.1.3.1 Protect

Protect is the most powerful of all access rights. If a user has Protect access on a directory, he may change the ACL protecting it to allow him any other rights. Since List access is required to read the directory in the first place, and Use access to attach to it, Protect should never be given without List and Use. In addition to allowing a user to set access on files and directories, Protect access controls setting of file attributes with SATR$$ and reading of passwords.

5.3.1.3.2 Delete

In the ACL system, delete is a per-directory (rather than a per-file) right. If Delete access is available on a directory, any file or directory immediately contained in that directory may be deleted. Because of the way in which PRIMOS deletes directories, however, Protect rights may be required in order to delete certain subtrees. Objects may have a "delete protect switch" set on an individual basis to avoid "inadvertent" deletion. This switch is set by the SET_DELETE command and SATR$$ subroutine (see below). Also, in addition to controlling deletion of files, Delete and Add access (see below) together control the right to rename files.

5.3.1.3.3 Add

Objects can only be created in a directory on which the user has Add access. Since newly created files are opened with all file accesses available, granting Add access without Write access (see below) allows users to create new files (e.g. providing a copy of a bug-driving program) but not to change them after they exist. Delete and Add accesses are required to rename files.

5.3.1.3.4 List

A directory may be opened for reading (e.g. for RDEN$$) if List access is available. List access does not imply Use access (see below); Use access must be explicitly granted in order to allow a user to attach to a directory.

5.3.1.3.5 Use

A directory may be attached to if Use access is available. The directory may not be read with RDEN$$, however, and if a file cannot be found for any reason, "No information" (E$NINF) is returned. Note that Use access is required on all directories--including MFDs--in order to be able to attach to them, and consequently to search them for entries.

5.3.1.3.6 Read

File contents may be examined if Read access is available.

5.3.1.3.7 Write

The contents of a file may be changed if Write access is available. Write access includes the ability to truncate the file.

5.3.1.3.8 Mapping of password rights into ACL accesses

Owner/non-owner rights in password directories are internally mapped into ACL accesses by the file system. This allows PRIMOS modules to treat all files as files, without worrying about whether they are in a password or ACL directory. Externally, users will still see OWNER/NON-OWNER for directories and owner/non-owner protection pairs for files. The mapping is done as follows:

```
Owner:      PDALU
Non-owner:  LU
Read:       R
Write:      W
```

Delete permission is granted to those with owner rights only to allow CNAME to work in password directories (since it requires Delete and Add). It is not checked when a file within a password directory is being deleted; instead, the per-file Delete rights in the PROTEC word are used. This is the only case in which file system code must check to see if the object being operated upon is in a password directory.

5.3.1.4 Implicit $REST Identifier

What happens when the list of identifiers in an ACL is incomplete? That is, what if neither the user nor any of his groups is explicitly listed in the ACL and there is no $REST identifier? In these cases, the system supplies a default "$REST:NONE" access pair, so that any users not explicitly named get no access.

5.3.2 Protecting File System Objects

This section describes how the commands of the new ACL system are used to protect objects in the file system. It is designed to be as tutorial as possible, and not to be a detailed reference. A detailed commands reference guide is provided in the next section.

5.3.2.1 Conversion to ACLs

Conversion from a password to an ACL directory is done automatically whenever the SET_ACCESS command is given on a password directory whose parent is an ACL directory. (The MFD may be converted even if its "parent" is not an ACL directory, of course.) This conversion can only take place after the disk has been brought up to PRIMOS revision 19 with FIX_DISK (see below), however. Note that since in the new ACL system Protect permission may be gotten from either the parent or the object itself, a user needs Owner (or Protect) rights on the parent or the target directory in order to convert a password directory with the SET_ACCESS command. This means that once the MFD is converted to an ACL directory, any top-level password directories with null owner passwords may be converted by anyone, unless List rights are not granted to the general user community by the MFD default ACL.

5.3.2.2 Setting protection on the MFD

If the MFD is an ACL directory, it must always be protected by either a specific ACL or an access category (since it has no "parent" from which to inherit default protection.) Normally, the MFD is protected by a specific ACL, since no categories will exist in a non-ACL directory. The MFD is the only directory which may be converted to an ACL directory without having a parent which is an ACL directory, for obvious reasons. Attempts to revert the MFD to default protection or to delete an access category which protects the MFD will be rejected.

5.3.2.3 Setting access on File System Objects

File system objects are protected with the SET_ACCESS command. Depending on how the command is used, it may set default, specific, or category protection, as follows:

```
SET_ACCESS <object>
```

reverts the <object> named to default protection.

```
SET_ACCESS <object> <ACL>
```

creates a specific ACL containing the <ACL> specified and protects the <object> with it.

```
SET_ACCESS <object> -CATEGORY <category_name>
```

places the <object> into the named access category.

Full details on the SET_ACCESS command are provided in the next section.

5.3.2.4 Using access categories

Most users will have a combination of access control needs. In some cases protecting all files in a like manner will be sufficient. In others, only one or two files need be protected in a unique manner, and specific protection suffices. There are times, however, when a certain group of files should be protected in a different way than any others, and when moving those files into a separate directory may be impractical or undesireable. It is for these latter times that access categories are provided. Categories are created by using SET_ACCESS and providing an <ACL> (the second case above). If the <object> for SET_ACCESS does not exist, an access category is created. Since categories have "lives of their own," they may be created before any objects are "put into" them, and furthermore may exist after all objects have been "removed from" the category. In general, access categories simply provide a straightforward and efficient way to group files together for access control purposes.

5.3.2.5 Priority ACLs

Sometimes the system administrator or operations staff require special access to the file system. In these cases, Priority ACLs may be used to supersede access rights granted in the file system itself. For instance, when doing system backups, the operator should be able to read any file on the disk so it can be backed up. Priority ACLs are set with the SET_PRIORITY_ACCESS command:

```
SET_PRIORITY_ACCESS <partition_name> <ACL>
```

Priority ACLs differ from "physical" ACLs in that they do <u>not</u> supply an implicit "$REST:NONE" access pair. If the user is not found in the priority ACL, his access is computed as if the priority ACL did not exist.

5.3.3 How access is calculated

When a user attempts to open a file or attach to a directory, the file system must calculate the user's access to the file or directory. Let us first describe what is in an ACL, then go on to how that information is used in access calculation.

5.3.3.1 The parts of an ACL

Each ACL has three distinct parts which are searched independently for a match on the user's id or groups. The three parts are:

5.3.3.1.1 The User ID section

All single users who are included in the ACL are in this section. If the accessing user's ID is found in this section, his access is determined by his entry here.

5.3.3.1.2 The Group ID section

If the user's user ID was not found in the user ID section, the group ID section is searched. The rights for all groups to which the user belongs that are found in this section are logically ORed together to determine the user's rights.

5.3.3.1.3 Leftovers ($REST section)

If the user was found in neither the user nor the group ID sections, he is granted the access specified by the entry for $REST. If no entry for $REST exists, the user is denied access to the object. (With the exception that Priority ACLs do not have a default $REST entry.)

5.3.3.2 Where access comes from

When an attempt is made to access a file system object, the following sequence of events occurs:

5.3.3.2.1 Check for a Priority ACL

If the partition on which the object is being accessed is protected by a Priority ACL, that ACL is searched for a match in the sections described above. If a match is found (recall that a Priority ACL need not have a $REST specifier), the access granted is that which was found in the Priority ACL.

5.3.3.2.2 Check for password protection

If the object is password-protected, access is calculated in the old manner, with the rights mapped as described above.

5.3.3.2.3 Check for default protection

If the object is default-protected, the access is simply taken from the default rights kept in the unit table entry of the current attach point.

5.3.3.2.4 Get rights from an explicit ACL

At this point the object is protected either specifically or by an access category. The appropriate ACL is read and searched as described above.

5.3.4 Common combinations of access rights

The flexibility of the ACL system is found in its various access rights. Combining these rights effectively is an important part of using the system effectively. Some of the more common and useful combinations of access rights are listed here.

5.3.4.1 Rights for the "owner" of a directory

Normally "owners" of directories will be granted all rights (PDALURW, indicated by the special mnemonic "ALL"). If the "owner" of the directory is really only supposed to use it, however, sometimes Protect permission will be left off, resulting in DALU rights (for directories). This allows the user of the directory to do anything but change protection and attributes of objects in the directory and its subtree.

5.3.4.2 Rights for other users in a group or project

When working with other people on a project, it is often useful to allow them to access each other's files but not to change or destroy them. In such cases, the combination LUR is used most often. It allows users granted these rights to examine files and directories and execute programs, but does not allow them to change anything.

5.3.4.3 Rights for outsiders

True outsiders are often denied rights altogether with the NONE special mnemonic. Sometimes, however, it is desired that certain files be accessible to anyone. In this case, granting UR access allows files whose names are known in advance to be read, but does not allow anything to be modified, and furthermore does not allow the directory to be perused for additional entries.

5.3.4.4 Passing information around

Sometimes it is useful to allow users to "pass" files back and forth, but at the same time the target user does not want his other files to be damaged. Certainly he could create a subdirectory containing only passed files, but this is cumbersome and inefficient. Instead, the user can grant ALUR rights, which allows new files to be created and written to (since new files are always opened with all rights), but does not allow existing files to be modified.

5.4 User Commands Specification

This section provides a formal specification of all user commands available under the new ACL system. It is designed as a detailed reference document which can be relied upon as the final arbiter of any disputes as to what is "correct" functioning of any of the ACL commands. Abbreviations are indicated by capitalizing the letters of the abbreviation in the examples.

5.4.1 Setting access

The SET_ACCESS (SAC) command is used to specify the complete set of access rights for either a category or a specific object. Its syntax has the following forms:

```
Set_ACcess <target>
Set_ACcess <target> <access_control_list> [-No_Query]
Set_ACcess <target> -LIKE <reference> [-No_Query]
Set_ACcess <target> -CATegory <category_name>
```

For purposes of this discussion, the term "file" means "file, directory, or segment directory." In all cases, if the <target> is a password directory whose parent is an ACL directory, the <target> is converted to an ACL directory. The <access_control_list> may theoretically contain up to 32 pairs (the maximum allowed in any one ACL), but may not in total be more than 160 characters long, including blanks.

5.4.1.1 When only a <target> is given

The <target>, which must be a file, is set to use the default access for the directory. This form may not be used if the <target> is an MFD.

5.4.1.2 When an <access_control_list> is given

The action in this case depends on whether or not the <target> exists, what its type is, and how it is currently protected.

5.4.1.2.1 <target> is a file

The ACL for the file is set as specified. If no specific ACL currently exists for the file, one is created. If a specific ACL does exist, the user will be queried before its contents are replaced (this query may be supressed with the -NO_QUERY option). Note that if the file was category-protected the category will not be changed.

5.4.1.2.2 <target> is an access category

The user is queried to determine whether or not he really wants to replace the contents of the category. If an affirmative response (or the -NO_QUERY option) is given, the category's ACL is replaced in its entirety by the specified <access_control_list>.

5.4.1.2.3 <target> does not exist

The user is queried, and if an affirmative response (or the -NO_QUERY option) is given, a new access category is created with the specified <access_control_list>. If the specified category name does not include the ".ACAT" suffix, it is added. Category names which are more than 27 characters long without the suffix with be rejected with the "Category name too long" message.

5.4.1.3 When the -LIKE option is given

In this case, both the <target> and <reference> must be existing files or access categories. If the <target> is an access category, its ACL is replaced by one which is identical to the ACL protecting the <reference>. If the <target> is a file, it is given a specific ACL identical to the ACL protecting the <reference>, regardless of how the <reference> is protected. If the <target> was category-protected it is removed from the category and the category remains unchanged. If the <target> was specific-protected, the old specific ACL is lost.

Note that the protection of any object can be changed from default or category to specific by using the SET_ACCESS command with the -LIKE option on the object itself.

5.4.1.4 When the -CATEGORY option is given

The <target> must be a file, and the <category_name> must specify an existing access category. The <target> is added to the access category. If the <target> was category-protected, it is removed from the old category, but the old category remains unchanged. If the <target> was specific-protected, the old specific ACL is lost.

5.4.1.5 Examples

Set a specific ACL on file "FOO":

```
SAC foo glennw:rwx .opsys:rx $rest:none
```

Put "FOO" into access category "BAR.ACAT":

```
SAC foo -CAT bar
```
or
```
SAC foo -CAT bar.acat
```

Revert "FOO" to default access:

```
SET_ACCESS foo
```

Protect "FOO" like "FOOEY" is protected:

```
SAC foo -LIKE fooey
```

Create a new access category called "NEW_CAT.ACAT":

```
SET_ACCESS new_cat user1:all user2:alur .opsys:ur -NQ
```

5.4.2 Examining access

The LIST_ACCESS (LAC) command allows users to examine the access rights for any file system object. Its syntax is:

```
List_ACcess [<object>]
```

If the <object> is omitted, the access rights for the current directory are given. If the <object> is an access category, its ACL is displayed. Otherwise, the ACL protecting the <object> is listed. Any priority ACL is listed after the normal ACL.

5.4.3 Changing access

The EDIT_ACCESS (EDAC) command is used to modify existing ACLs. Its syntax is:

```
EDit_ACcess <target> <access_control_list> [-No_Query]
```

The <target>, which may be either a specifically-protected file or an access category, has its ACL modified to include each of the new <id>s. If an <id> already exists, its <access> is changed. A null <access> indicates that the <id> should be removed from the list.

EDIT_ACCESS should not be used on files which are currently default or category-protected, but if it is the user will be queried to determine whether or not she wishes to create a new specific ACL. This query may be supressed with the -NO_QUERY option.

If EDIT_ACCESS is used in this latter way, the new specific ACL will be formed by modifying the protection provided by the old category or default protection with the specified <access_control_list> and placing the result into the new specific ACL. The default or category ACL is not changed. For example:

```
OK, LIST_ACCESS foo
"FOO" protected by default ACL (from "<MYDISK>MYDIR"):
        USER1:   ALL
        USER2:   ALUR
        .OPSYS:  UR
        $REST:   NONE

OK, EDIT_ACCESS foo user2:
"FOO" is default-protected.  Create specific ACL? YES
OK, LIST_ACCESS foo
ACL protecting "FOO":
        USER1:   ALL
        .OPSYS:  UR
        $REST:   NONE
```

5.4.4 Setting Priority Access

Priority access may be set with the SET_PRIORITY_ACCESS (SPAC) command, the syntax of which is given below:

```
Set_Priority_ACcess <partition_name> <access_control_list>
```

This command may be used only by the system administrator or from the system console. Priority access may be converted into the ability to override any access control on the disk by simply including a $REST specifier in the priority access list.

5.4.5 Listing Priority Access

When a priority ACL is in effect for a partition, its contents are always displayed in a LIST_ACCESS command. However, since it is possible to prevent users from accessing even the MFD with a priority ACL, the LIST_PRIORITY_ACCESS (LPAC) command allows users to examine the priority ACL on any partition.

```
List_Priority_ACcess <partition_name>
```

5.4.6 Removing Priority Access

The REMOVE_PRIORITY_ACCESS (RPAC) command is used to remove a priority ACL when it is no longer needed. This command may be used by any user.

```
Remove_Priority_ACcess <partition_name>
```

This command may be given only by the system administrator or from the system console.

5.4.7 Setting the Delete Switch

The delete switch for files (to prevent inadvertent deletion) may be set with the SET_DELETE command. Its syntax is:

```
SET_DELETE <pathname> [{-PROtect | -No_PROtect}] [-RePorT]
```

<pathname> is the name of the object on which the delete protection is to be modified.

If the -PROTECT option is given, or neither -PROTECT nor -NO_PROTECT is specified, the delete switch is set (deletion is prevented).

If the -NO_PROTECT option is given, the delete switch is cleared (deletion is allowed).

The -REPORT options requests that the results of executing the command be reported to the user.

The user must have delete access (D) to the directory that contains the object in order to set the delete protect switch on a file, directory, or segment directory. (The switch cannot be used on access categories.)

5.4.8 Reverting to a password directory

If for some reason a user decides that he no longer wants the advantages of an ACL directory, he may convert a directory back to password protection with the REVERT_PASSWORD external command. REVERT_PASSWORD operates only on the current directory, which may contain no access categories or ACL subdirectories. It should be used as sparingly as possible, if at all.

5.4.9 Changes to the CREATE command

The existing CREATE command works as it did before, but by default will create a directory of the type of its parent. That is, if used in an ACL directory CREATE will create an ACL directory; in a password directory it will create a password directory. A new password directory may be created in any type of directory by using the new -PASSWORD option. Thus the syntax of the CREATE command is now:

```
CReate <directory_name> [-PassWord]
```

5.5 Program interface

Anything which can be done with a command from the terminal should be achievable from a program. This section describes the program interface through which both PRIME-supplied and user-written subsystems may access the ACL system.

5.5.1 AC$CAT -- Add a file to a category

Files may be added to an access category with the AC$CAT call. Its calling sequence is:

```
dcl Ac$cat entry (char (128) var, char (32) var, fixed bin);
call Ac$cat (object_path, category_name, code);

object_path:   name of the object to be protected (input).
category_name: name of the category to which the object
               should be added (input).
code:          standard error code (output).
```

The object must exist and must be a file. The category must exist in the same directory as the object and must be an access category. If the object is a password directory and its parent is an ACL directory, the object will be converted to an ACL directory.

5.5.2 AC$CHG -- Modify existing ACL

Existing ACLs may be modified with the AC$CHG call. Its calling sequence is:

```
dcl Ac$chg entry (char (128) var, ptr, fixed bin);
call Ac$chg (name, acl_ptr, code);

name:    pathname of the object whose ACL is to be modified
         (input).
acl_ptr: pointer to the ACL structure (input).
code:    standard error code (output).
```

AC$CHG is similar to AC$SET, but rather than replacing the entire contents of the old ACL, AC$CHG updates the existing ACL with the new data. The object to be changed must be an existing access category or a specifically-protected file. Attempts to use AC$CHG on default-protected files will be rejected with error code E$DFPR (Object is default-protected), and attempts to use it on category-protected files will be rejected with status E$CTPR (Object is category-protected). As in the ACL commands, a null <access> half of the <access_pair> results in the <id> being removed from the ACL. Otherwise, if the <id> already exists in the acl its <access> list is simply changed, and if it does not exist it is added. For details on the ACL structure, see AC$LST, below.

5.5.3 AC$DFT -- Set default protection

An object may be set to use default protection with the AC$DFT gate. Its calling sequence is:

```
dcl Ac$dft entry (char (128) var, fixed bin);
call Ac$dft (name, code);

name: name of the object whose protection is to change (input).
code: standard error code (output).
```

The object must exist and be a file. If it is a password directory and its parent is an ACL directory, it will be converted to an ACL directory. Attempts to use Ac$dft on MFDs will be rejected with error code E$IMFD (Operation illegal on MFD).

5.5.4 AC$LIK -- Protect one file like another one

ACLs may be copied from one file to another with the AC$LIK routine. Its calling sequence is:

```
dcl Ac$lik entry (char (128) var, char (128) var, fixed bin);
call Ac$lik (target_path, reference_path, code);
```

```
target_path:    name of object to be protected (input).
reference_path: name of object from which to take ACL (input).
code:           standard error code (output).
```

Both target and reference objects must be existing files or access categories. A new specific ACL will be created with the ACL of the reference, regardless of how the target and reference are currently protected. The target and reference may be the same, in which case the effect is to force the target to be protected by a specific ACL. If the target is a password directory and its parent is an ACL directory, the target will be converted to an ACL directory.

5.5.5 AC$LST -- Read an ACL

ACLs are read using AC$LST. Its calling sequence is:

```
dcl Ac$lst entry (char (128) var, ptr, fixed bin, char (128) var,
            fixed bin, fixed bin);
call Ac$lst (name, acl_ptr, max_entries, acl_name,
             acl_type, code);

name:        pathname of the object for which information
             is desired (input).
acl_ptr:     pointer to return structure (input, points to
             output).
max_entries: most entries user's buffer can handle. The
             maximum number of entries in an ACL is 32 (input).
acl_name:    name of the ACL protecting the object (output).
acl_type:    type of the ACL protecting the object (output).
             Possible values are:
             spec_aclt (0) -- specific ACL.
             cat_aclt (1) -- access category.
             dft_spec_aclt (2) -- default access provided by
             specific ACL.
             dft_cat_aclt (3) -- default access provided by
             access category.
             acc_cat_aclt (4) -- object is an access category.
code:        standard error code (output).
```

If the name is null, the contents of the default ACL for the current directory are returned. If max_entries is zero, only acl_name and acl_type are returned. The acl_name returned (which is a full pathname) is determined by the following algorithm:

```
acl_name(object) = if (object category-protected)
                     then category name
                   else if (object specific-protected)
                     then object name
                   else acl_name(parent(object))
```

Acl_ptr points to a structure which looks like the following:

```
dcl 1 acl,
      2 version fixed bin,          /* Input, must be 2 */
      2 entry_count fixed bin,      /* Number of pairs */
      2 entries(*) char (80) var; /* <access_pair>s */
```

5.5.6 AC$RVT -- Revert to a password directory

The AC$RVT gate reverts the current directory to a password directory. Its calling sequence is:

```
dcl Ac$rvt entry (fixed bin);
call Ac$rvt (code);
```

code: standard error code (output).

AC$RVT will return an error code of E$CATF (access category found) if the directory contains any access categories, and E$ADRF (ACL subdirectory found) if it contains any ACL subdirectories. AC$RVT should be used sparingly, if at all, and is provided for compatibility reasons only.

5.5.7 AC$SET -- Create or replace an entire ACL

The AC$SET gate provides user programs with a method of creating and replacing the ACL belonging to a category or file.

```
dcl Ac$set entry (fixed bin, char (128) var, ptr, fixed bin);
call Ac$set (key, name, acl_ptr, code);
```

key: indicates caller's intentions (input).
Possible values are:
K$ANY -- create a new ACL if one does not exist,
K$CREA -- create a new ACL. If one already exists, return an error.
K$REP -- replace the contents of an existing ACL. If one does not exist, return an error.

name: pathname of the object to be protected (input).

acl_ptr: pointer to the ACL structure (input).

code: standard error code (output).

The acl_ptr points to a structure like that for AC$LST, above.

The action taker by AC$SET is determined by the type of the object named in the call and the key, as follows:

5.5.7.1 The named object is an access category

If the key is K$CREA, an error is returned. Otherwise, the category's existing ACL is replaced with the new one pointed at by acl_ptr.

5.5.7.2 The named object is a file

If the file is protected by a specific ACL and the key is K$CREA, an error is returned. Otherwise, a new specific ACL is created and the object pointed to it. Any old specific ACL is deleted. If the object is a password directory and its parent is an ACL directory, it will be converted to an ACL directory.

5.5.7.3 The named object does not exist

If the key is not K$REP, a new access category is created with the given name and ACL. Otherwise, an error is returned. It is the responsibility of the caller to ensure that the category name ends in ".ACAT".

5.5.8 CALAC$ -- Calculate access available on an object

The CALAC$ call allows programs to determine the accesses available to the user on any given file system object. Its calling sequence is:

```
dcl Calac$ entry (char (128) var, ptr, char (80) var,
                  char (80) var, fixed bin) returns (bit (1));
have_access = Calac$ (name, id_ptr, access_needed,
                      access_gotten, code);

name:          pathname of the object to check (input).
id_ptr:        pointer to the user id structure (input).
access_needed: a list of accesses required (input).
access_gotten: the list of accesses available (output).
code:          standard error code (output).
have_access:   true if access_needed is a subset of
               access_gotten (returned).
```

The user id structure pointed to by id_ptr is the same as that for GETID$, below. If id_ptr is null, the current user's id and groups are used. Use of non-null ID pointers is designed for protected subsystems only, and should be avoided by most callers.

The access_needed and access_gotten strings are in ASCII format, that is, they are strings consisting of mnemonic mode names or the special modes "ALL" and "NONE".

If the name is null, the rights for the current directory are returned.

If the object is password-protected, password rights are returned. If the CALAC$ call is made on the current directory, the string "Owner" is returned if the user has owner rights, and "Non-owner" is returned if the user is attached with non-owner rights. For files, a string of the form "<owner_rights> <non_owner_rights>" is returned, where the rights strings will be either a combination of the characters "r", "w" and "d" or the special string "nil". For password-protected objects the access_needed string is ignored and have_access is always set to true. This should pose no serious problems to callers of CALAC$ since the only programs which really benefit from access checking are protected subsystems, and none currently exist.

5.5.9 CAT$DL -- Delete an access category

Access categories may be deleted with the CAT$DL call. Its calling sequence is:

```
dcl Cat$dl entry (char (128) var, fixed bin);
call Cat$dl (name, code);

name: name of the category to be deleted (input).
code: standard error code (output).
```

The name must exist and must specify an access category. Specific ACLs may not be explicitly deleted; they are deleted by the system when the file which they protect is either deleted, is put into an access category, or reverts to default protection.

5.5.10 CREA$$ -- Create a directory

Under the ACL system there are two different types of directories: password directories and ACL directories. Although the calling sequence of CREA$$ has not changed, it will now create a directory of the type of the parent. That is, if CREA$$ is used in an ACL directory it will create an ACL directory; if used in a password directory it will create a password directory. Password directories may be explicitly created with the CREPW$ routine, below. There is no special routine to create ACL directories since CREA$$ will always create an ACL directory within an ACL directory, and an ACL directory may not have a password directory as its parent.

5.5.11 CREPW$ -- Create a password directory

CREPW$ is identical to the existing CREA$$, except it will create a password directory instead of an ACL directory.

5.5.12 DIR$RD -- Read directory entries

DIR$RD replaces RDEN$$ in primary function, that is, to read the contents of a directory sequentially, entry by entry. The positioning functionality of RDEN$$, which could only get people into trouble, has been eliminated. The returning of an entry with a given name has been taken over by ENT$RD, below. The calling sequence for DIR$RD is:

```
dcl Dir$rd entry (fixed bin, fixed bin, ptr, fixed bin,
                  fixed bin);
call Dir$rd (key, unit, return_ptr, max_return_len, code);
```

key:	indicates what to do (input): when (K$INIT), initialize to directory header. when (K$READ, 0), read from current position.
unit:	unit number on which directory is open (L access must be available on the directory; input).
return_ptr:	pointer to user's buffer (input, points to output).
max_return_len:	size of user's buffer (input).
code:	standard error code (output).

The return_ptr points to a structure with the following format:

```
dcl 1 dir_entry based,
      2 ecw,
        3 type bit (8),
        3 len bit (8),
      2 name char (32),
      2 pw_protection bit (16) aligned,
      2 non_dft_prot bit (1) aligned,
      2 file_info,
        3 long_rat_hdr bit (1),
        3 dumped bit (1),
        3 dos_mod bit (1),
        3 special bit (1),
        3 rwlock bit (2),
        3 spare bit (2),
        3 type bit (8),
        2 dtm,
          3 date,
            4 year bit (7),
            4 month bit (4),
            4 day bit (5),
          3 time fixed bin,
        2 spare(2) fixed bin;
```

All entries are as defined in the Subroutines Guide description of RDEN$$ except for non_dft_prot, which is set to true if the entry is not default-protected (that is, is protected specifically or by a category).

DIR$RD only returns entries for named objects. Thus, unlike RDEN$$ it will not return the ECW for the directory header. Since old partitions are no longer supported, the only types which a caller of DIR$RD will see are 2, for a file or directory, and 3, for an access category.

NOTE

Calls to DIR$RD and ENT$RD should not be made on the same directory file unit unless DIR$RD is called with the K$INIT key following each ENT$RD call.

5.5.13 ENT$RD -- Read directory entry with given name

ENT$RD is identical to DIR$RD in what it returns, but rather than going sequentially through the directory, ENT$RD returns data for a particular named entry. Its calling sequence is:

```
dcl Ent$rd entry (fixed bin, char (32) var, ptr, fixed bin,
                  fixed bin);
call Ent$rd (unit, name, return_ptr, max_return_len, code);

unit:           unit number on which the dir is open
                (L access is required; input).
name:           name of the entry to read (input).
return_ptr:     pointer to return structure (input,
                points to output).
max_return_len: size of user's buffer (input).
code:           standard error code (output).
```

The structure returned by ENT$RD is identical to that returned by DIR$RD. As noted above, however, ENT$RD and DIR$RD should not be used together on the same file unit.

5.5.14 GETID$ -- Determine a user's full identity

The GETID$ call returns the user's id and groups. Its calling sequence is:

```
dcl Getid$ entry (ptr, fixed bin, fixed bin);
call Getid$ (id_ptr, max_groups, code);

id_ptr:     pointer to full_id structure (input, points to
            output).
max_groups: maximum number of groups caller's full_id
            structure can handle (input).
code:       standard error code (output).
```

The structure pointed to by id_ptr looks like:

```
dcl 1 full_id,
      2 version fixed bin,
      2 user_id char(32) var,
      2 group_count fixed bin,
      2 groups(*) char(32) var;
```

version
    Version number of the structure. This must be supplied by the caller, and must currently be either 1 or 2.

user_id
    The user id of the current user.

group_count
    Number of groups returned to the caller. This will always be the minimum of max_groups as supplied by the user and the number of groups the user has. Currently, users may have up to 32 groups. If max_groups is zero, this field is not returned.

groups
    The list of groups currently valid for the user.

5.5.15 ISACL$ -- Get directory type

Since for purposes of compatibility ACL directories and password directories have the same type (as returned to users; internally they are different), some method of distinguishing between the two is needed. ISACL$ returns (PL/I) true if the directory specified is an ACL directory. Calling sequence:

```
dcl Isacl$ entry (fixed bin, fixed bin) returns (bit (1));
is_acl_dir = Isacl$ (unit, code);

unit:       file unit to check. -1 = current dir, -2 = home,
            -3 = initial (input).
code:       standard error code (output).
is_acl_dir: true if directory on "unit" is an ACL directory
            (returned).
```

5.5.16 PA$DEL -- Delete a Priority ACL

Priority ACLs are removed with the PA$DEL gate, callable only by user 1 and the system administrator. Its caling sequence is:

```
dcl Pa$del entry (char (32) var, fixed bin);
call Pa$del (partition_name, code);

partition_name: name of the partition from which to remove
                priority ACL (input).
code:           standard error code (output).
```

Note that both PA$DEL and PA$SET depend on the ability to determine who the system administrator is. This information is read from the SAD and placed into SUPCOM in ring zero at cold start. See FE-TI-853 for details on system administrators and the SAD.

5.5.17 PA$LST -- Read a Priority ACL

Priority ACLs may be read by any user with the PA$LST call. Its calling sequence is:

```
dcl Pa$lst entry (char (128) var, ptr, fixed bin, fixed bin);
call Pa$lst (name, acl_ptr, max_entries, code);

name:        name of any object on the partition whose priority
             ACL is to be read (input).
acl_ptr:     points to return structure (input, points to
             output).
max_entries: most entries caller can handle (input).
code:        standard error code (output).
```

Normally, some access to the partition is required in order to determine the logical device number and through it get the priority ACL. Since it is possible to disallow all access to a partition with priority ACLs, however, if PA$LST is called with only a partition name (in angle brackets), it will merely look the partition up in the disk table and no access is required.
Note that PA$LST does not actually search for the last entry in the pathname, but merely attaches to its parent. Thus, PA$LST may return information even if the object specified by the full treename is unavailable or non-existent.

5.5.18 PA$SET -- Add a Priority ACL

Priority ACLs may be added to a partition with the PA$SET call, which may be used only by user 1 and the system administrator. Its calling sequence is:

```
dcl Pa$set entry (char (32) var, ptr, fixed bin);
call Pa$set (partition_name, acl_ptr, code);

partition_name: name of the partition to be protected (input).
acl_ptr:        pointer to ACL structure (input).
code:           standard error code (output).
```

The acl_ptr points to an ACL structure as for AC$LST. Any existing priority ACL on the specified partition will be replaced by the new one.

5.5.19 RDEN$$ -- Read directory entries

RDEN$$ has been modified to return access category entries (with a type of 3) and the first reserved word now contains a bit indicating whether the object described in the entry is not default-protected. The calling sequence is unchanged. RDEN$$ is obsolete and has been replaced by DIR$RD, above. New structures returned are documented in the section on DIR$RD.

5.5.20 SATP$$ -- Set file attributes

The SATR$$ routine has a new key, K$SDL, which is used to set the delete switch. If the contents of ARRAY(1) are non-zero, the delete switch is set. If ARRAY(1) is zero, the switch is cleared. The SATR$$ calling sequence has not changed; for details see the subroutines guide.

6 BADSPOT HANDLING

6.1 INTRODUCTION

Previously, the way the physical copy utilities handled badspots on a source partition was by reading a file in the MFD called BADSPT which was created by MAKE if a partition was found to contain bad records when it was initialised. This BADSPT file provided information which enabled PHYSAV and COPY_DISK to avoid reading a track which contained a bad record. All records in this track were also marked 'in-use' in the DSKRAT file so that the File Management System was not able to access the bad record.

6.1.1 BADSPT file format

6.1.1.1 Old format

Currently, the BADSPT file is a save memory image. The file may be examined and modified by restoring it and referencing it with PSD/VPSD. BADSPT is restored into consecutive memory locations starting at location '1000 and ending at '1000 + 2 * N - 1 where N is the number of bad tracks in the partition. Each word pair in the BADSPT file contains the track and head numbers of a defective track on the disk. The BADSPT file is created by MAKE and is used by FIXRAT/FIX_DISK, COPY_DISK, PHYSAV, PHYRST and AINIT.

6.1.1.2 New Format

The new format BADSPT file has two major enhancements over the previous format:

1) Single bad records are marked, rather than whole tracks.

2) An EQUIVALENCE block has been defined, which enables software using the BADSPT file to indicate that it has been able to avoid writing to a badspot by writing the record elsewhere.

Since new format badspot files are not user-modifiable, their internal format is not documented here.

Starting from Rev 19, each partition will have a Rev Stamp. The new BADSPT file format will be allowed only on a Rev 19 style partition.

6.1.2 Source_Badspot_Handling

PHYSAV and COPY_DISK were previously only able to avoid reading bad tracks as marked in the BADSPT files of the source partitions. Therefore, these two utilities have been enhanced to avoid reading individual bad records as marked in the new format BADSPT files.

6.1.3 Target_Badspot_Handling

PHYRST and COPY_DISK previously had no notion at all of badspots on the target partitions. Target disk badspot handling has therefore been added to them.
Badspot handling for a target disk involves more than just avoiding bad records on the target disk, as the record that would have fallen on the badspot needs to be written elsewhere, i.e. it needs to be mapped to an available free record. The only way that PHYRST and COPY_DISK can know whether a particular record is free is by checking the DSKRAT file. Therefore, for each badspot entry in the target BADSPT file, a free record must be found from the source DSKRAT file, and the entry that says to which record address the badspot has been mapped to must be stored in the BADSPT file. This is achieved by adding four-word entries to the EQUIVALENCE block of the BADSPT file.

During the restore or disk copy, the programs can then access the EQUIVALENCE block of the target BADSPT file to:

1) map records that would have fallen on a badspot,

2) avoid overwriting those records that have had badspots mapped onto them

N.B. The records that have been mapped contain exactly the same information as the original record (except for the CRA).

6.1.4 FIX_DISK

Badspot handling for FIX_DISK involves fixing the file pointers associated with the bad records to point to the remapping records using the remapped information that is contained in the EQUIVALENCE block of the new format BADSPT file. It also makes available the good records on the target disk which correspond to bad records on the source disk. After FIX_DISK is run, the EQUIVALENCE block of the new format BADSPT file will be removed.

6.2 INITIAL_STATE_OF_PARTITIONS

As a rule, the format of the target partition will be dictated by the source partition. Also, the badspot handling feature will be available only for rev 19 format partitions. Because the target badspot handling involves using the DSKRAT file of the source partition to find free records, then the DSKRAT file must be correct. If you cannot be sure this is so, then FIX_DISK should be run on the source partition.

6.3 FINAL_STATE_OF_PARTITIONS

PHYSAV and COPY_DISK will leave source partitions exactly as they were.

PHYRST and COPY_DISK will only leave target partitions as exact copies of the original source partitions if the command line option -NOBADS is used, or if the source partition was a pre rev19 partition.

Otherwise:

1) If the target disk originally had a BADSPT file then afterwards it will contain that BADSPT file with the appended EQUIVALENCE block. Records will have been remapped as indicated by the EQUIVALENCE block.

2) If the target disk did not originally have a BADSPT file then afterwards it will still not contain a BADSPT file.

The exception to these rules is if badspot handling has been turned off by the program - for example if no free records were available on the partition for bad records to be mapped onto. In this case there will be no BADSPT file left on the target disk.

6.4 Compatibility

COPY_DISK, PHYSAV/RST will handle pre rev 19 partitions exactly as before. In other words, the target partition will be an exact copy of the source partition, and no badspot handling will be provided.
In this case, the message:

```
WARNING - SOURCE PARTITION IS PRE REV 19
NO BADSPOT HANDLING WILL OCCUP ON PARTITION pdev
```

will be issued.

6.5 USER_INTERFACE

If badspot handling has taken place during PHYRST or COPY_DISK, then for each affected partition the message:

BADSPOTS HANDLED ON PAPTITION pdev

will be put to the terminal at the end. FIX_DISK must be run on that partition before it is used for any reason other than as a target disk for PHYRST or COPY_DISK.

If the situation occurs where PHYRST or COPY_DISK are attempting to map a record round a badspot and there are no free records available, the message:

NO FREE RECORDS AVAILABLE ON PARTITION pdev
OK TO WRITE TO IT WITHOUT BADSPOT HANDLING (YES/NO)?

will be issued to the terminal. If the user types YES then the partition will be copied to without badspot handling, otherwise the program will exit, to allow the user to copy to a different partition with fewer badspots.

Upon finding a BADSPT file (on source or target partitions) which is in some way inconsistent, the message:

BAD BADSPT FILE ON PARTITION pdev - IGNORED

is issued.
If the BADSPT file of a source partition contains an EQUIVALENCE block, then the program will abort with the error message:

BADSPT FILE ON PARTITION pdev HAS AN EQUIVALENCE BLOCK
PLEASE RUN FIX_DISK

N.B. A BADSPT file not marked as special in the MFD is completely ignored.

7 COMMAND_PROCESSOR_ENHANCEMENTS

7.1 INTRODUCTION

This section describes the enhancements made to the Primos command processor at Revision 19. These changes are intended to provide the following general features in the command language:

- o simple commanc iteration (executing a given command once for each of an explicitly listed set of objects).
- o wildcard processing (executino a given command once for each file system object that satisfies certain selection_criteria).
- o name generation (generating file system names from a given name and a pattern on the command line).
- o treewalk processino (executino a given command over selected parts of a file system subtree).

7.2 OVERVIEW

Simple Iteration is indicated by means of an explicitly parenthesized list of blank-separated tokens. All other features are recognized by virtue of the special characters they must contain (such as @, ^ or + in a wildcard) or their exact character value (such as -after).

All of these features are conditionally processed, depending on whether the command "wants" the command processor to handle them or wishes to do so itself.

Internal commands provide "descriptor" information to the command processor to inform it whether the various features are to be processed by the command processor or not. This information is stored in the internal command table inside Primos.

If the command is a Static Mode (user) program, then the command processor makes the following default assumptions:

- o simple iteration, wildcards, generation patterns, treewalk pathnames should be expanded by the command processor
- o the generation pattern source name is the first object argument on the command line, except for the RFSUME and SFG commands, which are special-cased to cue oeneration patterns on the second object argument

- verification should not be requested by default

Exceptions

For CPL commands (whether invoked with RESUME, CPL or external commands) the command processor will perform simple iteration but will never expand wildcards, treewalk pathnames or generation patterns. Thus, CPL commands may do their own wildcard and name generation processing using the primitives provided in CPL.

Static Mode commands whose name begins with NX$ are treated like CPL commands. Static Mode commands whose name begins with NW$ have treewalk pathnames only expanded, but not wildcards or generation patterns.

The combination of these prefixes and CPL allow a site (or a user) having an existing command that does its own wildcarding or that already uses some of the selector control arguments (-AFTER, etc.) to continue to work.

Suppose such a command is FOO. The command is renamed to NX$FOO, and a simple CPL command FOO.CPL is added which looks like:

```
&args args: rest
nx$foo %args%
```

Then, users of the FOO command can continue to invoke FOO, which now invokes FOO.CPL instead. FOO.CPL simply passes the unexpanded arguments on to NX$FOO. The user is unaware of any of this.

7.3 FEATURE DETAILS

7.3.1 SIMPLE ITERATION

A simple iteration set is a blank-separated list (commas are accepted as in regular command lines) enclosed in parentheses. Nested parentheses are not allowed. If the command processor has been instructed to process simple iterations, the command is executed once for each item in the list. The current item replaces the iteration set in the command line on each pass. For example:

```
delete (a b c)
```

is equivalent to the three command lines

```
delete a
delete b
delete c
```

More than one iteration set may be present on the command line. In this case, the corresponding items are taken from each list on each pass. For example,

```
copy (a b c) (d e f)
```

is equivalent to

```
copy a d
copy b e
copy c f
```

If a list runs out of items before other lists do, the null string gets substituted for it on all subsequent passes.

A cross product effect may be obtained by using a pair of iteration sets with no intervening blanks or commas. The cross product is computed only between pairs, but there may be several cross product pairs per command line. An example is:

```
delete (a b c).(list bin)
```

which is equivalent to

```
delete a.list
delete a.bin
delete b.list
delete b.bin
delete c.list
delete c.bin
```

7.3.2 WILDCARDS

(The term "wildcard" as used here refers to the system standard wildcard convention for file system names, documented in the Primos User Guide.)

If the command processor has been instructed to process wildcards for this command (i.e. there is no command descriptor and the command name does not start with NX$ or NW$; the command is not a CPL command; or the descriptor so instructs), it will look for an object argument that appears to be a wildcarded file (path)name. If none is found, it is assumed that no wildcard processing is desired.

Only one wildcard pathname is allowed per command. The files specified are those in the directory given by the pathname, whose names match the wildcard part of the pathname.

The set of files selected may be further controlled via selection option arguments. These are special, reserved option arguments that are recognized by the command processor if wildcard processing is enabled. These option arguments are deleted from the command's arguments whether or not a wildcard pathname is used.

The selection option arguments may be chosen in any order from:

-before DATE, -bf DATE
: allows a file to be selected only if its Date Time Modified (DTM) is less than or equal to DATE. DATE is in the system standard date input format.

-after DATE, -af DATE
: is like -before, except the file is selected if its DTM is greater than or equal to DATE.

-file
: allows a file to be selected if it is a SAM or DAM FILE.

-directory, -dir
: allows a file to be selected if it is a DIRECTORY (UFD).

-segment_directory, -segdir
: allows a file to be selected if it is a SAM or DAM SEGMENT DIRECTORY.

-access_category, -acat
: allows a file to be selected if it is an ACCESS CATEGORY. NOTE: if none of -file, -dir, -seodir or -acat is given, then files of all types are selectable.

-verify, -vfy
: requests that the user be asked to verify each match with the wildcard(s) and selection criteria before any command is executed. Any matches that the user vetoes will be omitted from the set used for execution. The default is not to request verification, but certain internal commands specify that verification be requested by default.

-no_verify, -nvfy
: inhibits verification even if the command requested that it be performed by default.

The command processor then computes the list of files that match the wildcard and all the selection criteria. If verification is enabled, then the user is asked to give approval of each match before any command is actually executed.

A response of "next" to a verification question will cause the execution of the command for the given wildcard pathname to be aborted without processing any of the matched files. If simple iteration or treewalking is in use, the command processor will continue with the next iteration specified by them. The break key may be used to abort all processing.

The other legitimate responses are "yes", "y", "ok" for affirmative, "no", "n" or blank for negative. Upper and lower case are equivalent.

Then, the command is executed once for each match. The (path)name of each file matched is substituted in place of the wildcard pathname.

Example:

```
slist smith>dir>@.pl1 -file -after 7-1.12
```

might result in command lines which look like:

```
slist smith>dir>abc.pl1
slist smith>dir>def.pl1
            .....
```

and prints all PL1 files modified after 12:00 on July 1 of the current year.

Multiple Wildcard Pathnames

It was noted above that only one wildcard pathname is allowed per command. If it is desired to operate in a single command line on files in different directories using wildcards, or if more than one wildcard is needed to specify the files, wildcards may be combined with simple iteration as shown in the following examples:

```
delete (a>b>c>@@.list d>e>f>@@.list) -before 1-1
```

Simple iteration expansion occurs first, so that we have the equivalent command lines:

```
delete a>b>c>@@.list -before 1-1
delete d>e>f>@@.list -before 1-1
```

which then perform the wildcard deletions in the two directories, one after the other, as outlined above. Note that we could also have entered this command as:

```
delete (a>b>c d>e>f)>@@.list -before 1-1
```

and saved a little more typing at the expense of visual clarity.

An example showing use of multiple wildcards to specify the files is:

```
slist a>b>(@.pl1 @.ftn)
```

which prints all PL1 or FTN files in the directory a>b.

7.3.3 GENERATED NAMES

(The term "name generation" as used here refers to the system standard name generation convention for file names, documented in the Primos Users Guide.)

If the command processor has been requested to process generation patterns, it does so by recognizing them as object arguments containing at least one "=" character. A particular object argument on the command line is defined by the command as the generation pattern source argument; it is usually the first object argument.

The command processor replaces each generation pattern by the name it generates from the source argument. If the source argument is a wildcard (path)name, the current match filename is used as the source. Generation pattern processing is thus conceptually at the same time as wildcard processing if the latter is in use.

Examples:

```
mrgf a>b>filename.v1 a>b>=.v2 a>b>=.v3 -outf =.v4
```

will expand to:

```
mrgf a>b>filename.v1 a>b>filename.v2 a>b>filename.v3
        -outf filename.v4
```

Also

```
plp a>b>@.plp -b *>obj>=.bin -before 8-15.12
```

might expand to

```
plp a>b>abc.plp *>obj>abc.bin
plp a>b>def.pl1 *>obj>def.bin
            .......
```

and will compile all PLP files in a>b modified before noon of August 15 of the current year into directory *>obj.

WARNING: wildcard match names will <u>not</u> be used as the generation pattern source unless the wildcard (path)name argument is in the generation pattern source position on the command line. For example in:

```
mrgf foo.pl1.master a>b>foo.pl1.@ =.=.v1 =.=.v2
```

the generation pattern source is "foo.pl1.master", <u>not</u> the names that match the wildcard. This is because the wildcard is not the first object argument of the mrgf command.

7.3.4 TREEWALKING

If the command processor has been requested to perform treewalking, it examines the command line for treewalk pathnames. A treewalk pathname is a pathname whose directory portion contains exactly one wildcard name. (The final component may or may not be a wildcard, as appropriate.) No more than one treewalk pathname is allowed per command.

The non-wildcard directory part of the treewalk pathname specifies the starting directory. For example, in

```
a>b>c>@.src>@@.list
```

the starting directory is a>b>c. Beginning at this directory, each subdirectory whose name matches the directory wildcard name is visited. The order of visitation is top-down depth-first. The starting directory may or may not cause a command execution, at the user's option; the default is that it not cause an execution.

Each subdirectory whose name matches the directory wildcard name (@.src in the example) causes an execution of the command. Subdirectories whose name does not match the directory wildcard do not cause a command execution, but such directories are searched for subdirectories whose name might match. Exception: the starting node causes a command execution whether its name matches or not, if the user has requested that it cause a command execution.

In the example, the order might be:

```
a>b>c   {causes an execution only if requested by user}
a>b>c>abc.src
a>b>c>abc.src>modules.src
a>b>c>abc.src>primitives.src
a>b>c>def.src
a>b>c>ghi.src
a>b>c>ghi.src>z.src
a>b>c>ghi.src>z.src>zilch.src
    ......
```

For each directory that causes a command execution, the command is executed as if the directory part of the treewalk pathname up to the wildcard were replaced by the pathname of that directory. Hence, if the final component is a wildcard name, wildcarding is performed in each directory in which the command is executed.

For example, to delete all listing files in an entire subtree starting at a>b, but not including those in a>b, one would say:

```
delete a>b>@@>@@.list
```

To print the file INFO in each "source" directory inferior to a>b>c, one might say:

```
slist a>b>c>@.source>info
```

It is also possible to reference specific subdirectories of the walked tree. For example,

```
delete a>b>@@>trash>@@.junk
```

might delete all JUNK files in the subtree a>b, but only if they reside in a subdirectory called "trash". Usage of this construct is likely to produce error messages if some subdirectories of a>b do not contain a "trash" subdirectory.

Treewalk_Option_Arguments

Certain reserved option arguments can be used to control the order and depth of the walk. These arguments are always recognized by the command processor when a treewalk pathname is used, unless the command processor has been instructed not to process treewalk pathnames for this command.

When recognized by the command processor, these option arguments are omitted from the command line actually executed. The options are:

-walk_from N, -wlkfm N
: N is a positive decimal integer. Specifies that the command should only be executed in directories at level >= N with respect to the starting directory. The starting directory is level 1. The default is to start execution at level 2, the level below the starting directory. To include the starting directory and cause a command execution in it, use -walk_from 1.

-walk_to N, -wlkto N
: N is a positive decimal integer. Specifies that the walk should stop after level N with respect to the starting node. The default is to walk all levels.

-bottom_up, -botup
: specifies that the walk is to proceed in a bottom-up, depth-first order. An example of such an order is:

```
a>b>c>d>e
a>b>c>d>f
a>b>c>d>g
a>b>c>d
a>b>c>h
a>b>c>i
a>b>c
a>b>j
a>b     {causes execution only if -walk_from 1 given}
```

For example, to delete all files with a single component name from all subdirectories of a>b, including from a>b itself, one would use:

```
delete a>b>@@>@ -walk_from 1
```

7.4 INTERACTION_OF_COMMAND_ENVIRONMENT_FEATURES

Now that so many new features have been added to the Primos command processor, it has become necessary to define the order in which they are processed and the interactions between them. This information will be presented in the form of a chronology of the processing of a command line.

7.4.1 Abbreviation Expansion

The first thing that happens to the command line is that abbreviations are expanded if they are enabled. The abbreviation processor treats command functions and simple iteration lists as single tokens. This affects only their disposition when functions or iteration lists require abbrev parameters or are used as an argument for an abbrev parameter. For example,

```
let a = *>subdir>%1%

a (b c d)
```

expands to:

```
*>subdir>(b c d)
```

Thus the list (b c d) was treated as a single token and bound to parameter 1 of abbreviation "a".

```
additionally let b = *>bdir>%1%
and let c = *>cdir>%1%

(a b c) d e f
```

expands to:

```
(*>subdir>d *>bdir>d *>cdir>d) e f
```

Thus the list (b c d) was treated as a single token having 1 parameter to which "d" was bound.

Command function invocations are also treated as single tokens, but they have the additional property that they are expanded as if they were a separate command line. Thus, the first token following a left bracket is considered in the command position, and no abbrev parameters are taken from beyond the matching right bracket. For example,

```
let a = foo %2%.two %1%.one

[a b] c d
```

expands to

```
[foo .two b.one] c d
```

That is, "c" did not become the second parameter of "a" because "c" lies outside the function call brackets.

7.4.2 Syntax Suppressor

Following abbreviation expansion, the command processor checks to see if the first character on the command line is the syntax suppressor, "~". If so, processing of all the features described in the following sections is suppressed, and the command line is executed as-is with the "~" removed.

7.4.3 Multiple Command Processing

The command processor scans the command line for the command separator character, ";". This character delimits multiple commands on the same command line. The recognition of the command separator is disabled if the syntax suppressor was used, or if the command is ABBREV or AB. The latter exception is provided so that it is easy to define new abbreviations whose value contains the command separator character. For example,

```
abbrev -ac zot close all; delete @@ -no_verify
```

is a single command that defines an abbreviation, "zot", whose value is "close all; delete @@ -no_verify".

If recognition of the command separator is not disabled, the features described below are executed separately for each subcommand on the command line. For example, in

```
command1 [function1 args]; command2 [function2 args]
```

the order of execution is: evaluate function1; execute command1; evaluate function2; execute command2.

7.4.4 Variable and Function Evaluation

Once the current subcommand has been identified, variable references are evaluated. Each reference of the form "%variable_name%" is replaced by the value of "variable_name".

Next, command function references of the form "[function arguments]" are evaluated and replaced by their values. Evaluation proceeds from the inside out.

The fact that variables and functions are evaluated after the command separator has been processed means that, if the value of a variable or function contains a command separator, it will not be recognized as such.

The fact that functions are evaluated after variables means that, if the value of a variable contains a function reference, the function will be evaluated. Conversely, if the value of a function contains a variable reference, it will not be evaluated, since variable evaluation has already occurred.

If any error occurs during variable or function evaluation, such as a reference to an undefined variable or function, the command processor prints an error message and discontinues processing that subcommand.

7.4.5 Simple Iteration

All simple iteration sets (parenthesized lists) in the command line are identified. Conceptually, the command processor can be thought of as producing a series of command lines, one for each iteration specified by the simple iteration sets. For example,

```
com (a b c) y (d e f)
```

can be thought of as the series of commands

```
com a y d
com b y e
com c y f
```

In actual fact the command processor does not generate the command strings at this time; it deals with a kind of list structure at this level.

Each command line of the series is then further processed for treewalking, wildcarding and name generation, as outlined below.

7.4.6 Treewalking

The command line is examined to see if it contains a treewalk pathname. This is a pathname whose directory part (the part before the final name) contains a wildcard. There may be at most one such pathname per command (there may be more than one on the command line so long as each iteration yields no more than one).

The command processor will open the directory whose pathname appears before the directory wildcard in the treewalk pathname. It will visit the subdirectories of the tree as explained in the section on Treewalk Iteration above, and substitutes the pathname of each directory visited for the part of the treewalk pathname to the left of and including the directory wildcard.

For example,

```
com a>b>@@>@.list
```

might execute as if it were the series of commands

```
com a>b>c>@.list
com a>b>c>x>@.list
com a>b>c>y>@.list
com a>b>d>@.list
```

and so forth.

Commands of this series are conceptually passed on the the wildcarding step, below.

7.4.7 Wildcarding

Each command is scanned to see if it contains a pathname whose last component (entryname) is a wildcard. There may be at most one of these per command received at this step.

At the same time, the command is scanned for reserved option arguments (such as -after) which specify selection criteria that are applied in addition to the wildcard name. These reserved option arguments are removed from the final command whether or not a wildcard name is used.

The command processor opens the directory given by the treename, and selects those entries in the directory that match the wildcard and the other selection criteria. If verify mode is enabled, the command processor will ask the user to approve or disapprove each match.

The wildcard part of the pathname is then replaced with the actual name matched, and the command is passed on to the next step.

For example,

```
com @.list -after 12-1 -file
```

might execute as if it were the series

```
com a.list
com b.list
com c.list
   ....
```

7.4.8 Name Generation

The final step is name generation. The command processor searches for any pathname in the command that contains the character "=" in the entryname. Any number of such pathnames is permitted. Each "generation pattern", as a name containing an "=" is known, is replaced by the name it generates. The source name is usually the first object argument to the command, although individual commands may differ.

For example,

```
com abc.list ==.+old
```

would execute as

```
com abc.list abc.list.old
```

7.4.9 Execution

The command that emerges from the name generation step is then actually executed. After execution, whether the command produced a positive severity code (ER! prompt) or not, the next wildcard match, then the next treewalk step, and finally the next simple iteration step, is taken.

8 DISK_QUOTAS

8.1 When_Are_Quotas_Useful

Quotas are useful to limit the number of records that a directory may use. They may also be used to meter disk record usage and to size directories.

8.2 What_Are_Quotas

Quotas are limits placed on directory size. The limits are in disk record units. No directory with a quota is permitted to obtain records causing it to exceed its quota. This restriction is enforced on the entire subtree. If multiple quotas are in effect at various levels of the subtree, then the most restrictive quota is enforced.

A quota is always a positive integer value. No quota (quota = 0) allows unlimitted usage. Negative quotas are not allowed.

8.3 Commands

8.3.1 List_Quota (list quota information)

```
List_Quota <pathname> [-BRief]
```

Lists quota information for directory <pathname>. If <pathname> is omitted, the current attach point is used.

Output is of the form:

```
Maximum records allowed on "<pathname>" = <quota>.
Total records used = <tree_used>.
Records used in this directory = <dir_used>.
```

where <quota> is the quota max for the directory, <tree_used> is the number of records used in the directory tree and <dir_used> is the number of records used in this directory level. If the current directory's quota is being listed (<pathname> is null), the string "<Current directory>" will be substituted for the pathname in the output. If <pathname> is not a quota directory, the first line of output will state '"<pathname>" is not a quota directory.' rather than listing the maximum quota.

If the -BRIEF option is used, output is presented in the tabular form:

Max: <m>, Used: <t>, Records: <d>[, <pathname>]

where <m> is the maximum quota allowed (zero if <pathname> is not a quota directory), <t> is the tree used count, <d> is the directory used count, and <pathname> is the name of the directory (omitted if null). This format is particularly useful when using wildcards to size all subdirectories of a given directory.

Quotas are enforced by never allowing Total records used to exceed Maximum records allowed. Also neither Total records used nor Records used in this directory may be less than one.

8.3.2 Set_Quota (set_maximum_quota)

Set_Quota <pathname> -Max <n>

Set the maximum quota on directory <pathname> to the value <n> records. The user of the command must have Protect (or Owner) access to the parent directory of the directory whose quota is being set.

Exception condition 'file in use' may be raised when setting a quota on a directory without a current quota. The exception will occur if there are active users of the directory or its subtrees at the time the quota change from zero is requested. Note that this will occur on CMDNC0 when the system console is attached there. Also note that this only occurs when changing a quota from a current value of zero to a positive value. If the directory already has a non-zero quota this exception will not occur.

8.4 Subroutines

8.4.1 Q$READ (name, buf, buflen, type, code) - return quota info

This routine returns information about quota counters and the time-record product of disk record usage for the specified quota directory.

The caller must have Protect access to the directory, or Owner rights if it is a password directory.

When this primitive is invoked on a non-quota directory the type will be 1, quota related information (Max, time-record, product, and time) will be zero. Directory records used will indicate the sum of the records used by the files in that directory plus the records used by the directory file itself. Total records used will indicate the sum of the records used for all files inferior to this directory node. Quota directories will return a type equal to 0, and all of the quota

information. Directory records used and total records used will be the same as in the non-quota directory case.

The routine will enter as many values into the BUF array as is specified by BUFLEN, up to a maximum of 8. Entries which are reserved for future use will have an undefined value.

Usage: DCL Q$READ ENTRY (char(128) var, (8) fixed bin(31), fixed bin, fixed bin, fixed bin);

CALL Q$READ (NAME, BUF, BUFLEN, TYPE, CODE)

NAME Pathname of the directory whose quota is to be read. (input)

BUF An array containing the quota information. (output)
BUF(1) Data size of disk record (440 or 1024 words).
BUF(2) Number of records used in directory.
BUF(3) Number of records of MAX quota. (0 if non-quota)
BUF(4) Amount of total records used.
BUF(5) Time-record product (record-minutes). (0 if non-quota)
BUF(6) Date/Time last updated. (0 if non-quota)
date format is 16 bits word one,
YYYYYYYMMMMDDDDD
time is word two seconds since midnight divided by four
BUF(7) Reserved for future use.
BUF(8) Reserved for future use.

BUFLEN Number of entries in BUF. (input)

TYPE Type of directory. (input)
0 = Quota Directory.
1 = Non-Quota Directory.

CODE Standard error code. (output)
E$BPAR = BUFLEN is less than 1.
E$NTUD = Object is not a directory.
E$IMFD = Attempt to read quota on MFD (currently illegal).
E$NOQD = Disk has not been formatted for quotas.
E$NINF = List access was missing from the directory or its parent.
E$NRIT = Use access was missing at some intermediate node in the tree.

8.4.2 Q$SET (key, name, amount, code) - set quota max

This routine sets maximum quota on the specified directory. If the named directory is not already a quota directory, it will become one.

The following restriction applies to the use of this subroutine:

> The caller must have Protect access to the parent directory, or be its Owner if a password directory.

Usage:
```
DCL Q$SET ENTRY (fixed bin, char(128) var, fixed bin(31),
        fixed bin);

    CALL Q$SET (KEY, NAME, AMOUNT, CODE)
```

KEY — K$SMAX = Set maximum quota. (input)

NAME — Pathname of the directory whose quota is to be set. (input)

AMOUNT — The value to be set. (input)

CODE — Standard error code. (output)

- E$BKEY = A key other than K$SMAX was used.
- E$BPAR = A negative quota was supplied.
- E$DTNS = Date/time not set yet (required for record-time product).
- E$WTPR = The disk is write-protected.
- E$NRIT = Protect access was missing from the parent, or Use access was missing at some intermediate node in the tree.
- E$NINF = Use access was missing at some node in the tree, and List was not available on its parent; or, Protect and List were not available on the parent of the target directory.
- E$NTUD = The object is not a directory.
- E$IMFD = Quota not permitted on MFD.
- E$NOQD = Disk has not been formatted for quotas.
- E$QEXC = Used records greater than new maximum (WARNING).
- E$FIUS = The directory or one or more of its offspring was in use, and a quota is being set on this directory for the first time.

8.4.3 Use of the Accounting Meter returned by Q$READ

The system keeps an accounting usage meter in each quota directory. This meter is a summation of the time intervals that each disk record has been in use.

The accounting meter is a counter which acts as an unsigned number, which is to say that it counts to all ones and then goes to zero. The system also provides an indication of when the last update occurred.

The calculation used is given below. The USAGE is computed in record-minutes.

```
TIME = (Current date-time) - (Date-time quota last modified)
USAGE = USAGE + (Records used ) * TIME
```

An accounting program would use a similar algorithm to calculate the current record-time product.

8.5 Using Quotas

Disk record quota are an optional feature for each directory. If there is a maximum quota on a directory, then the system uses the quota restriction. Disk usage meters are recorded regardless.

8.5.1 Maximum Quota

Maximum quota is used to restrict a user to an amount of disk storage specified by the system administrator. This is accomplished by setting the MAX quota on the user's UFD. The subtree for the UFD will not be able to use more records than that maximum.

The maximum quota is an arbitrary value. The sum of the MAX quotas on all top level UFDs can exceed that which is available on the logical disk. In this case, the maximum quota does not guarantee the availability of records in the future. The user still competes with other users for available records, but the competition is controlled. Inferior directories can also have their MAX quota set. The setting is the same as that for top level directories. The Owner of the immediately superior directory merely sets the value of MAX quota of the target directory. The previous value, if any, is changed to the new value. The MAX quota of the current directory is unchanged. In this way the MAX quota is non-conserved.

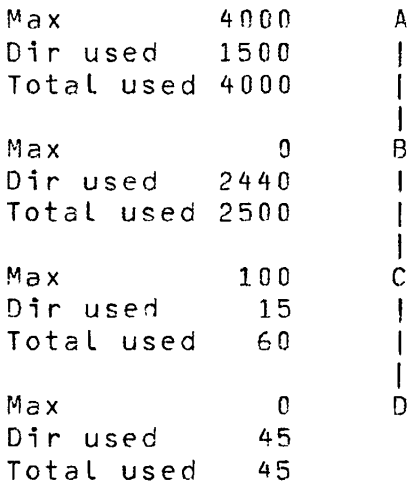
With a non-conserved system the project administrator has a privilege similar to that of the system administrator, namely that he can over-commit his allocated disk records. It should be noted, however, that he is not allowed to actually use more records than he is allocated.

The reason for this arbitrary maximum is to allow a user extra records on a short time basis for listings and other temporary files. Most users will periodically cleanup their directories in order to keep far enough below their maximum to allow ease in working. If this assumption is correct, the administrator can give out more records in maximum quota than actually exist because users will not use their allocated maximums at the same time. This gives better utilization of the disk, since users are sharing records for temporary use. Of course, if a site finds that most users tend to keep close to their maximum quotas and the disk full condition continues to occur, it can set the maximum quotas so that the total is equal to the size of the logical disk.

8.5.2 Quota Hierarchies

A logical disk can be made to use the quota feature by first modifying it with the new FIX_DISK. FIX_DISK will fill in the records used field in UFD headers. The owner of the MFD would then set the MAX quota on any top level UFDs he wishes. In this way he could restrict the number of records used by those UFDs. A directory only becomes a quota directory when its max quota is set.

When a file is created or extended the quota system will check all superior directories to insure that a MAX quota is not exceeded. In the example below B and D are non-quota directories and are therefore not checked. Let us take the example of adding a record to directory D. D has no quota so its quota is not checked. Its parent C has a quota of 100 records and a total records used of 60 records leaving a difference 40 records. 40 records minus the one we are adding is 39 which is greater than zero so we pass the test for directory C. Directory B has no quota so we go on to directory A. A has a quota of 4000 records and a total records used of 4000 leaving 0. Subtracting the record we wish to add leaves -1 records which is negative and we fail the quota test for directory A. Therefore, the record will not be allocated and a error will be returned.


Max 4000 A
Dir used 1500
Total used 4000
Max 0 B
Dir used 2440
Total used 2500
Max 100 C
Dir used 15
Total used 60
Max 0 D
Dir used 45
Total used 45

9 FILE_SYSTEM_UTILITY_COMMANDS

9.1 INTRODUCTION

The following sections are intended to provide a complete description of the file system utility commands. The commands are designed to perform the following basic functions:

- o File, segment directory, directory, and access category copying.
- o File, segment directory, directory, and access category deletion.
- o Setting the read/write lock for files and segment directories.
- o Displaying the contents of a directory.
- o Setting the protection keys for files and segment directories.

These commands are intended to replace, but are not compatible with, the current FUTIL subsystem. See the section, New Command Processor Features, for information about applying these commands to multiple files or directories in a simple manner.

Document_conventions

- o Lower case text enclosed in angle brackets ("<" and ">") represents an object whose actual value should be substituted, upper case text indicates a literal value. For example, "<date>" means substitute a calendar date and "ALL" would mean use the literal value "ALL".

- o Text enclosed in square brackets ("[" and "]") represents optional objects. Two or more objects separated by spaces represent optional choices, two or more objects separated by vertical bars, "|", represent a choice of mutually exclusive options.

- o objects followed by "..." represent multiple occurrences of such objects.

9.2 COPY

COPY will copy files, directories, segment directories, and access categories.

Usage: COPY <source_object> [<target_object>] [control_arguments...]

source_object

A standard treename specifying the location and name of the object to be copied. Read (R) access is required on this object.

target_object

A standard treename specifying the destination and name of the target object. If the target_object is omitted, the target directory is assumed to be the current directory, and the source object name is used for the target name. Append (A) access is required on the directory containing the target object. Delete (D) access is required on the directory containing the target object if the target object already exists.

9.2.1 control_arguments

Zero or more control arguments specified in any order from the the following list:

-QUERY, -Q

Specifies that COPY is to request that the user resolve unexpected or potentially dangerous situations. This is the default mode of operation.

-NO_QUERY, -NQ

Specifies that COPY is NOT to request the user's permission but to attempt to resolve those situations in the most intuitive fashion.

-LEVELS, -LV [<dec>]

Specifies that COPY is only to copy down to the level specified by "dec" when copying a directory tree. "dec" is a decimal integer from 0 to 999. If "-LEVELS" is omitted, the default is to copy the entire tree; if "dec" is omitted, the default is 0 (only copy the top level, the directory entry itself and none of its subentries).

-REPORT, -RPT

Specifies that COPY is to report the results of each successful copy operation.

-DELETE, -DL

Specifies that COPY is to delete the source object once it has been copied. The default is no deletion. This option requires delete (D) access on the source directory.

-DAM

Specifies that all SAM files copied are to be converted to DAM files. The default is to preserve the original file type.

-SAM

Specifies that all DAM files copied are to be converted to SAM files. The default is to preserve the original file type.

-FORCE

Specifies that COPY is to force delete rights for all delete-protected objects selected to be deleted. This includes both a target object that already exists and the source object if "-DELETE" is selected. This argument is most useful when overwriting a directory tree that may contain delete protected objects. This option requires protect (P) access on the appropriate directory.

The default is to request the user's permission to force delete an object, unless "-NO_QUERY" was specified, in which case the protected object(s) will NOT be deleted.

-INCREMENTAL, -INC

Specifies that COPY is only to copy those objects whose dump bit is off (= 0). (I.E. those files that have NOT been dumped to tape.) The default is to copy objects regardless of the dump bit setting.

This argument is intended to provide functionality similar to that provided by the MAGSAV INCREMENTAL command.

Note that if a directory is the object of the command, all entries within that directory are copied, regardless of their dump bit setting.

-REPLACE

Specifies that COPY is to only copy those objects which exist in the target directory.

9.2.2 Attribute_copying_arguments

The following arguments specify which attributes of selected objects are to be preserved or reset by COPY. If none are specified, the default is to use the system default. If one or more are specified, only those attributes are preserved, the rest will be reset to the system default.

The use of any of these arguments requires protect (P) access on the appropriate directory.

-DTM

Specifies that COPY is to preserve the date/time modified stamp of all source objects copied. The system default is to reset the date/time modified to the current date/time.

When a directory is copied, the use of this argument will cause the date/time modified stamp of each subentry in the directory to be preserved.

-PROTECT, -PRO

Specifies that COPY is to preserve the protection attributes of all source objects copied. This is done by protecting the target object with a specific access control list. The default is to use the default access in the target directory.

-QUOTA

Specifies that when a directory is copied the maximum quota information associated with it and any of its subdirectories is to be copied also. The system default for maximum quota information is no limit, i.e., there is no restriction on the maximum directory size.

-RWLOCK, -RWL

Specifies that COPY is to preserve the read/write locks of the source object. The default is to set the read/write locks to the system default.

Note that only files (i.e., DAM and SAM) and segment directories have user alterable locks, for all other file system types copied the read/write locks will have the system default.

-COPY_ALL, -CA

Specifies that COPY is to preserve all the attributes. It is the same as specifying "-DTM -PROTECT -QUOTA -RWLOCK".

9.2.3 Restrictions

- COPY will not allow the MFD, BOOT, or DSKRAT files of a MFD to be overwritten. In order to copy a boot file to a MFD the user should first RESTore the new boot to memory and then SAve it with the name "BOOT". Note that this restriction does not apply when these files exist in other than a MFD.

9.2.4 Usage_under_password_directories

Under password directories the requirement for access is different. In all cases owner access is needed on the target directory. Delete access is need on the appropriate file if COPY is going to delete it (source if "-DELETE" and/or target if it exists). If "-PROTECT" is specified then all the password parts of protection are copied. Protection attributes include protection keys (files, directories, and segment directories), and passwords (directories only).

The system default for protection keys is rwd nil (owner has all rights, nonowner has none); for passwords owner is blank, nonowner is null.

Copying the passwords of a directory requires owner rights in the source directory, if that directory is a password directory. If the user does not have owner rights COPY will request the user's permission to copy the directory. If "-NO_QUERY" was specified, the directory will be copied without requesting the user's permission. If the directory is copied, it will acquire the system default passwords.

9.3 DELETE

DELETE will delete files, directories, segment directories, and access categories.

Usage: DELETE <target_object> [control_arguments...]

target_object

A standard treename specifying the location and name of the object to be deleted. Delete (D) access is required on the target directory.

9.3.1 control_arguments

Zero or more control arguments specified in any order from the following list:

-QUERY, -Q

Specifies that DELETE is to request that the user resolve unexpected or potentially dangerous situations. This is the default mode of operation.

-NO_QUERY, -NQ Specifies that DELETE is NOT to request the user's permission but to attempt to resolve those situations in the most intuitive fashion.

-REPORT, -RPT

Specifies that DELETE is to report the results of each successful deletion.

-FORCE

Specifies that DELETE is to force delete rights for all delete-protected objects selected. This argument is most useful when deleting a directory tree that may contain delete protected objects. This option requires protect (P) access on the appropriate directory.

The default is to request the user's permission to force delete an object, unless "-NO_QUERY" was specified.

9.3.2 Restrictions

- DELETE will not delete the MFD, BOOT, or DSKRAT files in a MFD. Note that DELETE may be used to delete these files if they exist in other than a MFD.

9.3.3 Implications

- Query will always be requested for directory and access category deletion, unless "-NO_QUERY" was specified.
- Verification is requested of the wildcards handled by the command processor. The command processor option "-NO_VERIFY" will suppress this.

9.3.4 Usage under password directories

Under password directories the requirement for access is different. In all cases delete access is needed on the target object. If the file does not have delete then owner access is needed on the target directory.

9.4 LD - List Directory

LD displays a directory and, optionally, the various attributes of entries in the directory. The user may select entries based on all the standard command processor ways and also by how the object is protected.

Usage: LD [<target_object> [<wild_cards>...]] [control_arguments...]

target_object

Specifies both the directory to be listed, and the first wildcard name. For example, "a>b>@.list" would specify entries in the directory A>B whose names match "@.LIST". If pathname is omitted, "@@" is assumed; that is, all entries in the current directory are selected.

wild_cards

Specify additional wildcard names. An entry is selected if it matches either the entryname part of pathname or one of the wild_cards.

9.4.1 control_arguments

Zero or more control arguments specified in any order from the following list:

-NO_HEADER, -NHF

specifies that the header line is not to be output. The header line contains the pathname of the directory listed, the access rights (in parentheses), the records used by this directory if available, and the quota used if this is a quota directory.

-SPECIFIC_PROTECTED, -SPECP

specifies that those entries that are specific protected will be selected.

-DEFAULT_PROTECTED, -DFLTP

specifies that those entries that are default protected will be selected.

-CATEGORY_PROTECTED, -CATP [<cat_name>]

specifies that those entries that are protected by the access category "cat_name" will be selected. If "cat_name" is missing then all entries that are protected by access categories will be selected.

-NO_SORT, -NSORT

Specifies that the entries listed not be sorted. The default is to sort by ascending NAME within TYPE. TYPEs are always sorted according to the order: file, segment directory, directory, access category.

-SORT_DTM, -SORTD

specifies that the entries be sorted by descending DTM within TYPE. -SORT_NAME must not also be specified.

-SORT_NAME, -SORTN

Specifies that the entries be sorted by ascending NAME only (not within TYPE). -SORT_DTM must not also be specified.

-REVERSE, -RV

Specifies that the sort order be reversed from its default. Note that the sort order of TYPEs is never affected.

-DETAIL, -DET

Specifies that all attributes be displayed for each entry selected. From left to right these are:

access rights available to this user (for password directories, the protection keys are displayed).

size of entry in physical disk records.

quota of entry in physical disk records (directories only).

type of entry.

setting of concurrency lock on entry (" " for system, "excl" for N readers or 1 writer, "updt" for N readers and 1 writer, and "none" for N readers and N writers).

incremental dump switch ("dmp" if the entry has been dumped).

delete-protection switch ("pr" if protected).

date-time modified.

name of entry.

and type of protection (name of access category protecting entry, or (Specific) for specific protected, or blank for protected by default);

The default output format is to list only the name of each entry, four across. To print a subset of "detail" format information, use one or more of the following options.

-PROTECT, -PRO

Specifies that protection information (access mode, delete-protect switch, and type of protection) be printed for each entry.

-DTM

Specifies that date-time-modified be printed for each entry.

-SIZE

Specifies that size information (size of entry, quota for directories only) be printed for each non-access category entry. A size of -1 will be reported for any entry for which the user does not have R (or L) permission.

-SINGLE_COLUMN, -SGLCOL

Is useful only if the default (names only) format is used. In this case, specifies that names are to be printed one per line instead of four per line.

9.4.2 Usage under password directories

Under password directories the access listed is the protection keys as owner nonowner.

9.5 RWLOCK

RWLOCK will set the read/write concurrency locks for files and segment directories.

Usage: RWLOCK <target_object> [<lock>] [control_arguments...]

target_object

A standard treename specifying the object whose read/write lock is to be modified. This command required protect (P) access on the target directory.

lock

Read/write lock, may be one of the following:

SYS - use system read/write lock (default)
EXCL - N readers OR 1 writer (exclusive OR)
UPDT - N readers AND 1 writer
NONE - N readers AND N writers

9.5.1 control_arguments

Zero or more control arguments specified in any order from the following list:

-REPORT, -RPT

Specifies that RWLOCK is to report the results of each successful lock change operation.

9.5.2 Restrictions

- Only files and segment directories currently have user alterable read/write locks. If a wildcard name is specified with no file type selection arguments, only files and segment directories will be selected.

9.5.3 Usage_under_password_directories

Under password directories the requirement for access is different. In all cases owner access is needed on the target directory.

9.6 PROTECT

Set protection keys for files, directories, and segment directories. This command is useful only in password directories.

Usage: PROTECT <target_object> [<owner> [<nonowner>]]
[control_arguments...]

target_object

Standard treename specifying the object to be protected. Owner access is needed on the target directory.

owner, nonowner

Protection keys, must be selected from the following list:

nil - no access (default)	rw - read/write access
r - read access	rd - read/delete access
w - write access	wd - write/delete access
d - delete access	rwd - read/write/delete access

Note: the order of letters is not important. I.E. "wd" is the same as "dw".

If either owner or nonowner is omitted, the default is nil - no access.

9.6.1 control_arguments

Zero or more control arguments specified in any order from the following list:

-REPORT, -RPT

Specifies that PROTECT is to report the results of each successful protection operation.

9.6.2 Restrictions

o PROTECT requires protect (P) access in the target directory.

9.6.3 Implications

- Although the PROTECT command may be used to modify the protection keys of objects in ACL directories, the keys are ignored when accessing those objects. But if the directory were converted back to a password directory, the changed protection keys would be in effect.

- If a wildcard name is specified with no file type selection arguments, the default will be to select files, directories, and segment directories.

10 Overview of FIX_DISK

FIX_DISK reads every physical record in every file, UFD, and segment directory, and checks that the information in each record header is consistent with the UFD that contains the record. If the current UFD is a Quota UFD, FIX_DISK also checks the consistency of its quota information. If any inconsistency exists, an error message is generated.

FIX_DISK builds its own record availability table (RAT) while it is traversing the existing file structure and compares its RAT with the DSKRAT file. If discrepancies are found, an error message is generated.

If recuested, FIX_DISK will attempt to repair mismatched pointers, correct quota information, truncate/delete defective files, and replace the defective DSKRAT file. The disk will then be in a consistent state.

If requested, FIX_DISK will convert a pre-rev 19 partition into a rev 19 partition. It involves initializing the quota information, changing the BADSPT file to the new format, and creating a rev stamp. If the current partition is a rev 19 partition and a equivlence section exisit in the BADSPT file, FIX_DISK will map the bad records into their equivlence records and fixes the file system pointers to point to the equivlence records. When FIX_DISK has completely traversing the file system structure, the equivlence section of the BADSPT file will be deleted from the BADSPT file. If a badspot is encountered in a rev 19 partition, it will be added to the BADSPT file. If the BADSPT file does not exist, one will be created.

FIX_DISK determines whether a UFD is a quota UFD by examing the maximum quota word in the UFD header. If it is not zero, it is a quota UFD. If the MFD of a partition is a quota UFD, that partition is a quota partition. Otherwise it is not a quota partition and quota information fields are ignored. When FIX_DISK has finished traversing all the subtrees of a quota UFD, the quota information is checked against the records used determined by FIX_DISK. If any inconsistency exists, an error message is generated. If requested, the incorrect quota information is fixed unless the quota used is greater than the maximum quota. Because FIX_DISK cannot and should not decide which records to release to correct the problem, it just marks the quota system as in an inconsistent state. Since the records used of this quota UFD has exceeded its quota, it cannot draw any addition records. The user must delete records or increase the directory's quota to resolve this conflict. FIX_DISK determines whether a UFD is an ACL UFD by the file type field for the UFD in the UFD entry of it's parent. FIX_DISK will verify that for an ACL UFD file entries point to valid ACLs or Access Categories or default, Access Categories point to valid ACLs, and ACLs point back to the same object that points to them. If there is an error and fixing has been requested, for file entries with bad ACL pointers, it will set the ACL pointer of the file entries to the default value. Access Catecories or ACLs with errors will be deleted.

FIX_DISK should be run on a regular schedule or whenever there is reason to expect that the file structure or the quota system is damaged.

10.1 Usage:

FIX_DISK -DISK <physical disk> [control arguments]

<physical disk> is the the physical disk number on which FIX_DISK is to be run. The disk MUST be assigned first, unless the -comdev option is being used.

The control arguments are optional. They may be selected in any order from the list below. If no control argument is selected, FIX_DISK only generates error messages if errors are detected.

-fix

Besides printing file structure error messages, FIX_DISK corrects quota information, truncates or deletes defective files, generates a corrected DSKRAT if the current one is bad, and maps the badspot records to the BADSPT file if -fix is specified. If omitted, FIX_DISK will not perform any disk modifications.

-ufd_compression {-cmpr}

If specified along with -fix, FIX_DISK compresses UFDs, eliminates entries flagged as being deleted files or directories.

-command_device {-comdev}

If specified, the disk being fixed is the command disk and FIX_DISK must be invoked via the system console. FIX_DISK will be the only user in the system. If there is any other users, they will be logged out automatically.

-no_quota {-nq}

If specified, it assumes that the partition is not a quota partition and the quota checking mechanism in FIX_DISK will be turned off.

-convert_19

If specified, the current partition will be converted into a rev 19 style disk. If a BADSPT file has already existed, it will be converted into the new format. All quota information is initialized, and all warning/error message related to quota will not be printed. A rev stamp will be created. This option must be used with -fix option.

-level [n]

If specified, the decimal number n that follows is the lowest level in the tree structure in which directory names are to be printed. If omitted, FIX_DISK will print up to level 2 directories (MFD and all directories in MFD file).

-file

If specified, the file names in all directories are printed.

-max_nested_level {-max}

If specified, the decimal number that follows is the maximum depth that directories are allowed tc be nested. If omitted, the maximum depth is set to 100. (see -auto_truncation)

-auto_truncation {-at}

If specified, FIX_DISK automatically truncates directories that are nested too deeply in a directory tree. If omitted, FIX_DISK will abort if the maximum depth is reached.

-interactive {-int}

If specified, and the current DSKRAT is bad or missing, questions will be asked so that FIX_DISK can reconstruct a consistent DSKRAT. If omitted and the current DSKRAT is bad or missing, FIX_DISK will abort.

The motivation of implementating this feature is to allow users to replace a bad or missing PAT. FIX_DISK computes the number of records in the partition from the disk number. In case of ambiguity, FIX_DISK asks resolving questions, answerable by either YES or NO.

-dufe (delete unknown file entry)

If specified, all unknown file entries are eliminated. If omitted, all unknown file entries are left untouched, no compressions are performed on the UFDs in which the unknown file entries reside and the DSKRAT will not be altered except in the case of the DSKRAT indicates a particular record is free but that record is actually in use.

The motivation of implementing this feature is to avoid accidental deletion of valid file entries by running the wrong version of FIX_DISK. (e.g. an older version that does not recognize the new file types has to be run.) However there is a drawback of not deleting unknown file entries. The File System advances to the next file entry by using the length field of the current file entry. If the current file entry is garbage, the File System may bypass good file entries by using its length field.

10.2 Description of Error Messages

The backward pointer is bad. It should be YY instead of XX

The backward pointer of a record does not point back to the previous record of the file. In the case of the first record of a file, its back pointer is not zero. If -fix option is specified, the back pointer is fixed to point to the previous record if the BRA word of this record matches the first record address of this file. The file is truncated if the BRA word of this record does not match the first record address of the file.

The Beginning Record Address (BRA) pointer is bad. It should be YY instead of XX

The beginning record address word of the records within the file except the first record should point to the first record of the file. If -fix option is specified, the BRA pointer is fixed.

System file is bad, ignored

An error, which would normally cause deletion of a file, has been found in one of the special files BOOT, MFD. or DSKRAT in the MFD. FIX_DISK aborts.

The current record address (CRA) is bad. It should be YY is XX

The current record address word of this record does not match the current address. This message may be preceded by ten disk error messages because this problem could indicate a disk drive problem. If -fix option is specified, the file is truncated.

UFD nesting exceeds maximum specified

Directories may be nested to a depth of N levels. (default N = 100) FIX_DISK cannot follow the directory tree because the user has nested directories to more than N levels. FIX_DISK ignores this directory unless -at option is specified in which case directories that are nested too deeply in the directory tree will be truncated.

The record header of DSKRAT file is bad
The number of heads is different. It should be YY is XX
The physical record size is different. It should be YY is XX
The DSKRAT header has wrong length. It should be YY is XX

The information contained in the DSKRAT header does not correspond to the information computed from the disk number. Either the disk number is incorrect or the DSKRAT header contains incorrect information. If -int option is omitted, FIX_DISK aborts. Otherwise FIX_DISK asks:

FIX DSKRAT?

A NO response causes FIX_DISK to abort.

The file structure of DSKRAT is bad

This message is obtained if the DSKRAT file contains any bad record pointers, or contains inconsistent information. If either -int or -fix is omitted, FIX_DISK aborts. Otherwise FIX_DISK attempts to reconstruct the DSKRAT file. FIX_DISK computes the number of records in the partition from the disk number. In case of ambiguity, FIX_DISK asks resolving questions, answerable by YES or NO, such as: 40 MB storage module?

FIX_DISK then asks

Split partition?

If part of the disk is to be used for paging then answer YES, otherwise answer NO. If the answer is YES, FIX_DISK then asks

Paging records (decimal)?

The user should type in the number of records to be used for paging.

FIX_DISK then prints the disk number, file records, and paging records.

Partition XX File-records XX Paging-records XX

and asks:

Parameters OK?

If the numbers are incorrect, answer NO and FIX_DISK will attempt to recompute the numbers again.

The father pointer is bad. It should be YY is XX

The father record address word of the first record of a file does not point to the beginning record address of the file in which this file is entered (its father). If -fix option is specified, the father pointer is fixed to point to the BRA of its father.

Unknown file type XX, Record = YY, Word = ZZ

The file type XX in the file entry is unknown. It is either an illegal or a new file type that is not known by this version of FIX_DISK. If -dufe option specified, this file entry is deleted. If omitted, this file entry is left untouched, no compression is performed for the UFD in which this file entry resides, and the DSKRAT will not be altered except in the case of the DSKRAT indicate a particular record is free but that record is actually in use.

File_type_mismatch

The file type in the first record of the file does not agree with the file type of the UFD entry. If -fix option is specified, the following action will be taken. The file is deleted if both file types are either legal or illegal. Otherwise, the one that is legal is assumed.

The_first_file_entry_of_the_MFD_file_is_not_DSKRAT
The_second_file_entry_of_the_MFD_file_is_not_MFD
WARNING:_The_third_file_entry_of_the_MFD_file_is_not_BOOT
The_partition_may_need_to_be_remade

FIX_DISK checks that the first three entries in the MFD are DSKRAT, MFD, and BOOT. If any of these entries is missing, FIX_DISK aborts.

The_UFD_header_is_missing or The_length_of_the_UFD_header_is_incorrect

The UFD header is either missing or contain bad data. If -fix option is specified, the UFD file is deleted.

The_forward_pointer_XX_is_bad,_it_is_not_in_the_range_of_the_current_partition

The address that the forward pointer points to is not between zero and the maximum record address of this partition. If -fix option is specified, the file is truncated.

The_forward_pointer_XX_points_to_a_record_that_belongs_to_another_file

The record that the forward pointer points to belongs to another file. This error may occur if the current DSKRAT is bad or the BADSPT file is changed after the previous FIX_DISK was run. If -fix option is specified, the file is truncated.

The_forward_pointer_of_the_top_level_index_record_of_a_DAM_file_is_not_zero

The top level index must only be one record long, therefore the forward pointer of this record must be zero.

The_index_level_of_this_DAM_file_is_incorrect._It_should_be_YY_instead_of_XX

The index level word of this record is incorrect. It should be zero for SAM files or one less than the previous level for DAM files. If -fix option is specified, the index level word is fixed.

The_DAM_index_is_too_long_to_represent_the_DAM_file

The data records of a DAM file are shorter than its index indicates. If -fix option is specified, the index is truncated.

The index of this DAM file is too short to represent the data records

The data records of a DAM file are longer than its index indicates. If -fix option is specified, the index is fixed.

The tree used count is bad. It should be YY instead of XX

The tree used word of this quota UFD does not match the quota used that is calculated by FIX_DISK. If -fix option is specified, the tree used is fixed.

The directory used count is bad. It should be YY instead of XX

The directory used count word for this directory (all the files and nonquota UFDs belong to this directory and the directory file itself) does not match the directory used count that is calculated by FIX_DISK. If -fix option is specified, the directory used count is fixed.

The next index does not match the forward pointer of the current data record

The pointers of the index section and the data section do not agree. If -fix option is specified, the following actions will be taken. The back pointer of the record that is pointed to by the DAM index and the back pointer of the record that is pointed by the forward pointe of the current data record are examined. The record with the back pointer points to the previous data record will be chosen. If neither back pointer points to the previous record or both back pointer point to the previous record, the file is truncated.

Inconsistent entry. Record = XX, Word = YY

Information in a file entry in a UFD is not self-consistent and cannot be reconciled. If -fix option is specified, the entry of this file is changed to vacant.

Disk read/write error. Record = XX Track = YY Head = ZZ sk An error occured while reading/writing record XX. If -fix option is specified, the file is truncated and this badspot record is added to the BADSPT file.

EOF occurs in the middle of an entry

A directory ends in the middle of the last UFD entry. If -fix option is specified, the entry will be deleted.

The Quota system may be incorrect

This message is issued if the partition was changed under DOS. Since DOS doesn't support quotas, there may be directories on this partition with incorrect quota information.

Partition not shutdown correctly during the previous session

This message is issued if the partition was not shutdown with the SHUTDOWN command under Primos. If the system crashed or the disk drive was spun down instead, this message will result.

The word count of record XX is bad

The data word count of a record is not reasonable. For every record except the last record, the data word count should equal the record data size. The data word count of the last record should be between zero and the record data size. If -fix option is specified, the word count is set to record data size.

Physical Device number {-DISK} is missing

The physical device number is not specified in the command line.

Bad physical device number

The physical device number that is specified in the command line is bad.

2 files point to the same record

Two or more files on this partition use the same record. If -fix, the second or later file to reference the record will be deleted.

The Directory/Segdir is longer than 64K!

The maximum size of a UFD/SEGDIR is 64K words. If one exceeds this limit, it will be truncated if -fix is specified.

The BADSPT file is bad, ignored

The BADSPT file that is found by FIXDISK is bad, this file will be treated just like an ordinary file instead of a special BADSPT file.

File entry at word XX does not reference an ACL or Access Category

The ACL pointer of a file entry doesn't point to a valid ACL or Access Category. If -fix, it is changed to the default value.

Access Category at word XX does not reference an ACL

The ACL pointer of an Access Category doesn't point to a valid ACL. If -fix, it is deleted.

Access category at word XX is not pointed at by ACL it points to

The ACL pointer of an Access Category points to an ACL which doesn't point back to it. If -fix, it is deleted.

File_entry_at_word_XX_is_not_pointed_at_by_ACL_it_points_to

The ACL pointer of a file entry points to an ACL which doesn't point back to it. If -fix, it is set to the default value.

ACL_at_word_XX_does_not_point_to_a_file_entry_or_Access_Category

The owner pointer of an ACL doesn't point to a file entry or Access Category. If -fix, the ACL is deleted.

ACL_at_word_XX_is_not_pointed_at_by_object_it_points_to

The owner pointer of an ACL points to an object which doesn't point back to it. If -fix, it is deleted.

Cannot_allocate_segment_for_XX

Fix_disk tried to dynamically allocate a segment for XX and failed. Fix_disk will abort.

11 EVENT LOGGING

Event logging within Primos consists of keeping track of certain predefined important events which occur to the system as a whole. Events are separated into two classes: system events and network events. Each class of events is logged by a separate, but similar, mechanism.

11.1 Event Logging Changes For Rev19.0

The event logging mechanism has been changed for Rev19.0. The intended goals were as follows:

(1) To provide user 1 with the capability to properly separate his home and current attach points.

(2) To provide the system administrator with more control over the event logging mechanism.

(3) To create a more efficient mechanism.

11.1.1 Enabling and Disabling Event Logging

11.1.1.1 Rev18

Event logging is normally enabled by default. Two methods are available in order to enable or disable event logging.

(1) The config directives LOGREC and NETREC will enable event logging if set to a non-negative value. If positive, the value represents a maximum soft quota on the number of file system records allowed for the corresponding event logging files, LOGREC and NETREC in the ufd CMDNC0. These values are set at 4096 by default. If negative, event logging is disabled.

(2) Event logging could be disabled while the system is running by simply changing the name of either LOGREC or NETREC in CMDNC0. To reenable, simply change the filenames back to LOGREC or NETREC.

11.1.1.2 Rev19.0

As with the event logging mechanism present in Rev18, event logging is normally enabled by default. Rev19.0 also provides two methods to either enable or disable both system and network event logging.

(1) The config directives LOGREC and NETREC will disable event logging if set to a negative number and will enable event logging if set to 0 or a positive value. These config directive no longer set a "nonenforced quota" on the event logging files in Rev19.0. The size of the event logging files may be restricted by setting a file system quota on the respective event logging directories, via the "SET_QUOTA" command.

(2) The second method of enabling or disabling event logging is via the "EVENT_LOG" command. This is extra functionality for Rev19.0. This command may be used to either enable or disable both network and system event logging while the system is running. The user interface is simply: "EVENT_LOG [-NET] -ON | -OFF".

The Rev18 method of enabling or disabling event logging while the system is running. i.e., changing the name of either LOGREC or NETREC, is not supported in Rev19.0.

11.1.2 Event Logging Database

11.1.2.1 Rev18

Event logging in Rev18 takes place to two files, "LOGREC" and "NETREC", both located in the command directory, CMDNC0. The files "LOGREC" and "NETREC" do not exist on the master disk, and the event logging mechanism does not create "LOGREC" or "NETREC" if they do not exist. The system administrator must create these files for event logging to take place. A system administrator may effectively disable event logging by changing the names of these files.

Both system and network event logging is performed by user 1, during the one minute update period. If event logging was enabled and the logging files exist, these files are opened and closed approximately once every minute.

11.1.2.2 Rev19.0

11.1.2.2.1 Event Logging Directories

The system event logging file resides in the ufd LOGREC* and the network event logging file resides in the ufd PRIMENET*, both on logical device 0. Poth ufds are present on the master disk. If they get deletec event looging will not take place until there is some human intervention, as there exists no code to create them if they oo not exist.

11.1.2.2.1.1 Access Rights

System event logging is performed by user 1, i.e., "SYSTEM". Therefore, user "SYSTEM" should be given "ALL" access rights to the ufd "LOGREC*". Network event logging is performed by user "NETMAN". Therefore, user "NETMAN" should be given "ALL" access rights to the ufd "PRIMENET*". System administrators and system operators should have the capability to both purge and write to the input logging files, via the LOGPRT command. Therefore, the ".ADMINISTRATORS" group should be given at least "ULPW" access rights to both "LOGREC*" ano "PRIMENET*". Regular users, i.e., "$REST", should be given at least "ULR" access rights to both directories.

11.1.2.2.2 Event Logging Files

Event logging in Rev19.0 takes place to two files, "LOG.MM/DD/YY" for syster event logging and "NET_LOG.MM/DD/YY" for network event logging. The string"MM/DD/YY" refers to either the date on which a cold start of the machine occurred or to the date on which the "EVENT_LOG" command created a new event logging file. That is, for both classes of events, a new event logging file is created only if a cold start occurred for the first time on a given day or the "EVENT_LOG" command is issued on a day which does not correspond to the date of the present event logging file.

11.1.2.2.3 Assuring Stable Attach Points

Network event logging is performed by "NETMAN" every 30 seconds. System event logging is still performed asynchronously by user 1 during the one minute update. In order to have event logging take place as much as possible if explicitly enabled while at the same time preserving current attach points for both user 1 and NETMAN, both event logging files remain open while the system is running. This sharp difference between Rev18 and Rev19.0 introduces some complexities which are covered in the next section.

11.1.3 User Visible Changes Using The System Console

User 1 is normally connected with the system console. Since the system event logging file is opened with special protection rights so that it remains open, certain Primos commands will yield results that differ from those of Rev18.

11.1.3.1 "CLOSE" Command

The system event logging file may be opened or closed ONLY by issuing the EVENT_LOG command. Therefore, issuing the Primos "CLOSE" command will NOT close the file. An attempt to close the system event logging file explicitly via either the command "CLOSE <filename>" or "CLOSE <fileunit>", will yield either of the two error messages:

"C'ant close unit <fileunit> Insufficient Access Rights. close"
or
"Insufficient Access Rights. <filename> close"

An attempt to close the file via either the commands "CLOSE ALL" or "CLOSE -ALL" will also NOT close the file but will not print an error message.

11.1.3.2 "STATUS UN" Command

Typing the "STAT UN" command on the system console subsequent to typing a "CLOSE ALL" command will show that the system event logging file unit is still open.

11.1.3.3 Shutting Down Logical Disk 0

11.1.3.3.1 Rev18

Both event logging files are opened and closed during the time period that the actual logging to the disk files takes place. Therefore, shutting down and subsequently readding logical device zero will not effect event logging.

11.1.3.3.2 Rev19.0

For Rev19.0, both event logging files are opened ONLY when event logging is enabled, either at cold start or when the "EVENT_LOG -ON" command is issued. Network event logging is performed by a dedicated process, NETMAN, which is ALWAYS attached to the ufd PRIMENET*. If logical disk zero is shut down, the network event logging mechanism proceeds to (1) cleanup its database, and (2) attach back to PRIMENET*, and (3) attempt to reopen the network event logging file. Shutting down logical disk 0 will effectively disable system event logging since

the system event logging file is closed. Therefore, when the disk is readded, system event logging will not take place until event logging is reenabled via the "EVENT_LOG -ON" command. This is not considered to be a problem because logical device zero will most often be shut down only for system cold start. If this does occur, informative error messages will be printed on the system console to let the OPERATOR know the status of system event logging. See the next section for an explanation of system console error messages.

11.1.4 Error Handling

11.1.4.1 Rev18

The event logging mechanism in Rev18 ignores ALL errors incurred while logging significant events. The only time a message was sent to the console was when the event logging exceeded its "SOFT QUOTA". In other words, all errors were dropped on the floor at a time when it is most important to perform event logging.

11.1.4.2 Rev19.0

The event logging mechanism in Rev19.0 is designed so as to have event logging take place as much as possible. With that in mind, ALL errors are reported to the system console. ERROR MESSAGES ARE PRINTED TO THE CONSOLE EVERY "5 MINUTES" for both system and network event logging. Three types of errors are handled specially: (1) QUOTA EXCEEDED, (2) DISK FULL, and (3) DISK SHUTDOWN.

11.1.4.2.1 QUOTA EXCEEDED

Event logging files may grow indiscriminantly. In the interest of preserving disk space, it is advised that a quota be set on the directories LOGREC* and PRIMENET*, and that old input logging files be processed via the LOGPRT command, the resulting output files spooled, and the input logging files deleted. If the quota on either directory is exceeded while event logging is taking place, either of the two messages:

"Exceeding quota on LOGREC*. System event logging not taking place. (LOGEV2)"

or

"Exceeding quota on PRIMENET*. Network event logging not taking place. (NETEV2)"

is printed to the system console every 5 minutes. The problem may be corrected by (1) processing the old input event log files with LOGPRT, spool the output files, and deleting the old input log files or (2) increasing the quota on the corresponding directory. The error messages may be disabled by disabling event logging via the "EVENT_LOG"

command.

11.1.4.2.2 DISK FULL

This error is separate from the event logging mechanism but could frequently occur if the event logging files are allowed to grow without bound and the event logging directories are not purged of OLD input logging files. If this error occurs, either of the two messages:

"Disk full. System event logging not taking place. (LOGEV2)"
or
"Disk full. Network event logging not taking place. (NETEV2)"

is printed to the system console every 5 minutes. The problem may be corrected by (1) processing the old input event log files with LOGPRT, spool the output files, and deleting the old input log files or (2) by having the system administrator free-up unused disk space. The error messages may be disabled by disabling event logging via the "EVENT_LOG" command.

11.1.4.2.3 DISK SHUTDOWN

Shutting down logical device zero will close the event logging files if event logging was previously enabled. Subsequently adding the disk will not cause system event logging to be reenabled. If this occurs the following error messages:

"Disk has been shut down. System event logging not taking place. (LOGEV2)"
"Disk has been shutdown...Please reenable system event logging. (LOGEV2)"

will be sent to the system console ONCE and system event logging will be disabled. The only method of reenabling system event logging is to issue the "EVENT_LOG -ON" command.

11.1.4.2.4 Other Errors

All other errors will cause the following error messages:

"<text of error message> System event logging not taking place."
or
"Error <error number> detected while logging network event." "Network event logging not taking place."

to be sent to the system console every 5 minutes, until the error is corrected or event logging is disabled via the "EVENT_LOG" command.

11.1.5 Delogging By Using The LOGPRT Command

System input event logging files are located in the ufd LOGREC* and network input event logging files are located in the ufd PRIMENET*. The files NETREC and LOGREC no longer exist. There will probably be many event logging files in each directory, depending on the frequency of cold starts being performed and also how often the "EVENT_LOG -ON" command is issued to restart event logging.

There exists a '-INPUT' option to the LOGPRT command to indicate the name of the input logging file. The pathname will usually be of the form "LOGREC*>LOG.MM/DD/YY" or "PRIMENET*>NET_LOG.MM/DD/YY". If no input file name is specified, LOGPRT will use as a default file the most recently created event logging file in either of the directories LOGREC* or PRIMENET*.

12 USER PROFILES

12.1 Overview

In any installation, each user has both individual and group identities. While the definition and use of individual identities is essentially the same from installation to installation, group identities may vary. For instance, one installation might be homogeneous enough to have only one user group, another installation might have several groups but each user in only one group, and a third installation may have many groups, with each user in one or more of them. The User Profiles system addresses the entire range of those installations. The system is engineered in such a way that its services can be tailored according to the needs of various kinds of installations. User profiles functionalities could be broadly classified into two categories: User registration and operational control. User registration provides for secure and easy login to the system, whereas operational control provides for consistent usage and control of directories, files, and system resources like CPU, I/O, etc.

Four general concepts have been introduced here. These four concepts--the individual user, user groups, user registration and operational control--form the basis of the user profiles system. At revision 19, the user registration phase of profiles has been implemented.

12.2 Functional Definitions

12.2.1 User Profile

A User Profile is loosely defined as those parametric entities of a user's operating characteristics which create a unique environment for that user. Some of these parameters may be defined by the user while the others are defined by the system and/or project administrator. This information is stored in different files in the file system and access to them is controlled by the system administrator.

12.2.2 User Identity (User ID)

A User ID is a name up to 32 characters long, beginning with an letter, and followed by any combination of letters, digits, and the special characters ".", "_", and "$", with lowercase mapped to uppercase. A user id uniquely identifies a user to the system on which the id is defined and also to any other systems within the defining system's common naming sphere. The user id is not required to be the same as the User File Directory (UFD) name which is currently used in Primos to identify a user.

12.2.3 User Login Password

A User Login Password is a string of up to 16 characters which is known only by the user. It may contain any ASCII characters except Primos reserved characters, with lowercase characters mapped to uppercase. At login time the system checks the password supplied by the user against the pasword kept in encrypted form in the validation files. The user is allowed to log in only if the supplied password matches the saved one. Passwords for new users are set by the system administrator, but may be changed by either the user or system administrator at any time. Since passwords are stored in encrypted form, they may not be read by anyone. Each user technically must have a password, but if the system administrator allows it, that password may be "null" (all spaces), thus allowing the user to log in without supplying a password. Note that since the user profiles mechanism detaches user id from UFD, login passwords bear no direct relationship to the password, if any, on the user's initial attach point.

12.2.4 Project

A Project is defined to be a group of users with similar attributes and system usage.

12.2.5 Attributes

12.2.5.1 ACL Groups

Each user may be assigned up to 16 ACL Groups by the system administrator, and up to 16 more by each project administrator to whose project the user belongs. Projects may be assigned up to 16 groups by the system administrator. Groups assigned to projects by the system administrator limit the groups which the administrator of that project may assign to users. The system administrator has the power to choose whether groups may be assigned on a system-wide (per user) basis or on a project-based (per-project) basis, or both.

12.2.5.2 Initial Attach Point (ORIGIN)

An Initial Attach Point is a UFD or a sub-UFD in the file system to which the user is attached when he logs in. Since the user identity is no longer necessarily related to a top-level UFD in the file system, the initial attach point places the user in a specified place in the file system where he can commence his work. The initial attach point is a project-based attribute. It is represented by a pathname up to 16 levels deep, and is currently restricted to local partitions.

12.2.6 System Administrator (SA)

The System Administrator is a person who is responsible for controlling access to the system, distributing system resources, and generally monitoring the system. By definition, the system administrator is the most trusted person in the system. He also has the option of delegating responsibility to other users.

12.2.7 Project Administrator (PA)

A Project Administrator is a user who is responsible for maintaining activity within a particular project. He need not be a member of the project. The project administrator's responsibilities and privileges are delegated to him by the system administrator.

12.2.8 User

A User has fewer privileges than the system and project administrators. His project and group affiliations are granted to him by either the system administrator or the project administrator.

12.3 User Registration

A password validation scheme is used, that is, each user is required to have a personal password. Before a user can use the system, he must register with the system administrator. Upon registration, a user is given a user identity which uniquely identifies him to the system, a user login password which is associated with that id, and one or more project affiliations. All of these assignments are under the control of the system administrator.

12.3.1 Password Validation

Every user must have a personal password. If that password is null, however, the user effectively has no password. The system administrator may decide for security reasons that every user must have a non-null password. This option is controlled by a command to the Profile Editor, described in a later section of this document.

12.3.2 Project Validation

After the user has successfully passed the password validation, the system will attempt to connect the user to a project. A user must belong to at least one project. This may be the special project "DEFAULT," however, which is reserved for systems choosing not to use projects or for users not belonging to any specific project.

12.3.2.1 User Does not Specify the Project

If a user does not supply a project at login time the system will use the user's default login project. This default project id is set by the system administrator, and may specify any of the valid projects in the system. If the system administrator chooses not to assign a default login project for a user, the system will always ask for a project id.

12.3.2.2 User Specified Project ID

If the user supplied a project id at login the system will attempt to connect the user to the project specified. If unsuccessful, the user will be barred from logging into the system.

12.3.3 Attribute Set Up

Once the validation process has been completed, the system proceeds to set up the user's attributes according to his profile. At the project level, any attributes not supplied by the user's own profile are taken from the project profile. That is, if the user's profile shows no groups, and the project profile has groups ".DEFAULT" and ".OPSYS," the user will be given groups ".DEFAULT" and ".OPSYS."

12.3.3.1 ACL Access Groups

After a user is logged in and connected to a project, the system will activate all the ACL access groups given to the user, i.e. make the association of the user and the access groups known to the ACL system. These include up to 16 groups which the user has regardless of project, plus up to 16 more which the project administrator has assigned.

12.3.3.2 Initial Attach Point

The system will set up a user's initial attach point when he logs in. After that, the user's home attach point and current attach point are also set to the initial attach point. The home and current attach points are also set to the initial attach point whenever a ring three reinitialization occurs.

12.3.4 External Login/Logout

Currently Primos supports the notion of an "external login program," a SAVE file called "LOGIN" which resides in CMDNC0 and is invoked whenever a user logs in or out. In order to provide increased flexibility, a second program, called "LOGOUT" and to be run when the user logs out may now be placed in CMDNC0. "LOGIN" will still be run at login time. For compatibility purposes at revision 19, when a user logs out the system will first search CMDNC0 for a "LOGOUT" program, and if none is found will then search for a "LOGIN" program. After revision 19 the system will search only for "LOGOUT" on logouts.

12.3.5 Initial Process

Each user may supply an "initial process" which is invoked by Primos when the user logs in. Once the validation and setup has been completed, and after the external login program (if any) has been run, the system will search the user's initial attach point for a file with the name "LOGIN.@", where "@" may be any of "CPL", "COMI", or "SAVE" (the suffices are searched in that order, and the first one found will be invoked). This file will also be invoked whenever the user's ring three environment is reinitialized, such as after a "DELSEG 6002" command. For this reason, if the file found is "LOGIN.CPL," the CPL program will be passed one argument by Primos: "LOGIN" if the program is being run during a normal login sequence, or "ERROR" if it is being run as the result of a ring three reinitialization. Note that the initial process is not executed for phantoms.

12.4 Support of non-ACL Systems

For compatibility purposes, minimal support of user registration is provided at revision 19 for non-ACL systems. If the profile editor discovers during initialization that the MFD is not an ACL directory, it will allow the user the opportunity to convert the MFD. If this option is not chosen, a non-ACL SAD will be created.

12.4.1 Restrictions on non-ACL Systems

Since multiple-level protection of the SAD databases is not feasible without ACLs, non-ACL systems may not create projects other than the system default project ("DEFAULT"), and its project administrator must be the system administrator. No protection other than system default settings is provided for the SAD.

12.4.2 The Profile Editor without ACLs

If a non-ACL SAD is created, all references to groups and ACLs in the section on EDIT_PROFILE should be ignored.

12.5 New and Changed Commands

A number of new commands have been added to Primos to enable the user to utilize the features of User Profiles, and the LOGIN command has been changed. These commands are listed below.

12.5.1 LOGIN

The new login command format is as follows:

```
LOGIN  [<user_id> [<password>] [-ON <system_name>]]
       [-PROJECT <project_id>]
```

Any arguments may be omitted on the command line, but the <password> and -ON option may only be given if the <user_id> is, and any arguments which are required will be requested by the system. For passwords, the system will turn off echoing (to project the password) before requesting it. The -ON option must be explicitly given in order for the user to effect a remote login.

At the system administrator's option, entering passwords on the login line may be disallowed, thus forcing all users to enter their passwords with echoing turned off. In this case, the syntax of the LOGIN command becomes:

```
LOGIN [<user_id> [-ON <system_name>]] [-PROJECT <project_id>]
```

This option is controlled by a command to the profile editor, described in detail later.

12.5.2 LIST_GROUP (LG)

The LIST_GROUP command, which takes no arguments, is used to list the ACL access groups the user is associated with. The command does not take any arguments. A list of group names (32 maximum) will be printed.

12.5.3 ORIGIN (OR)

The ORIGIN command changes the home and current attach point of the user to the initial attach point. This command takes no arguments.

12.5.4 CHANGE_PASSWORD (CPW)

The CHANGE_PASSWORD command allows the user to change his login password. Its format is:

```
CHANGE_PASSWORD <old_password>
```

The system will ask the user to enter the new password. After that, the system will ask the user to enter the new password a second time for verification. If the new password entered the first time is different from the new password entered the second time, the system will print an error message indicating that the passwords do not match and the old password is left unchanged. This is to make sure that the user does type in the new password he intended to use instead of a different password as a result of typographical or transmission errors. The entire password interrogation process is done in a half duplex mode so that characters are not echoed back on to the terminal screen or paper. If the <old_password> is incorrect, an error messsage is printed and the password is not changed. If the system administrator disallows null passwords, CHANGE_PASSWORD will not allow a user to make his password null.

12.5.5 ADD_REMOTE_ID (ARID)

The ADD_REMOTE_ID command allows a user to set up identities which are to be used by NPX slaves working on his behalf on systems outside his common naming sphere. It takes the same arguments as the LOGIN command, i.e.:

```
ADD_REMOTE_ID <user_id> [<password>] -ON <system_name>
              [-PROJECT <project_id>]
```

A <user_id> and <system_name> must be supplied. If no <password> is given, the <user_id> on system <system_name> must have a null password, or the user's slave will not be allowed to communicate with that system. Similarly, if no <project_id> is given, a default login project must be associated with the <user_id> on system <system_name>. The command will not request missing options. If the <system_name> matches a name already in the user's list of systems outside the common naming sphere, the new ID will replace the existing one. There is currently a limit of 16 different nodes in this list.

12.5.6 LIST_REMOTE_ID (LRID)

The LIST_REMOTE_ID command will display the ID's set up by previous ADD_REMOTE_ID commands. Its syntax is:

```
LIST_REMOTE_ID [-ON <system_name>]
```

If the -ON option is given, only the ID for system <system_name> will be listed; otherwise the entire list of up to 16 system/ID sets will be displayed. Passwords are never displayed.

12.6 Profile Editor (EDIT_PROFILE)

In order to facilitate the work of the system administrator in setting up the user profile data bases, the EDIT_PROFILE external command is provided. This program provides the system and project administrators with a simple yet powerful tool for manipulating the user profile data bases. It may be used to add, change, and delete information about users, projects, and all their respective attributes in a conversational manner. The EDIT_PROFILE command may be issued by any user, but since the data bases are protected by ACLs, only authorized administrators may access the data, and EDIT_PROFILE in fact checks access to the data files at initialization and rejects users who are not privileged.

12.6.1 Invoking the Profile Editor

The Profile Editor is invoked by issuing the EDIT_PROFILE external command. The format of this command is:

```
EDIT_PROFILE [<parent_tree>] [-PROJect <project_id>]
```

If the <parent_tree> is not given, the MFD on ldev zero is assumed. Otherwise, <parent_tree> is a pathname describing where in the file system the SAD to be edited exists. This feature is provided for two basic reasons: 1) So that a system administrator with jurisdiction over many nodes of a network may edit any SAD from one terminal, and 2) to allow administrators or developers to create a SAD on a disk on one system, and then move the disk to another system. The <parent_tree> pathname is the name of the parent directory (at the top level, an MFD) of the SAD to be edited. If not an MFD, this directory must be an ACL directory for which the user has Protect and Add permissions. If the -PROJECT option is given on the command line, the <project_id> specified will become the "current project" as defined in the attach_project command, below.

12.6.2 Profile Editor Modes of Operation

EDIT_PROFILE has three basic modes of operation, depending on who is using it and when it is being used. These three modes are:

12.6.2.1 System Initialization Mode

This mode is used when Primos first comes up and no SAD is present on the command device. EDIT_PROFILE can be invoked by the System Administrator in order to create the SAD from scratch. It is at this time that the SA sets the size of the user validation file (UVF) in order to accomodate some number of potential users, creates his own entry in the UVF, and sets up the SAD. All these functions are handled by EDIT_PROFILE.

12.6.2.2 System Administrator Mode

This is the normal mode of operation for EDIT_PROFILE. When in SA mode, the System Administrator may change the validation information for any user or project on the system, or examine any of this information.

12.6.2.3 Project Administrator Mode

PA mode is a simple subset of SA mode, restricted only by the limitations placed on the ability of Project Administrators to access certain files. All information except that in the UVF is available for examination, but only those files which specifically allow write access to the PA may be changed.

12.6.3 EDIT_PROFILE in its various modes

This section describes EDIT_PROFILE from a user's point of view in each of the three above-mentioned modes (Initialization, System Administrator, and Project Administrator).

12.6.3.1 Format of the descriptions

In the following descriptions, these conventions apply: 1) Abbreviations for options and commands are denoted by capitalizing the letters of the abreviation. For example, if an option is shown as "DeFauLT," then its abbreviation is "DFLT." 2) Things enclosed by angle brackets ("<" and ">") are descriptive names which are to be substituted with the real name of the object when EDIT_PROFILE is actually run. 3) Objects enclosed in brackets ("[" and "]") are optional. The command description will usually indicate whether or not EDIT_PROFILE will prompt for these optional arguments.

12.6.3.2 Initialization mode

When EDIT_PROFILE is run when no SAD exists, it is considered to be in initialization mode. Since there is no SAD, no users will be able to log into the system, and therefore EDIT_PROFILE must be run from the system console. This description assumes that this is true.

12.6.3.2.1 What initialization mode does

EDIT_PROFILE's initialization mode sets up the SAD, and its immediate contents: the UVF, MGF, and MPF. It also creates an entry for the system administrator in the UVF and, optionally, the system default project. The user has the option of choosing groups associated with each user, each project, or both. If desired, the system default project will also be set up. Once this is done the system administrator can log into the system and use EDIT_PROFILE in system administrator mode.

12.6.3.2.2 EDIT_PROFILE in initialization mode

When EDIT_PROFILE runs in initialization mode, it will ask the user various questions which tell EDIT_PROFILE what the SAD should look like. These questions and descriptions of them appear below.

12.6.3.2.3 Creating the SAD

EDIT_PROFILE gives you the option of not creating the SAD when it discovers that it does not exist. If you answer "NO" to the question "SAD does not exist. Create it?", EDIT_PROFILE will immediately be terminated. Otherwise, EDIT_PROFILE will attempt to create the SAD. If the SAD's parent directory is already an ACL directory, the SAD and its protecting ACLs will be created. If the parent is not an ACL directory (allowable only when the parent is the MFD), EDIT_PROFILE will give you the opportunity to convert it to an ACL directory. If you choose to leave the MFD password protected, certain restrictions, described above, apply.

12.6.3.2.4 Determining where groups will be set

ACL groups may be associated with a user regardless of his current project, because of his current project, or both. If groups are disabled at any level, all questions which EDIT_PROFILE might otherwise ask about groups at that level will be supressed. Initialization mode gives you the opportunity to set up groups only at the level or levels which you desire. These initially selected levels may later be changed by commands which are available in System Administrator mode.

12.6.3.2.5 Selecting a size for the UVF

The next question asked is "*** Creating user validation file. Projected number of users?". This question should be answered with the number of users that are currently on the system or that you expect to be on the system in the near future. (That is, the number of "loginable UFD's" on the system.) If you specify zero, the default minumum size of 20 will be selected automatically by EDIT_PROFILE. Note that EDIT_PROFILE will choose the actual size of the UVF based on the size you give, and that the chosen size will always be larger than the specified size, possibly much larger. This is because EDIT_PROFILE has a list of allowable file sizes which is designed to minimize search time for user id's.

12.6.3.2.6 Defining the entry for the system administrator

One of EDIT_PROFILE's most important tasks in initialization is to set up an entry for the system administrator. At this point you will be asked for the name (user id) of the system administrator. His attributes will be set later.

12.6.3.2.7 Creating the System Default project

After the system administrator's name has been supplied, EDIT_PROFILE gives you the chance to set up the system default project, called "DEFAULT." If you plan on having only one project on the system, or allowing a default project to "catch" users who currently belong to no specific project, you should answer YES to the question 'Create project "DEFAULT"?'. As long as DEFAULT is the only project on the system, users will automatically be both added to the system (registered) and added to project DEFAULT. If you answer YES to this question, the system administrator will automatically be made the project administrator of project DEFAULT, and you will be queried in order to set up his entry.

12.6.3.2.8 Adding the system administrator to a project

In order to be able to log into the system, each user must belong to some project. If you do not choose to set up DEFAULT as described above, the SA must be added to some other project. If the SA is to be the project administrator for that project, his entry may be created by answering YES to the question "Create administrator entry?" or by giving the -create_pa option to the add_project command. If DEFAULT is created as described above, it will automatically be made the SA's default login project.

12.6.3.2.9 Protecting the databases

EDIT_PROFILE automatically creates ACLs which protect the validation files and project directories to allow access only to the system administrator. When projects are added later, entries for the various project administrators will be created as the projects are added, again automatically by EDIT_PROFILE. During initialization mode, EDIT_PROFILE also grants full access to user SYSTEM, since initialization must be done at the system console.

12.6.3.3 An Example of EDIT_PROFILE in Initialization Mode

In the following example, user input is underlined.

```
Profile editor [rev 19.0] in initialization mode 13 Jan 81 14:55:25.
SAD does not exist, create it? YES          /* Set up a new system
Do you want SYSTEM-wide groups, PROJECT-based groups, or BOTH? BOTH
*** Creating User Validation File.  Projected size? 0  /* Use default
System administrator name? MY_NAME          /* System Administrator is me

Create project "DEFAULT"? YES               /* Someplace for everyone

Set system-wide attributes for user "MY_NAME":
    Password: <my_password>                 /* SA should have password!
    Groups: .ADMINISTRATORS
*** New group added to system: .ADMINISTRATORS
    Default project: DEFAULT
*** New project added to system: DEFAULT

*** User Validation File created 13 Jan 81 14:56:00.
    20 entries in prime area; file is 1 records long.

*** Master Project File created 13 Jan 81 14:56:04.

*** Master Group File created 13 Jan 81 14:56:04.

Set limits for project "DEFAULT":
    Groups: .DEFAULT .OPSYS                 /* Two groups to start
*** New group added to project: .DEFAULT
*** New group added to project: .OPSYS

Set attributes for user "MY_NAME" in project "DEFAULT":
    Groups: .OPSYS
    Initial attach point: <MY_DSK>MY_DIRECTORY /* Fully qualified

Set profile attributes for project "DEFAULT": /* Set project profile
    Groups: <cr>                            /* No groups by default
    Initial attach point: <MY_DSK>MFD       /* Default to top
Project "DEFAULT" created.
    20 entries in prime area; file is 1 records long.
Check entry? NO                             /* We're reasonably sure
> LU MY_NAME -PROJ                          /* Now look at our handiwork
```

```
*************************************************************************
System-wide attributes for user "MY_NAME":

    Groups: .ADMINISTRATORS
    Default login project: DEFAULT

Attributes for user "MY_NAME" in project "DEFAULT":

    Groups: .OPSYS
    Initial attach point: <MY_DSK>MY_DIRECTORY
*************************************************************************
> Q                                        /* SA may now log in
```

12.6.3.4 After the SAD is set up

After setup of the SAD is complete, initialization mode basically becomes system administrator mode. Unless user SYSTEM has been set up as the system administrator, however, once the initialization mode session of EDIT_PROFILE has been terminated, EDIT_PROFILE may not be run from the system console.

12.6.4 System Administrator mode

The normal mode of operation for EDIT_PROFILE is system administrator mode. All commands to add, change, and delete attributes of users and projects are available to the user. These commands are described below.

12.6.4.1 Add project

The add_project command (abbreviated AP) creates a new project directory in the SAD and sets up all the files associated with it according to the options given in the command. The format of the add_project command is:

```
Add_Project [<project_id> [-PA <pa_name>] [-CReate_pa]
            [-SIZE <entry_count>] [-No_Query]
            [-LIKE <like_reference>] [-PROFile]]
```

<project_id> is the name of the project to be created. It must be given if any of the other options are.

The -LIKE option allows you to create a new project with the same attributes that the project named by <like_reference> has.

The -PA option defines the name of the project administrator for the new project. If it is not given, or the <pa_name> is null, EDIT_PROFILE will request the <pa_name>.

The -CREATE_PA option specifies that you wish to create an entry in the new project for the project administrator specified by the -PA option. If not given, you will be asked whether or not you want to create the administrator's entry. Note that the project administrator need not belong to the project or projects which he administers.

The -PROFILE option specifies that a project profile is to be set up at this time. If not given, the project profile will default to null entries.

The -SIZE option allows you to specify the size of the new project, that is, the number of users that the validation files are designed to handle. If not given, the project is created with the default size of 20. Note that if at any time while you are adding users to the system or a project EDIT_PROFILE detects that the validation files are overloaded, it will give you the opportunity to rebuild the files to accomodate more entries.

The -NO_QUERY option supresses the "Check" and "Change" questions after the project is created.

12.6.4.2 Add_user

The add_user (AU) command adds the specified user to the project and/or system data bases and creates the user profile.

```
Add_User [<user_id> [-LIKE <like_reference>] [-NoQuery]
         [-PassWord [<password>]] [-PROFile] [-SYStem]
         [{-PROJect | -DeFauLT} [<project_id>]]
         [-Verify_NS]
```

The user is added only to the system unless the -PROJECT or -DEFAULT option is given, there is only one (default) project on the system, or no options are given and there is a current project. The user can be added just to a project if none of the -PASSWORD, -DEFAULT, or -SYSTEM options is used but -PROJECT is. If the -PROJECT or -DEFAULT option is given without a <project_id>, the current project will be used if one exists, otherwise the <project_id> will be asked for.

The -LIKE option allows you to specify some other, already existing, user whose attributes the new user is to assume. If the -PROJECT or -DEFAULT options is being used, the user named in the <like_reference> must belong to the same project as the new user.

The -PASSWORD option supplies the login password for the new user. If this option is not given or is given with no password, the password will be requested. The -PASSWORD option implies the -SYSTEM option, below.

The -PROJECT option specifies the project to which this user is to be added. A user may belong to more that one project, but may be added to only one project at a time. Use of this option does not affect the user's default login project.

The -DEFAULT option is the same as the -PROJECT option, but also makes <project_id> the user's default login project. If the -DEFAULT option is not specified when a user is initially entered into the system, the default login project will be asked for. The -SYSTEM option, below, is implied by the -DEFAULT option.

If the -PROFILE option is given, you will be queried in order to set up a user profile. Otherwise, the user's attributes will be taken from the project profile.

The -SYSTEM option specifies that the user is to be added at the system level. This is the default option in system administrator mode. The -SYSTEM option implies both the -PASSWORD and -DEFAULT options.

The -VERIFY_NS option searches the SADs of all systems within the common naming sphere for a user id identical to the one being added, and if one is found a warning is printed. This option is designed to make administration of networked systems easier. See also the Verify_user command, below.

The -NO_QUERY option supresses the "Check" and "Change" questions after the user is added to each validation file.

12.6.4.3 Attach_project

The attach_project (ATP) command sets the "current project" for all subsequent operations on users. Operations on other projects are not affected. However, if a current project exists its <project_id> will be substituted whenever a <project_id> is called for but omitted.

ATtach_Project [<project_id>]

<project_id> is the name of the project to which to attach. If not specified it will be asked for. The <project_id> must name a valid project. Note that if there is only one project on the system, it becomes the current project automatically when the profile editor is invoked.

12.6.4.4 Change_project

The change_project (CP) command changes the attributes or reserved space in some project on the system.

```
Change_Project [<project_id> [-PROFile] [-LIMits] [-LIST]
                [-PA [<pa_name>]] [-SIZE [<entry_count>]]]
```

<project_id> is the name of the project whose entry is to be changed.

The -PROFILE option specifies that the profile is what is to change.

The -LIMITS option specifies that the master project limits (MPP contents) are to change.

The -PA option specifies that the name of the project administrator is to change, and optionally gives the <pa_name> of the new PA.

The -SIZE option specifies that the <entry_count> reserved in the PVF is to be changed. If <entry_count> is not given, it will be requested. The -SIZE option must be explicitly given in order to change the <entry_count> in the PVF; if it is not supplied the user will not be asked for a new size.

Note: The -SIZE option should be used as infrequently as possible, especially for large projects, as the entire PVF must be rebuilt whenever it is used. However, using the -SIZE option to decrease the space reserved in the PVF can be a useful tool in conserving disk space. The REBUILD command may also be used to change the <entry_count> in a project, and is the preferred way to do this if other attributes of the project are not to be changed.

The -LIST option will display the project attributes after the change_project command has been executed.

12.6.4.5 Change_system_administrator

The change_system_administrator (CSA) command allows you to change the name of the system administrator. This command should be used only when absolutely necessary. Its syntax is:

```
Change_System_Administrator [<new_sa_name>]
```

If you do not supply a <new_sa_name>, one will be requested. After that, EDIT_PROFILE will ask you if you are really sure you want to change the system administrator. If you answer positively, all the ACLs protecting the SAD and the project directories will be changed, and the profile editor session will be terminated.

Note that the CSA command rebuilds all ACLs from scratch, so that if you have hand-altered any ACLs, those changes will be lost.

12.6.4.6 Change_user

The change_user (CU) command allows you to change the system-wide and/or project-based attributes of any user on the system.

```
Change_User [<user_id> [-PassWord [<password>]]
            [-PROJect <project_id>] [-SYStem] [-LIST]]
```

<user_id> is the name of the user whose attributes are to change. If given, the <user_ic> must be the first argument to the command.

The -PASSWORD option specifies the new password for the user. If the <password> is not supplied but the -PASSWORD option is given, you will be asked for the <password>.

The -PROJECT option specifies that project-specific attributes are to be changed, and optionally defines the project in which to change them. If no <project_id> is given and there is no current project (see the ATTACH_PROJECT command above), a project will be asked for.

The -SYSTEM option specifies that the user's system-wide attributes (system-wide groups and default login project) are to be changed. Since password changing by the system administrator should be rare, the password will only be requested if the -PASSWORD option is given.

12.6.4.7 Delete_project

The delete_project (DP) command deletes the specified project directory from the SAD. If any users have this project as their default login project, they will be reset to have no default login project. If any users are in the project when the DP command is given, you will be given the option of not deleting the project.

```
Delete_Project [<project_id>]
```

<project_id> is the name of the project directory to be removed. If it is not specified and there is a current project, the current project will be deleted and the user detached from that project. If no <project_id> is given when there is no current project, the <project_id> will be requested.

12.6.4.8 Delete_user

The delete_user (DU) command removes the specified user from the system and any projects to which she belongs, or just from one specified project.

Delete_User [<user_id> [-PROJECT [<project_id>]]

<user_id> is the name of the user whose entries are to be removed. If specified it must be the first argument to the command. Otherwise, the <user_id> will be requested.

If the -PROJECT option is given, the user will be deleted only from the specified project. Otherwise the user will be removed from the system as well as any projects to which she belongs. If the -PROJECT option is given without a <project_id>, the current project will be used, or a <project_id> will be requested if there is no current project.

12.6.4.9 Detach_project

The detach_project (DTP) command clears the current project, which was set by some previous attach_project command. If there is no current project, this command has no effect.

DeTach_Project [<project_id>]

<project_id> is the name of the project to which EDIT_PROFILE is currently attached. It is completely optional since there can be only one current project; however, if a <project_id> is given it must match the name of the current project.

12.6.4.10 Force_password

The Force_password (FPW) command allows you to specify that users should not be allowed to type their passwords on the login line, thus forcing passwords to be entered at the system's request with echoing turned off. This feature is provided for security-conscious installations which are concerned about compromise of passwords. Its format is:

Force_PassWord [{-ON | -OFF}]

The -ON option enables this feature, while the -OFF option disables it. If neither -ON or -OFF is given, -ON is assumed.

Another useful command for security-coonscious installations is the no_null_password command. See below for details.

12.6.4.11 Help

The help command provides you with information on how to use EDIT_PROFILE.

```
HELP [<command_name>]
```

If no <command_name> is given, the help command prints a list of all legal commands to EDIT_PROFILE, the argument, if any, which the command takes, and the options to the command. If a <command_name> is given, the command's format (including the main argument, all options, and all arguments to options) is displayed.

12.6.4.12 Initialize

The Initialize command is used by CONVERT_PROFILE to create the SAD as efficiently as possible. It is not normally used by the system administrator, and may be used only in initialization mode. It is documented here for completeness.

The syntax of the Initialize command is:

```
INITialize <treename>
```

<treename> is the name of the file containing the initialization data. This data is described by lines of the form:

```
<user_id> <initial_attach_point> [-PW <password>]
```

All users are put into project DEFAULT, and no groups are set. Duplicate user IDs are skipped, and all errors in input lines cause those lines to be ignored.

12.6.4.13 List_project

The list_project (LP) command lists the attributes of the specified project, as well as those of either one or all users in the project. The project limits (from the MPP) are always listed by this command. Other attributes which may be listed are controlled by the options.

```
List_Project [<project_id> [-PROFile] [-USER <user_id>]
              [-ALL] [-OUTput <filename>] [-APPend] [-TTY]]
```

The -PROFILE option requests that the project profile be listed in addition to the MPP contents.

The -USER option specifies that the profile for user <user_id> is to be listed. To list a user's data only, see the list_user command.

The -ALL option specifies that all user profiles should be listed.

Since the -ALL option may require a great deal of time and paper to list at the terminal, the -OUTPUT option allows you to direct the output from this command into a file. Currently, this file will always be created in the SAD, and is thus by default protected so that normal users may not read it. If the -APPEND option is given, output will be added to the end of the existing contents of <filename>; otherwise <filename> will contain only the output from this invocation of the list_project command.

The default when the -OUTPUT option is given is that output goes only to the named file. If the -TTY option is also given, output will be directed to the terminal as well.

12.6.4.14 List_system

The list_system (LS) command displays attributes of the system as well as for all users, groups, and projects in combinations selectable by the user.

```
List_System [-USers] [-GRoups] [-PROJects] [-DETail] [-ALL]
            [-SORT] [-OUTput <filename>] [-TTY] [-APPend]
```

The -USERS option specifies that the system-wide attributes of all users on the system should be listed.

The -GROUPS option specifies that all groups currently on the system should be listed.

The -PROJECTS option specifies that all projects currently on the system should be listed along with their attributes.

The -ALL option combines the features of the above three options.

If the -DETAIL option is given, it affects the other options in the following way: The -USERS list will contain the list of projects to which each user belongs. The -GROUPS list will show which users and projects have which groups as attributes. The -PROJECTS list will show which users and groups belong to each project.

The -OUTPUT, -TTY, and -APPEND options work just as they do for the list_project command, above.

The -SORT option specifies that all entities in the listing should be sorted alphabetically. This includes <group_name>, <project_id>, and <user_id>. This option has not yet been implemented.

If no options are given, the system administrator name and system attributes will be displayed. System attributes may include the following:

```
SAD not ACL protected
System-wide groups enabled (ACL systems only)
Project-based groups enabled (ACL systems only)
Non-DEFAULT projects in use (ACL systems only)
Passwords always requested at login (Force_password command)
Null passwords not allowed (No_null_password command)
```

12.6.4.15 List_user

The list_user (LU) command displays attributes for the named user.

```
List_User [<user_id> {-PROJECT [<project_id>] | -ALL}]
```

<user_id> is the name of the user whose attributes are to be shown. If specified it must be the first argument to the list_user command. If not supplied it will be requested.

The -PROJECT option specifies the <project_id> of the project from which the user's attributes are to be read. If no <project_id> is given and there is no current project, a <project_id> will be requested; otherwise the current project will be used.

The -ALL option requests EDIT_PROFILE to check all projects in the SAD for this user's <user_id> and to display the attributes associated with each project to which the user belongs.

12.6.4.16 No_null_password

The no_null_password (NNPW) command forces all passwords entered by the system administrator or changed by users through use of the Primos change_password command to be non-null. Its format is:

```
No_Null_PassWord [{-ON | -OFF}]
```

If neither option is given, -ON is assumed. When this command is given (with no arguments or the -ON option), any users who currently have null passwords will be listed so that the system administrator may assign them passwords. This feature is provided to enable security-minded installations to be sure that user id's may not be compromised by simple trial and error methods. Security-conscious installations may also be interested in the force_password command, above.

12.6.4.17 Rebuild

The rebuild (RFB) command reconstructs the UVF or some specified PVF to hold a new number of users. At this time the associated data files are also reconstructed, and any "dead" entries are removed.

```
REBuild [-PROJECT [<project_id>]] [-SIZE <entry_count>]
```

The -PROJECT option specifies that a PVF is to be rebuilt. If no <project_id> is given and there is no current project, the <project_id> will be requested. If the -PROJECT option is not used, the UVF will be rebuilt.

The -SIZE option specifies the new minimum size of the appropriate validation file. If not given, EDIT_PROFILE will automatically resize the file according to the number of entries currently in use.

Because the rebuilding process necessarily involves destruction and reconstruction of all files in the SAD or a project directory, the rebuild command should never be used when users may be logging into the system. It is strongly recommended that the rebuild command be used only when a MAXUSR 0 is in effect.

Before any changes are made to the files, backup copies are made so that in case of problems you can replace the old files and return to the status quo. For the UVF, these files are called <filename>.OLD, e.g. UVF.OLD. For PVFs, the files are placed into the BACKUP directory of the appropriate project directory, e.g. SAD>DEFAULT>BACKUP>PVF.

12.6.4.18 Set_default_protection

The Set_default_protection (SDPR) command is used to restore the ACL protection in the SAD to its default state. On both password systems, it will also make sure the read/write lock settings on all files are correct. It may also be used to convert from a password to an ACL SAD. Its syntax is:

```
Set_Default_PRotection [-CoNVert]
```

The -CONVERT option is specifiec if a password SAD is being converted to ACLs.

12.6.4.19 Set_project_groups

The set_project_groups (SPG) command allows you to enable or disable groups at the project level. Set_project_groups takes either an -ON or -OFF option; if neither is given, -ON is assumed. Note that when groups are turned off by this command, all that is actually done is to indicate to the system that project groups are not to be set. No groups are "removed" from projects or users. Groups will be "cleaned up" when the next rebuild is done.

12.6.4.20 Set_system_groups

The set_system_groups (SSG) command enables or disables groups at the system level. It works analagously to the set_project_groups command.

12.6.4.21 Verify_user

The Verify_user (VU) command allows administrators of networked systems to determine on which systems within the common naming sphere a given user ID exists. Its format is:

```
Verify_User {<user_id> | -ALL}
```

If a <user_id> is given, all SADs in the naming sphere are searched for the given ID and a list of systems on which it is found is printed. If the -ALL option is given, all users in the SAD will be checked against all other SADs.

12.6.5 Project Administrator mode

When a project administrator invokes EDIT_PROFILE, he may execute basically the same commands as the system administrator, but in a more restricted environment. Also, when EDIT_PROFILE runs in project administrator mode, it asks the user for a <project_id> immediately upon startup. This <project_id> then becomes the current project. The commands available in project administrator mode and how they differ from the same commands in system administrator mode are described below.

12.6.5.1 Add_user

The add_user command adds users only to projects. Therefore, the -SYSTEM, -DEFAULT, and -PASSWORD options are not available. Since a project administrator will usually have a current project, the command is generally used only with a <user_id> and the -PROFILE option.

12.6.5.2 Attach_project

This command works exactly as it does in system administrator mode.

12.6.5.3 Change_project

The change_project command may be used only to change the project profile or rebuild the PVF.

12.6.5.4 Change_user

The change_user command may change only the project-based data for a user. Furthermore, the restrictions imposed by the MPP apply when setting these attributes.

12.6.5.5 Delete_user

Users may be deleted only from projects, specifically, only those which the project administrator administers.

12.6.5.6 Detach_project

This command is available in project administrator mode solely to allow administrators who administer more than one project to avoid confusion by having no current project. If you administer only one project, avoid use of this command.

12.6.5.7 Help

The HELP command works as it does in system administrator mode, but only lists those commands and options which may be used in project administrator mode.

12.6.5.8 List_project

This command works exactly as it does in system administrator mode.

12.6.5.9 List_user

This command works exactly as it does in system administrator mode, but system-wide attributes will not be displayed and users' attributes will be displayed only for projects which you administer.

12.6.5.10 Rebuild

The rebuild command may only be used for projects which you administer.

12.6.6 Restrictions

At the initial release of EDIT_PROFILE and the User Profiles system, the following restrictions apply to the system:

12.6.6.1 Number of users supported

At rev. 19 of Primos, the UVF is designed to hold up to approximately 5000 users with optimum efficiency. There is no real restriction on the actual number of users which may be in the UVF, but if more than 5000 are present, decreased efficiency may result.

The largest table size built into EDIT_PROFILE is 7516. Once the hash table reaches this size, overflow messages will not be printed by the Add_user command, since a larger table may not be built anyay.

12.6.6.2 Number of groups supported

A maximum of 4096 different groups may exist in the MGF.

12.6.6.3 Number of projects supported

Because of restrictions in the ACL system and the way projects are protected, an absolute maximum of 32 different project administrators may exist on any system without changing the way files in the SAD are protected. The number of projects is essentially unbounded (4096 may theoretically be supported), but unless the ACLs in the SAD are hand-modified by the system administrator, realistically only about 25 different project administrators may exist.

12.6.6.4 Number of users supported in each project

A PVF has the same size restriction that the UVF does, that is, approximately 5000 users without performance penalties. If a reasonable amount of data overlap between users exists, this number should be supportable without difficulty. If each user has a unique initial attach point pathname and/or list of groups, however, the PDF may become full when approximately 2000 users have been added to the project. We anticipate that data overlap, use of the project profile for default values, and relatively small project size will prevent this problem from occurring; if it does become a problem, though, the size of the PDF may be increased at future revisions.

Table of Contents

Subject: PRIMOS2

Release: 19.0

Date: February 5, 1982

1 New Functionality

None.

2 User Visable Bug Fixes

Model 3 1600/6250 BPI Capable Magtape Drive Operational Note:

The tape density on a model 3 1600bpi/6250bpi capable magtape drive can be set via software or via the density select switch on the front panel of drive, but not both at the same time. When the density is set via a software call to T$MT or an ASSIGN command (which also calls T$MT), the front panel switch is disabled from having any further effect on the tape density. The front panel switch can only be re-enabled by another call to T$MT that enables the switch (see T$MT changes below).

Under PRIMOS2 (DOS), these drives are initialized to enable the front panel density select switch. Under PRIMOS4, density setting is handled at magtape at ASSIGN time. If PRIMOS4 crashes and a tape dump is to be taken, the act of hitting the "MASTER CLEAR" switch causes these drives to revert to their initial state of 1600 bpi with the front panel switch disabled and the dump is therefore taken at this density.

3 Internal Fixes, Not User Visable

None.

4 Outstanding Problems

None.

ABSTRACT

At REV 19, Primos may be coldstarted using a procedure that takes the system from depressing the start switch to Primos IV in one step. This procedure uses additional front panel switch settings (switches 4 and 5) and a new command in CMDNC0 (Primos).

1. Introduction

At REV 18, three software systems are used during coldstart. They are Boot, Primos II and Primos IV. At REV 19 a fourth system has been introduced, the Primos command. It is installed in CMDNC0 and is instrumental in simplifying the coldstart procedure. Subsequent sections of this document specify in detail the software required and procedures to be followed to perform a simplified coldstart at REV 19.

2. Software Required

2.1 Boot - must be from a REV 19 Master Disk or created by a REV 19 MAKE.

2.2 Primos II - must be REV 19 dated 11/18/80 or later. Must be installed in DOS>*DOS64.

2.3 Primos command installed in CMDNC0.

2.4 Primos IV installed in a directory on the partition to be coldstarted.

3. Use of Front Panel Switches 4 and 5

3.1 Switch 4 down, switch 5 down. No change from REV 18 procedure.

3.2 Switch 4 up, switch 5 down. Do not prompt for 'Physical Device ='.

3.2.1 Front panet switches are interrogated by software and the device is automatically started up. For example, if coldstarting from physical device 460, switch setting 10114 will startup device 460, 1060, etc. Switch setting 10134 will start up device 660, 1260, etc. Note this may be used only with the top partition on a device.

3.2.2 When the system prompts 'OK:', it is running Primos II. At this point the Primos command is used to bring up Primos IV. The command is issued as PRIMOS <pathname>, where <pathname> is the pathname of the directory containing the run files for Primos IV. The Primos command remembers the pathname so the next time typing just PRIMOS is sufficient. Initially the pathname defaults to PRIRUN.

3.3 Switch 4 up, Switch 5 up.

Physical device is automatically started up from front panel switch setting and Primos IV is automatically brought up from the pathname saved in the Primos command.

For example, switch setting 14114 will bring up Primos IV on device 460, (or 1060, etc.) and switch setting 14134 will bring up Primos IV on device 660. Note this may be used only with the top partition on a device.

4. Example of Coldstart using Device 460

Assume Primos IV runfiles are in a directory called OPSYS.

- o Power on.
- o Turn rotary selector to Stop/Step
- o Master clear.
- o Turn Address/Data switch to Address.
- o Set '10114 in the sense switches (switches 4, 10, 13, 14 up).
- o Turn selector to load.
- o Press Start.
- o Turn selector to Run.
- o Type PRIMOS OPSYS on the system console.

To reboot the system:

- o Turn rotary selector to Stop/Step
- o Master Clear
- o Turn Address/Data switch to Address.
- o Set '14114 in the sense switches (switches 4, 5, 10, 13, 14 up).
- o Turn selector to Load.
- o Press Start.
- o Turn selector to Run.

* NOTE: All products have been modified to conform to master disk standards. For a description of these modifications, please read INF019>STANDARDS.RUNO.

Subject: PROTECT

Release: 1.0

Date: 8/19/82

1 New Functionality

New command.

2 Problems Fixed

None.

3 Outstanding Problems

None.

4 Environment

Needs PRIMOS 19.0.65 or greater. (19.0.respin)

5 Installation and Build Procedures

Needs fsulib.build.cpl to be run first. Other than that it is standard.

6 Replacement

This command replaces the earlier PROTEC command. It differs from that command only in that it uses mnemonic codes rather than numeric keys to specify protection rights.

7 Usage

The usage information for the command follows.

Usage: PROTECT pathname [code1] [code2] [options]

pathname specifies the directory entry on which the protection rights are to be changed. Used in password directories, not in ACL directories.

code1 defines owner rights to the file.

code2 defines non-owner rights.

options may be selected in any order from the list below.

Codes may have the following values:

NIL	No access of any kind allowed
R	Read access only
W	Write access only
D	Delete only
RW	Read and write
RD	Read and delete
WD	Write and delete
RWD	Read, write, and delete (all rights)

Option descriptions:

-REPORT, -RPT
requests that the results of executing the command be reported to the user.

Note: The default protection codes associated with a newly created file in a password directory are RWD NIL (owner has all rights, non-owner has no rights). Defaults for the PROTECT command, however, are NIL NIL. Thus, if PROTECT is given without codes, neither owner nor non-owner has any access to the file in question.

Subject: PSD

Release: 19.0

Date: January 7, 1982

The FRAC instruction, which is obsolete at rev 19.0, has been removed from PSD's tables.

Subject: REVERT_PASSWORD

Release: 19.0

Date: January 21, 1982

New Functionality

The REVERT_PASSWORD command allows conversion of an ACL directory back to a a password directory. It operates only on the current directory, and takes no arguments. Before the command may be used, the directory must be cleared of any access categories or ACL subdirectories.

Further information on ACLs and REVERT_PASSWORD may be found in the section of the rev document dealing with ACLs.

Problems Fixed

N/A.

Outstanding Problems

N/A

Environment

REVERT_PASSWORD is an external command (residing in CMDNC0) which is useful and works only for PRIMOS revision 19.0 and later.

Build and Install Procedure

Standard.

Subject : RJE Phase 2 Emulator Common Code - RJESRC

Release : 1.0

Date : 04/07/82

1. New Functionality

This is a replacement product. The RJOP is the new operator interface to all the emulators. The RJQ is the new utility for queuing files, and replaces the previous SEND utilities.

Other changes in functionality include :-

A) Debug facilties have been improved,
B) The aoility for a single operator to control more than 1 emulator at a time has been added,
C) Files can be submitted from withir a user program.
D) Improved queuing facilities,
E) There is now better control of file transmittion

For a oescription of new functionality, please refer to IDR4036.

2. Problems Fixed

Emulator Name	RJESRC
Issue No.	1.0
TAR/POLER No.	37597
Changed by	Suzy
Date Fixed	09/21/81
User Visibility	None

Fix

The routine RJESRC>CARDSPOOL>I$BC03.PMA no longer has a missing quote in the declaration:

```
DATA    '275.LS.8.OR.'243
```

Emulator Name	RJESRC
Issue No.	1.0
TAR/POLER No.	28835
Changed by	Suzy
Date Fixed	04/19/82
User Visibility	There was no ability to translate lower-case characters within the CARDSPOOL routines.

Fix

There is now the facility to allow Cardspool to handle lower-case characters correctly.

Emulator Name	RJESRC
Issue No.	1.0
TAR/POLER No.	40494
Changed by	Pete
Date Fixed	04/19/82
User Visibility	The Work-Stations would appear to hang if called from within a CPL program.
Fix	The RJOP and RJQ can be run from within a CPL program.

3. Outstanding Problems

There are no known outstanding problems.

4. Environment

This release of the product requires PRIMOS 19.0.

5. Installation and Build Procedures

These are standard.

Subject: PWLOCK

Release: 1.0

Date: 8/19/82

1 New Functionality

New command.

2 Problems Fixed

None.

3 Outstanding Problems

None.

4 Environment

Needs PRIMOS 19.0.65 or greater. (19.0.respin)

5 Installation and Build Procedures

Needs fsulib.build.cpl to be run first. Other than that it is standard.

6 Usage

The usage information for the command follows.

Usage: RWLOCK pathname [lock] [options]

pathname specifies the directory entry on which the concurrency lock setting is to be changed.

lock read/write lock, specified as follows:

sys - use system read/write lock (default)
excl - N readers OR 1 writer (Exclusive OR)
updt - N readers AND 1 writer
none - N readers AND N writers

options may be selected in any order from the list below.

Option descriptions:

-REPORT, -RPT
requests that the results of executing the command be reported to the user.

SUBJECT: SEG

RELEASE: Rev19.0

DATE: August 19, 1982

1 New Functionality

- SEG now automatically loads SPLLIB whenever the Pure Fortran Library is loaded. The subcommands LI and PL do this. The result is that Rev19.0 SEG cannot be run on any system which does not have the SPL library.

2 Problems Fixed

POLAR NO.	Description
36524	SEG now reports an error if user loads a SEG file in the VLOAD subprocessor.

- SEG now deletes SEG files which were previously trashed for various reasons (e.g. user hit control-p in the middle linking session).

- If a user tries to use segment 4035, which is used by SEG internally for its own symbol table, a warning message will be given and flow of control will always return to SEG subcommand level. A warning error code will be returned at the end of the session. This is an improvement over revision 18.2 SEG which would always abort to PRIMOS under this circumstance, even if uninitialized data was the only code loaded into 4035.

3 CMDSEG

At rev 19.0 CMDSEG has been completly rewritten. The following enhancements/fixes have been made.

- The user DOES NOT have to be attached to the directory SEG.

- The resulting has a meaningfull name takeen from the name of the segment directory.

- CMDSEG is now written in CPL.

- CMDSEG and CMDSG1 have been combined into one program.

CMDSEG is invoked by:

- R SEG>CMDSEG <segment directory pathname> <run filename>

CMDSEG may also be invoked :

- R SEG>CMDSEG

CMDSEG will then prompt for the pathname of the segment directory.

4 Environment

- This version of SEG requires Rev19 Primos, Rev19 PFTNLIB, and rev19.0 SPLLIB.

5 Installation and Build Procedures

- Standard installation and build procedure.
- This version of SEG.BUILD.CPL will not operate correctly with rev18.2 SEG, but is compatible with all earlier versions of rev18 SEG.

Subject: SET_DELETE

Release: 1.0

Date: 8/19/82

1 New Functionality

New command.

2 Problems Fixed

None.

3 Outstanding Problems

None.

4 Environment

Needs PRIMOS 19.0.84 or greater. (19.0.respin)

5 Installation and Build Procedures

Standard.

6 Usage

The usage information for the command follows.

Usage: SET_DELETE pathname [options]

pathname specifies the directory entry on which the delete protection is to be modified.

options may be selected in any order from the list below.

Option descriptions:

-PROTECT, -PRO
requests that the object be delete protected. This is the default.

-NO_PROTECT, -NPRO
requests that the object to NOT be delete protected.

-REPORT, -RPT
requests that the results of executing the command be reported to the user.

Note: A user must have delete access (D) to the directory that contains the object in order to set the delete protect switch on a file, directory, or segment directory. (The switch cannot be used on access categories.)

ABSTRACT

SIZE has been substantially changed for revision 19.0. It reports file size in 1024-word records instead of 440-word records. It now is able to size segment directories, UFDs, and access categories in terms of the number of entries in them. It now prints the treename being sized on the output line so that use of the rev 19.0 command preprocessor's wildcard feature (e.g. "SIZE @@") produces identifiable results. Also, its output format is changed to make the output more readable.

SIZE is now in V-mode, but is still a static-mode program (not an EPF). Therefore, SIZE will not run under DOS. If it is needed to get the size of a file under DOS, FUTIL may be used (see the SCAN command).

Finally, the build procedure and source files now conform to the rev 19.0 master disk standard.

SIZE now reports file size in 1024-word records, not 440-word records. To get it to report in old-style 440-word records, do "SIZE <treename> -NORM".

SIZE can now size segment directories, UFDs, and access categories in terms of the number of entries in them. For segment directories and UFDs, the number of top-level entries is printed. In the case of UFDs, this number includes all top-level entries, i.e. files, segdirs, sub-ufds, ACLs, and access categories. For access categories, the number of entries in the category is printed. When sizing files, SIZE types "n records", but for other file-types (segdirs, ufds and access categories), it types "n entries".

SIZE now outputs the treename it was given on the size line. This is redundant if "SIZE <file>" is given and <file> is not wildcard-laden, but otherwise it is crucial to determine which SIZE output corresponds to which file. Since SIZE has no knowledge of wildcard processing, the command preprocessor for rev 19.0 PRIMOS will do the wildcard iteration for SIZE automatically.

Therefore, to size all file system objects in the current UFD, do "SIZE @@". To size all file system objects in or below the current UFD, do "SIZE *>@@>@@ -WALK_FROM 1".

To make the output more readable, and so that different file system object types may be discerned from each other readily, most of SIZE's output is now in lower-case. The exceptions are the keywords "SEGDIR" and "UFD", when a file system object of the corresponding type is sized. Also, the input treename will be output in upper-case (unless quoted by the user in lower-case).

For each file system object type, SIZE now also reports the file type in detail. For example, a "file" can be a "sam" or "dam" file, as can a segment directory. A UFD can now be an "acl" or "pwd" (password) UFD; and there is also the "access cat" (access category).

SIZE still reports the number of words in a file, but not in segment directories or ACLs, since that value is meaningless to the average user. It does report the size of segment directories in terms of the maximum number of entries it can hold; multiplying this by 2 yields it size in words. The size in words of a directory (acl or password) is also reported.

Note: since the command preprocessor does not tree-walk segment directories, "SIZE *>@@>@@" will not size any of the files inside a segment directory. For this capability, FUTIL should be used. However, doing "SIZE *>segdir_name>@@" will scan inside the segment directory, and SIZE will work correctly.

Example outputs of SIZE:

```
OK, SIZE MYUFD>@@
     3 entries in access cat "MYUFD>PROTECT-EM.ACAT"
     1 entry   in access cat "MYUFD>AB.ACAT"
    11 entries in acl UFD    "MYUFD>JUNK.UFD" (467 words)
    26 entries in acl UFD    "MYUFD>MAIL" (1058 words)
     1 entry   in acl UFD    "MYUFD>PRETENDER" (23 words)
     1 record  in sam file   "MYUFD>BBOARD_PROFILE" (26 words)
     2 records in sam file   "MYUFD>XSTAT.SAVE" (1034 words)
     4 records in dam file   "MYUFD>O_RUNIT" (3606 words)
     4 entries in sam SEGDIR "MYUFD>SWU.SEG" (65 total)
     1 entry   in dam SEGDIR "MYUFD>DATABASE" (5 total)
    33 entries in access cat "MYUFD>TEST.ACAT"
    24 entries in pwd UFD    "MYUFD>OLD_SOURCE" (5040 words)
     2 entries in access cat "MYUFD>MAIL_DIRECTORY.ACAT"
```

Details: when SIZE is going to print a number greater than 32767 in front of "entries" or "records", it will allow 12 full character positions for the number instead of 6, i.e.:

```
       32768 records in dam file   "MYUFD>OH_MY_GOD_ITS_BIG"
```

Also, if you manage to get SIZE to size a ufd via ".DEL..DEL.", i.e. giving it an entryname with the decimal value -1, it will correctly handle the outputting of that treename. If the treename consists only of ".DEL..DEL.", it will refer to "current acl UFD" or "current pwd UFD" as appropriate. Otherwise, it will omit the ">.DEL..DEL." at the end of the treename when displaying it. This functionality is useful for sizing the current ufd ("SIZE .DEL..DEL.") or sizing a top-level ufd without knowing which disk it is on ("SIZE FOOBAR>.DEL..DEL.").

Note: the current ufd functionality is a side-effect of the fact that calling SRCH$$ with a name in which the first word is -1 causes it to open the current ufd; the SRSFX$ subroutine passes the -1 on through. However, this may not work in the future, and therefore will not be supported at that time.

SLIST

New Functionality

The maximum line length has been increased from 140 to 1024 characters.

User visible bug fixes

If a file has a last line not terminated by a newline character, it will now be output correctly.

Internal bug fixes

None

Outstanding problems

None

Subject : VSRTLI

Release : Rev. 19.0

Date : December 22, 1981

1 New Functionality

None.

2 User Visible Bug Fixes

None.

3 Internal Bug Fixes

Internal changes required by VISTA were incorporated into VSRTLI.

4 Outstanding Problems

None known.

ABSTRACT

The Spool subsystem has undergone many changes at rev 19.0. Few of these changes are visible to the end user of the SPOOL program, however.

The changes that are visible to the end user are FVFU support, the header page changes, and the incompatible change to the SPOOL command which no longer allows "implied open" operations (this change should be welcomed by many a user!).

An important change is that the subroutine interface to the Spool subsystem, namely SPOOL$, is now via dynamic links; the SPOOL$ library is now a shared library. If all programs are rebuilt using the new SPOOL$ and VSPC0$ libraries in LIB, then future compatibility can be maintained at much lower cost.

Among the other changes are: the Q.CTRL file is now larger, because a second array of areas was added to the end to support future needs. The Spool subsystem now supports 32-character usernames, as PRIMOS rev 19 does. Plus a host of other minor changes.

NOTE: only the user visible functionality described in this document is supported. Some descriptions of Spool internals are included here for your convenience. These are not supported and will probably be changed in the near future.

1 EVFU SUPPORT

At rev 19.C, SPOOL provides additional support for the electronic vertical format unit (EVFU). The EVFU provides the functionality of the paper tape loop familiar to operators. This allows the administrator to assign a 'channel' number to a particular line of a form. Then a user can insert a special 'skip to channel' code in the spool file which will cause the printer to advance the form until it reaches the line with the designated channel.

There are twelve channels (numbered 1 to 12) available for this use. They are subject to the following constraints:

1. Channel 1 will always designate the first line of the form.

2. Each line of the form may have at most one channel associated with it.

3. Only channel 12 may be assigned to more than one line of the form.

The maximum length of the EVFU (also the maximum number of lines on a form) is 132 for the 300 lpm printer/plotter and 143 for the 'Band' printer.

The operator assigns channels to form lines by creating a file with the editor called an EVFU file. This file is then incorporated into a printer environment by using PROP. The spooler phantom will then load the EVFU into the printer when the environment is started with PROP.

1.1 Building an EVFU File

An EVFU file is an ASCII file that is created and modified by the operator using ED. An EVFU file must reside in the SPOOLQ UFD when it is referenced by the PROP command. Each line of the file corresponds to a line of the form being defined. The number of lines in the file must equal the number of lines on the physical form. Each line in the EVFU file will contain the channel number that is to designate that line on the form. For example, if channel 5 is to designate line 15 of the form, then the operator will put a '5' (ASCII) in line 15 of the EVFU file.

Note that since channel 1 will always refer to the first line of the form, there must be a '1' in the first line of the EVFU file. Lines that are to have no channel assigned to them may contain a '0' or be blank. The channel number must be the first non-blank character(s) on the line. A comment may follow a channel number.

Although the printer environment contains a copy of its EVFU file, that file should not be deleted. This is because it is the most convenient description of the contents of the EVFU. Also, the file must be present if the environment is ever to be modified, since it will be reinterpreted. The administrator may want to protect the EVFU files using ACLs.

2 SHARED SPOOL$ LIBRARY

The SPOOL$ subroutine is now V-mode only, and is shared. The VSPOO$ file in LIB is simply a dynamic link to SPOOL$, and the SPOOL$ file in LIB is an R- to V-mode interlude which calls a dynamic link to SPOOL$.

The sequence to share SPOOL$ which must be in C_PRMO is as follows:

```
SHARE SYSTEM>S$2167 2167
R SYSTEM>S$4000 1/12
```

This shared library uses the linkage area from 6001/67000 to 6001/67767 inclusive.

Because the libraries in LIB are now dynamic links, it is recommended that all programs which use SPOOL$ be reloaded, so that they use dynamic links instead of old (and obsolete) copies of SPOOL$. Naturally, all such programs will decrease somewhat in size, since a dynamic link (even the R-mode interluae) is much smaller than SPOOL$ itself was.

3 CHANGES TO Q.CTRL

The Q.CTRL file is now 32607 words long, whereas previous to revision 19 it was 8207 words long. The addition 24400 words comes from an area at the end of the file consisting of 200 122-word entries. These entries are used to hold additional information per queue request, and have some room left for information needed in the future.

SPOOL$ will automatically expand Q.CTRL to the new size when it is first called upon to submit a queue item and the Q.CTRL file is only 8207 words long.

The second area for a queue entry is only valid if the first word of the first area is > 1. Old copies of SPOOL$ will put 1 in the first word to indicate that an entry is present, and 0 to indicate that an entry is deleted. The new SPOOL$ puts a 2 there to indicate that an entry is present, and that the information in the second area corresponding to this entry is valid. All Spool code, old and new, checks for the first word being zero or non-zero when pulling entries out of the spool queue, so compatibility is maintained here.

However, obviously a rev 18 spooler phantom will not make use of the second area, which may have been written into by a rev 19 SPOOL$ subroutine on a different node. The converse is true; a rev 19 spooler phantom will have to substitute null information for the info in the second area if the entry it is looking at was submitted by a rev 18 despooler.

The format of the second area is currently:

```
dcl 1 second_area(200),  /* 200 entries. */
      2 user_name char(32),  /* Long form of username. */
      2 file_dtm bin(31),  /* Date-time modified of file. */
      2 pathname char(80) var,  /* Pathname or home ufd. */
      2 copy bin,  /* 0 if invalid info, 1 if copy, 2 if open. */
      2 seconds_spooled bin,  /* Seconds file spooled. */
      2 extra bin(61);  /* Unused information, inited to zeroes. */
```

<copy> is used to determine whether <pathname> is the pathname of the spooled file, or the pathname of the home ufd of the user when the SPOOL -OPEN was done.

<seconds_spooled> is used as the fifth word in the call to TIMDAT, the first four of which are in the first area.

Note: the Q.CTRL file is now 32607 words long. Please notice that 32607 is very close to 32767. which is the largest number a FORTRAN program can manipulate correctly with INTEGER*2 arithmetic. Anyone who intends to modify the Spool subsystem so that it can handle more than 200 entries or so that it has more information per entry should be aware that all calculation done by the Spool subsystem to position into the file is INTEGER*2 calculation. Therefore, attempts to reference entry number 250 would cause either wrap-around into a random area of the file, or perhaps an error code (such as "Beginning of file") from PRWF$$. To fix this, all PRWF$$ calls with positions must be checked and converted to use INTEGER*4 arithmetic.

4 CHANGES TO SPOOL COMMAND

4.1 Spooling EVFU Formatted Files

If your system has an EVFU-driven printer, files may be printed which contain codes instructing the printer to skip to a designated channel. Such a code consists of an octal 3 in the first byte of the line and the channel number in the second byte. A channel number may be specified by an octal number (1-12) or by an ASCII character - a letter from 'A' to 'L' in upper or lower case. Files that use this mechanism must be in "no format" mode when the codes are encountered. As always, this mode is entered by typing '-NOFMT' on the SPOOL command line or by inserting a 1 (octal) and a 0 (octal) into the first two bytes of a line in the file.

Note that your system administrator will have to advertise that a particular form name is associated with a particular EVFU format. Also, a user who spools a file containing EVFU channel codes will have to type the appropriate -FORM name on the SPOOL command line. It is important that a spool file does not attempt to skip to a channel number which has not been defined in the EVFU in the printer on which it is printed. Printers respond in various ways to this

error. Some skip an entire page, while others hang and display an error code. The proper use of the -FORM parameter should prevent this problem.

4.2 Implied -OPEN Now Illegal

The SPOOL command no longer allows "implied -OPEN" commands. In other words, a "SPOOL" command that does not have either: a) a treename; b) a "-LIST" option; c) a "-CANCEL" option; or d) a "-OPEN" option; is considered illegal at rev 19.0. Previously, it would be assumed that the "-OPEN" function was desired.

This would cause the command "SPOOL" alone to output the message "PRTnnn opened", when usually the user just wanted to know how to use SPOOL. Now, when one of the above 4 things are not specified on the command line, usage information is printed, and the error

Missing argument to command. Filename required (SPOOL)

is generated.

4.3 Network List Feature

The SPOOL -LIST command now has a parameter that allows the user to list all queues that his system has access to.

By specifying "SPOOL -LIST *", all started up disk are searched for SPOOLQ ufds, and displayed. Only queues which have information are displayed. Command lines such as "SPOOL -LIST * FORM WHITE" are legal.

If the "*" specifier is used, and no information was displayed, the message "All queues are empty" is displayed. If a qualifier had been given, the message "No entries in any queue matched" is displayed.

Note: this capability may not be supported in the future. Also, it is not suggested that it be used often, due to the network costs.

4.4 Network Information File

With regards to the previous section documented "SPOOL -LIST *", this section describes a new optional file to be installed in SPOOLQ if desired: it is named "NETWORK_INFORMATION.SPOOL". It consists of two ASCII lines. The first line should be the name of the system on which the SPOOLQ ufd resides, and the second line should be "REV.19" if it is a rev 19.0 PRIMOS system, or "REV.18" if it is a pre-rev 19.0 PRIMOS system. If omitted, or if not "REV.18" or "REV.19", rev 19 is assumed. The rev information is used only to determine which method to use to find out if a file is printing or not.

The first line (node name) information is only used by SPOOL -LIST. It puts the text "System " in front of that line before it lists the queue, but only if entries in the queue will be displayed.

4.5 Record Size Change

The Spool subsystem now sizes files in terms of 1024 word records, not 440 words records. This will result in different file sizes reported by the SPOOL command.

4.6 Overly Large Files

If a file is spooled which is larger than 32767 records, its size will be recorded as 32767 records regardless of how large it actually is. Therefore, it will show up in a SPOOL -LIST as 32767 records long, and the trailer page of the listing (if it ever spools) will show 32767 records.

4.7 32-Character Usernames

The SPOOL command supports 32-character usernames. The only difference this will cause is in a SPOOL -LIST; if a username is longer than 6 characters, the username will be displayed on a line by itself, and the information on the spool request will be displayed on the next line. Multiple contiguous identical long usernames will not be redisplayed, for speed.

4.8 File Printing Indicator

In a SPOOL -LIST outout, if a file is printing, a "*" will be printed next to its PRT number. If any "*"s are printed in the SPOOL -LIST, the message "* means file being printed" will appear at the end of the SPOOL -LIST. Note that this functionality may not be supported in the future.

4.9 Effect of -AS Changed Slightly

As will be described in the section pertaining to the header page, the full pathname of the spooled file is used on the third line of the header page when appropriate (and possible). This means that the "-AS" option will now only override what is seen in a SPOOL -LIST command and in the banner on the header page. It will only override the third line of the header page if the "-OPEN" option was used when submitting the file, or if the pathname information was not present.

5 CHANGES TO HEADER AND TRAILER PAGES

5.1 Full Pathname Capability

By having SPOOL$ perform calls to GPATH$ to get the full pathname of a spooled file, this pathname is printed on the header and trailer page of the listing. For a submission of a file without -OPEN, this means that the third line of the header page will be the full pathname of the submitted file (if obtainable), and the line "Pathname: <path>" will appear below the banner.

For a submission with -OPEN, the third line will be whatever the filename specified was (or -AS name), and the line "Opened from: <path>" will appear below the banner. In this case, <path> is the home ufd of the user when the submission was made.

The text "No pathname information" will replace the "Pathname: <path>" or "Opened from: <path>" text if the submission was made by a prev-rev 19 SPOOL$ subroutine.

The text "(info unavailable)" will replace "<path>" if the call to GPATH$ failed. A reason for this is, for example, that the pathname was longer than 80 characters.

Since the same pathname for either -OPEN or file copy submission is also printed on the trailer page, it should be easier to verify that listings are burst correctly.

5.2 Date/Time Modified Information

SPOOL$ also obtains the file date/time modified information when a file copy submission (i.e. not -OPEN) is performed. This information is printed below the "Pathname: <path>" line on the header page.

The text "No file modified date/time" will replace the "File last modified: <date/time>" text if the submission was made by a pre-rev 19 SPOOL$ subroutine.

The <date/time> field can be all zeroes if the UFD which contained the file only allowed "U" (Use) access to the submitting user.

Note that all of the date/time fields printed on header and trailer pages are accurate only to within 4 seconds.

5.3 Trailer Page Size in Records Changes

The "Records: n" message on the trailer page now reflects 1024-word records, not 440-word records.

Also, files longer than 32767 records will show up as 32767 records in this message.

5.4 Header/Trailer Page Text in Upper/Lower Case

The text printed on header and trailer pages is now in combinations of upper and lower case for increased readability. Due to an internal restructuring, a printer environment which has UPCASE ON will correctly convert all that text to upper case before printing it. This was not true in previous revisions, which forced all internally-generated text to the spooler phantom to be entered in upper case.

5.5 Trailer Page Changes

In addition to the date/time when the file finished printing, and the number of lines read and size of the file in records, the trailer page now contains: the username of the submitter, the "-AT" name (if not blank), the "PRT" number, the "-FORM" name (if not blank), and the pathname of the file (or the filename).

This should make verification of bursting easier.

5.6 Slash Character Printed in Banners

The "/" character, which is a legal filename character, is now printed in banners. Previously, it was replaced by a blank.

6 CHANGES TO PROP COMMAND

6.1 The EVFU Command

After an operator has created an EVFU file in the SPOOLQ UFD, he can create or modify an environment to use it. The PROP subcommand EVFU is used for this. This subcommand takes one of three options; -ON, -OFF, or -NAME <EVFU_file_name>. (Formerly, ON and OFF were missing the dashes.) -OFF is used with printers that do not contain EVFU's. -ON is used when the EVFU will not be used explicity to define channels. The spooler uses it to define the form length. For example, the 300 lpm printer/plotter which is equiped with a forms length switch would have EVFU turned off. The same device without the forms length switch would use an environment with EVFU turned on, if no use was to be made of the channels.

In addition to naming the EVFU file, an environment must have the LINES parameter set to equal the number of lines in the EVFU file (also the number of lines on the form). And as before, the TYPE parameter should be set to 0 for the 300 lpm printer/plotter and 1 for the band printer.

6.2 Starting an Environment with EVFU

The paper in the printer should be aligned to its top of form before the environment is started with PROP. This is because the printer assumes this alignment when the EVFU is loaded. If the printer is powered off the contents of the EVFU will be lost. In this case the environment should be stopped with PROP, the paper aligned to top of form, and the environment restarted.

6.3 Numeric Parameter Handling

The numeric parameters for a printer environment except for TYPE (i.e. LINES, LARGE, LIMIT, UPPER, LOWER, HEADER, WIDTH, and LINES) now have more intelligent handling.

There are now minimum and maximum values defined for each parameter, and defaults for when just the parameter name is typed. Also, some of the parameters can have the value "off". The word "off" can be used when specifying a parameter in place of its value, if legal, and is displayed in the DISPLAY command.

The default, minimum and maximum table is as follows:

Parameter	Default	Min	Max	
LENGTH	38	10	32767	
LARGE	20	0	32767	"off" is same as 0
LIMIT	off	0	32767	"off" is legal
UPPER	63	0	63	
LOWER	0	0	63	
HEADER	1	0	2	"off" is same as 0
WIDTH	108	10	140	
LINES	off	16	32767	"off" is same as 0

For parameters where "off" is the same as 0, like LINES, specifying 0 as a value is the same as specifying "off". In the case of LINES, therefore, specifying "LINES OFF" and "LINES 0" is identical - but notice that whereas "LINES 0" is legal, "LINES 1" is not, because it is too small a value. In the case of "LIMIT", since "off" is legal, and 0 is a different value, specifying "LIMIT OFF" means no limit on spooled files, whereas specifying "LIMIT 0" means print only empty files (effectively disables spooling by that printer).

It is illegal to specify "off" values for parameters that do not allow them, such as "UPPER". In a DISPLAY or PROP -DISPLAY, values which correspond to "off" values will be displayed as "off".

7 CHANGES TO THE SPOOLER PHANTOM

7.1 EVFU Capability

Explained in sections on EVFU, PROP and SPOOL.

7.2 Automatic Sizing of -OPEN Files

When a spooler phantom automatically figures out the size of a file submitted with "-OPEN" (whose size is unknown), it uses 1024-word records, not 440-word records, which will produce different file sizes at rev 19.0.

Also, after sizing the file, it now reads the Q.CTRL file in again and rescans the file. This way, it may decide to print the file whose size it just scanned, whereas previously it just went back to sleep if no other files were ready to print.

7.3 Wake Up Time

The spooler phantom now wakes up every minute, rather than every two minutes. Note that because multiple spoolers use a single semaphore, it is possible that the wake up time may be delayed when spoolers go to sleep at different times.

7.4 File Name Change

The spool program itself, which used to be "*SPPHN", is now "SPPHN.SEG", and is a SEG file.

CHANGES REQUIRED FOR REV. 19 STANDARDS COMPLIANCE

1. The Master Disk partitions are no longer A1, B1, and B2. The new partitions and their contents are:

   U1 - PRIMOS and unchargeable software run files
   V1 - PRIMOS and unchargeable software source files
   C1 - Chargeable software run files
   D1 - Chargeable software source files

2. All chargeable software is now released as two directories, except for language compilers. PRODUCTSRC now contains the products source and build files, and PRODUCT contains the files needed to run the product.

   Language compilers are now organized into three directories. PRODUCTSRC contains the compiler source, PRODUCTLIBSRC contains the run-time library source, and PRODUCT contains the run files of the library and the compiler.

   There is no change in the organization of unchargeable software.

3. All build files have been converted from cominput files of the form C_PRODUCT to cpl files of the form PRODUCT.BUILD.CPL.

4. Files in system directories CMDNCO and LIB now have standard suffices. Files in other system directories have standard suffices where an appropriate suffix has been defined. No suffices have been defined for shared segments in SYSTEM or for files in SYSOVL. Files in SYSCOM have been duplicated so that both suffixed and unsuffixed files exist. All files in source directories, including insert files, now use the new suffix scheme.

5. Installation and share files remain as cominput files, but now have standard suffices. Installation files of the form C_INSTALLPRODUCT are now PRODUCT.INSTALL.COMI. Share files of the form C_SHAREPRODUCT are now PRODUCT.SHARE.COMI.

6. All files now have a standard three line header which contains the following information:

   LINE 1: FILENAME,TREENAME, PROGRAMMER IDENTIFICATION, DATE
   LINE 2: One line description of the file
   LINE 3: Copyright information

7. At Rev. 19 some products may identify themselves by a rev number different from the master disk release number. This number refers

to the release of that particular piece of software, rather than the master disk release.

The following is a brief summary of how to install software on a Rev 19 system.

For each chargeable product, there is a cominput file to copy files needed to run the product from the Master Disk to system directories on the user's command disk. Each cominput file is named PRODUCT.INSTALL.COMI (example: COBOL.INSTALL.COMI). Each installation file resides in the directory containing the run files of the product (example: directory COBOL).

In addition to copying commands, libraries, and shared segments, the installation file copies the PRODUCT.SHARE.COMI file from the product directory to ufd SYSTEM, if the product uses shared segments.

In some cases, the number of files to be installed during an initial installation of a product may be greater than the number of files required for a reinstallation of a product. In addition, an initial installation may require that a directory be created on the user's command disk but a reinstallation will not require this. In those cases, there is a separate cominput file provided for inital installations called PRODUCT.INITINSTALL.COMI. As an example, the installation file for COBOL is listed below.

```
/* COBOL.INSTALL.COMI, COBOL, JPC-RHB, 04/01/80
/* installs COBOL into system directories
/* Copyright (C) 1980, Prime Computer, Inc., Wellesley, MA  02181
/*
FUTIL
FROM COBOL>CMDNCO
TO CMDNCO
COPY COBOL.SAVE,NCOBOL.SAVE
FROM COBOL>SYSTEM
TO SYSTEM
COPY C2014A,C2014B,C4000,CO2016
FROM COBOL
TO SYSTEM
COPY COBOL.SHARE.COMI
FROM COBOL>SYSOVL
TO SYSOVL
COPY C$$COD
FROM COBOL>LIB
TO LIB
COPY VCOBLB.BIN,NVCOBLB.BIN
QUIT
CO -CONTINUE 6
```

```
CO -END
```

For each chargeable product that uses shared segments, there is a cominput file called PRODUCT.SHARE.COMI (example: COBOL.SHARE.COMI) that contains commands that will install the shared files when invoked at the supervisor terminal. Each command share file resides in the directory containing the product (example: COBOL). These cominput files assume that the installation cominput files have copied all required share files and the share cominput file from the directory containing them to ufd SYSTEM. The group of SHARE commands is preceded by the command OPRPRI 1 and followed by the command OPR 0. SHARE cominput files to install shared libraries must include the shared library package number. As an example, the share command file for COBOL is shown below.

```
/* COBOL.SHARE.COMI, COBOL, JPC-RHB, 04/01/80
/* shares COBOL compiler and library
/* Copyright (C) 1980, Prime Computer, Inc., Wellesley, MA  02181
/*
OPR 1
SHARE SYSTEM>C2014A 2014 700
SHARE SYSTEM>C2014B 2014 700
R SYSTEM>C4000 1/3
SHARE SYSTEM>C02016 2016
SHARE 2014
OPR 0
CO -CONTINUE 6
CO -END
```

The file C_PRMO.TEMPLATE is supplied in ufd PRIRUN on the Master Disk. This file contains commands to invoke each of the PRODUCT.SHARE.COMI cominput files. A user must examine the file and delete those commands that invoke SHARE.COMI files that the user has not purchased. After the file has been modified, it must be copied to CMDNCO and named C_PRMO.

```
/* C_PRMO.TEMPLATE, PRIRUN, JNK, 07/21/81
/* TEMPLATE FOR MAKING C_PRMO FILE FOR BRINGING UP PRIMOS
/* Copyright (C) 1980, Prime Computer, Inc., Wellesley, MA  02181
CONFIG -DATA                  /* specify CONFIG file after -DATA
ADDISK                        /* specify local disks to be added
AMLC TTY                      /* specify AMLC lines
OPR 1                         /* SHARE REQUIRES OPR 1
SHARE SYSTEM>ED2000 2000      /* SHARE the editor  -  ED
SHARE SYSTEM>S2050 2050 700/* BRING UP SHARED LIBRARIES
R SYSTEM>S4000
SHARE 2050
OPR 0
PROP PRO -START              /* START SPOOLER PHANTOM
PH BATCHQ>PH_GO              /* STARTUP BATCH MONITOR
```

```
CO SYSTEM>BASICV.SHARE.COMI 7     /* SHARE BASICV COMPILER
CO SYSTEM>COBOL.SHARE.COMI 7      /* SHARE COBOL COMPILER AND LIBRARY
CO SYSTEM>DBG.SHARE.COMI 7        /* SHARE DEBUGGER
CO SYSTEM>DBMS.SHARE.COMI  7         /* SHARE DBMS
CO SYSTEM>DPTX-DSC.SHARE.COMI 7    /* SHARE DPTX-DSC
CO SYSTEM>DPTX-TCP.SHARE.COMI 7    /* SHARE DPTX-TCP
CO SYSTEM>EMACS.SHARE.COMI 7       /* SHARE EMACS
CO SYSTEM>FED.SHARE.COMI 7         /* SHARE FED
CO SYSTEM>FORMS.SHARE.COMI 7         /* SHARE FORMS LIBRARY
CO SYSTEM>FTS.SHARE.COMI 7           /* SHARE FTS
CO SYSTEM>F77.SHARE.COMI 7           /* SHARE F77 COMPILER
CO SYSTEM>MIDAS.SHARE.COMI 7         /* SHARE MIDAS LIBRARY
CO SYSTEM>PASCAL.SHARE.COMI 7        /* SHARE PASCAL COMPILER
CO SYSTEM>PL1G.SHARE.COMI 7     /* SHAPE PL1G COMPILER
CO SYSTEM>POWERPLUS.SHARE.COMI 7            /* SHARE POWER
CO SYSTEM>VISTA.SHARE.COMI 7                /* SHARE VISTA
CO SYSTEM>VRPG.SHARE.COMI 7       /* SHARE VRPG
CLOSE 7
OPR 1
SHARE SYSTEM>SP2121 2121
R SYSTEM>SP4000 1/10           /* SHARE SPL LIBRARY
SHARE SYSTEM>S$2167 2167
R SYSTEM>S$4000 1/12           /* SHARE SPOOL LIBRARIES
OPR 0
/* SET THE DATE AND TIME **********
CO -END
```

The following command files exist in ufd SYSTEM on the Master Disk. They are for doing installations of an entire Master Disk. These files must be edited to include the MFD names and passwords. They should be run in the order listed.

CREATE.STD.COMI
CREATE.ALL.COMI
INSTALL.STD.COMI
INSTALL.ALL.COMI

CREATE.STD.COMI should be run if the disk you are installing TO does not have standard master disk directories such as CMDNCO, SYSTEM, LIB, etc.

```
/* CREATE.STD.COMI, SYSTEM, DJF, 08/10/81
/* Creates standard master disk system directories
/* Copyright (c) 1981, Prime Computer, Inc., Natick MA 01760
/*
/* /* this file creates the system directories required for
/* an initial installation of Primos and unchargeable software
/*
A MFD XXXXXX   /* ADD MFD NAME + PASSWD OF DISK YOU WISH TO INSTALL ON
CREATE BATCHQ
CREATE CMDNCO
```

```
CREATE DOS
CREATE HELP*
CREATE LIB
CREATE PRIRUN
CREATE SEG
CREATE SEGRUN*
CREATE SPOOLQ
CREATE SYSCOM
CREATE SYSOVL
CREATE SYSTEM
CREATE TOOLS
CREATE RJSPLQ*              /* NEEDED ONLY FOR RJE INSTALLATIONS
CO -END
```

CREATE.ALL.COMI creates 'special' system directories which are needed to install some chargeable software. For example, to install FORMS you must have a top level FORMS* directory. The user should edit this file and delete the lines which reference software that has not been purchased.

```
/* CREATE.ALL.COMI, SYSTEM, JPC-JNK, 07/21/81
/* creates system directories needed to install chargeable software
/* Copyright (C) 1980, Prime Computer, Inc., Wellesley, MA  02181
/*
ATTACH MFD                  /* PASSWORD SHOULD BE ADDED
CREATE DBMSLB               /* REQUIRED FOR DBMS INSTALLATION
CREATE EMACS*               /* REQUIRED FOR EMACS INSTALLATION
CREATE FORMS*               /* REQUIRED FOR FORMS INSTALLATION
FUTIL                       /* REQUIRED FOR INITIAL INSTALLATION ONLY
FROM FORMS>FORMS*
TO FORMS*
UFDCPY
QUIT
CREATE FED*                 /* REQUIRED FOR FED INITIAL INSTALLATION
CREATE TOOLS                /* REQUIRED FOR PL1G INSTALLATION
CREATE POWER*               /* REQUIRED FOR POWER OR POWERPLUS INSTALLATION
FUTIL                       /* REQUIRED FOR INITIAL INSTALLATION ONLY
FROM POWERPLUS>POWER*
TO POWER*
UFDCPY
QUIT
CREATE POWERCM              /* REQUIRED FOR POWER OR POWERPLUS INSTALLATION
CREATE FAM                  /* REQUIRED FOR PRINET OR X.25 INSTALLATION
CREATE PRIMENET*            /* REQUIRED FOR PRINET OR X.25 (FAMII) INSTALL
CREATE VISTA*               /* REQUIRED FOR VISTA INSTALLATION
CREATE FTSQ*                /* REQUIRED FOR FTS INSTALLATION
CREATE RJSPLQ*              /* REQUIRED FOR RJE INSTALLATION
co -continue 6
co -end
```

INSTALL.STD.COMI installs all unchargeable software.

```
/* INSTALL.STD.COMI, SYSTEM, JPC-JNK, 08/06/81
/* Installs PRIMOS and utilities from the Master Disk
/* Copyright (C) 1980, Prime Computer, Inc., Wellesley, MA  02181
/*
/* change XXX to the current release number, 19.0 would be M190U1
/*
FUTIL
FROM <MXXXU1>CMDNCO
TO CMDNCO
UFDCPY
FROM <MXXXU1>DOS
TO DOS
UFDCPY
FROM <MXXXU1>LIB
TO LIB
UFDCPY
FROM <MXXXU1>SPOOLQ
TO SPOOLQ
UFDCPY
FROM <MXXXU1>SYSCOM
TO SYSCOM
UFDCPY
FROM <MXXXU1>SYSTEM
TO SYSTEM
UFDCPY
FROM <MXXXU1>SYSOVL.
TO SYSOVL
UFDC
F <MXXXU1>PRIRUN
TO PRIRUN
UFDCPY
F <MXXXU1>BATCHQ
TO BATCHQ
UFDCPY
FROM <MXXXU1>SEGRUN*
TO SEGRUN*
UFDC
F <MXXXU1>TOOLS
TO TOOLS
UFDC
F <MXXXU1>SEG
TO SEG
UFDC
FROM <MXXXU1>HELP*
TO HELP*
UFDC
FROM <MXXXU1>RJSPLQ*
T RJSPLQ*
UFDCPY
QUIT
CO -END
```

INSTALL.ALL.COMI installs all chargeable software by running the individual install files. The user must edit this file and delete the lines which reference software which has not been purchased.

```
/* INSTALL.ALL.COMI, SYSTEM, JPC-JNK, 07/21/81
/* installs all products from the Master Disk
/* Copyright (C) 1980, Prime Computer, Inc., Wellesley, MA  02181
/*
/* NOTE -- When installing  PRINET & X.25 pauses are encountered
/*         while running the command files to allow you to delete
/*         existing segment directories.  If you are running the
/*         install files as part of this master all.install.comi
/*         command file it is necessary to type 'co continue 22'
/*         rather than 'co continue' to resume the command file
/*         properly after a pause.
CO BASIC>BASIC.INSTALL.COMI 22
CO BASICV>BASICV.INSTALL.COMI 22
CO COBOL>COBOL.INSTALL.COMI 22
CO DBG>DBG.INSTALL.COMI 22
R DBMSEX>DBMS.INSTALL.CPL
CO DPTX-DSC>DPTX-DSC.INSTALL.COMI 22
CO DPTX-TSF>DPTX-TSF.INSTALL.COMI 22
CO DPTX-TCF>DPTX-TCF.INSTALL.COMI 22
CO EMACS>EMACS.INSTALL.COMI 22
CO RJSPLQ*>RJECOM.INSTALL.COMI 22
CO EM1004>EM1004.INSTALL.COMI 22
CO EM200UT>EM200UT.INSTALL.COMI 22
CO EM7020>EM7020.INSTALL.COMI 22
CO EMGRTS>EMGRTS.INSTALL.COMI 22
CO EMHASP>EMHASP.INSTALL.COMI 22
CO EMX80>EMX80.INSTALL.COMI 22
CO EMXBM>EMXBM.INSTALL.COMI 22
CO FORMS>FORMS.INSTALL.COMI 22
CO FED>FED.INSTALL.COMI 22
CO FTN>FTN.INSTALL.COMI 22
CO FTS>FTS.INSTALL.COMI 22
CO F77>F77.INSTALL.COMI 22
CO MIDAS>MIDAS.INSTALL.COMI 22
CO PASCAL>PASCAL.INSTALL.COMI 22
CO PL1G>PL1G.INSTALL.COMI 22
CO POWERPLUS>POWERPLUS.INSTALL.COMI 22
CO PRINET>PRINET.INSTALL.COMI 22
CO RPG>RPG.INSTALL.COMI 22
CO VISTA>VISTA.INSTALL.COMI 22
CO VRPG>VRPG.INSTALL.COMI 22
CO X.25>X.25.INSTALL.COMI 22
CLOSE 22
CO -END
```

STRIP_ACLS is a tool which removes all ACLs from a directory tree. By default, it removes ACLs from the subtree defined by the current attach point, but any tree may be specified on the command line. The user has the option of having STRIP_ACLS run FIX_DISK after it has finished removing the ACLs.

The command syntax of STRIP_ACLS is:

```
R STRIP_ACLS -HELP
R STRIP_ACLS [<treename>] [-FIX_DISK <pdev> [-COMDEV]] [-RePorT]
```

The former format causes STRIP_ACLS to list its syntax.

The latter is used to actually execute STRIP_ACLS. If no <treename> is supplied, the current directory is assumed as the starting point. Protect access is required on all directories in the subtree; if ACLs are to be removed from an entire partition, it is recommended that a Priority ACL be placed on the partition before running STRIP_ACLS.

Note

> STRIP_ACLS should not be invoked by treename; since STRIP_ACLS is a recursive CPL program, it must reside in the current directory when invoked.

If the -FIX_DISK option is given, FIX_DISK will be run after the ACLs have been removed from the disk. The <pdev> must be supplied with this option. The -COMDEV option tells FIX_DISK that the command device is being fixed. STRIP_ACLS supplies the -FIX, -CMPR, and -DUFE options automatically.

Note that if the -COMDEV option is given and STRIP_ACLS was invoked from the command partition, STRIP_ACLS will terminate abnormally after FIX_DISK has run. This does not cause a problem since all conversion and FIX_DISK activities have been completed at this point.

Caution

> If STRIP_ACLS is being used to remove all ACLs from a disk, we recommend that you use the -FIX_DISK option to clean up the disk after the ACLs have been removed. If you wish to use the disk under a rev 18 system after removing the ACLs, you MUST invoke FIX_DISK with the -CMPR option (supplied automatically by the -FIX_DISK option) before the disk may be used under rev 18.

If the -REPORT option is given, each directory which is converted back to a password directory by STRIP_ACLS will be listed as it is converted.

Note that ACLs are not the only rev 19 structure which cannot be supported by rev 18 systems. If you run a disk which has had all its ACLs removed by STRIP_ACLS on a rev 18 system, the following restrictions are still imposed by the new BADSPT file format:

- You may not use any rev 18 physical disk utilities (e.g. COPY_DISK, PHYSAV).

- You may not use the disk for paging (applies to split partitions only).

Note also that all ACLs removed by STRIP_ACLS are lost; in order to restore ACL protection to the disk, the ACLs must be replaced manually.

Subject: UPCASE

Release: 19.0

Date: June 6, 1981

New Functionality

UPCASE has been updated to meet the revision 19.0 standards.

User Visible Bug Fixes

None.

Internal Bug Fixes

None.

Outstanding Problems

None.

Environment

This revision of UPCASE should be build and run on revision 19.0 or later PRIMOS.

Build and Install Procedure

This program may be built and installed by resuming the file UPCASE>UPCASE.Build.CPL.

Subject : V-mode Fortran Library

Release : Rev. 19.0

Date : December 1, 1981

1. New Functionality

   none

2. Problems Fixed

   A. The following Polers have been fixed :

      11985 F$INQF no longer closes any Command_Input_File which the user happens to have open at the time of inquiry.

      22050 MIN [F$MI11] is now a separate module, and gives correct results if the arguments are within legal bounds.

      27250 F$IO77 does correct outputs with leading zeros only when necessary.

      28524 SQRT allows more accurate comparisons between returned results.

      29556 DEXP returns 1.0 if its argument is 0.0 .

      29621 F$IO77 now accepts B-format statements with surrounding blanks.

      30110 P$ATOA traps numbers with multiple decimal points and delivers an error message.

      30249 F$IOFTN operates properly on multiple, internal sequential commas.

      31198 NAMEQ! now checks for lower case 'a'.

      31488 F$IO77 handles non-word aligned charcter output from internal formats correctly.

      32166 F!IOFTN now handles encodes of non-word aligned characters correctly.

      32365 P$ATOA now correctly handles rounding of floating point numbers.

      32724 CABS no longer overflows if the argument is within legal bounds.

32927 This is a duplicate of 31488.

35387 F$IOFTN now accepts B-format statements which may be surrounded with blanks.

40213 F$CLOS can delete a scratch file from a passworded UFD, providing that the program is initiated by a user with owner status.

40357 This is a duplicate of 22050.

3. Outstanding Problems

A. There are some outstanding Polers listed in the on-line POLERS data base.

4. Changes

A. Binary F77 I/O now works 20 faster.

B. NAMELIST support for F77 has been unshared as we needed the room in the shared library (and NAMELIST was thought to be little used).

C. Two more logical units were added to the IOCS system for use by PL1. The units are 140 and 141 for printer units 0 and 1, respectively. FTN / F77 programs may reference these units as well.

D. The block that determines whether or not the Library is initialized is now located in the Command Processor Stack, which we are running under upon entry.

Subject: VPSD

Release: 19.0

Date: June 12, 1981

1 Modifications

VPSD has been modified to conform to the Standard for Master Disk Release.

2 Installation and Build Procedures

Standard: build using VPSD.BUILD.CPL.

* NOTE: All products have been modified to conform to master disk standards. For a description of these modifications, please read INF019>STANDARDS.RUNO.