SOFTWARE RELEASE BULLETIN
NOS/VE (CYCLE 18)

DISCLAIMER: NOS/VE is intended for use as described in this document. CONTROL DATA CORPORATION cannot be responsible for the correct operation of features or parameters not described.

1.0 INTRODUCTION

1.0 INTRODUCTION

The NOS/VE Cycle 18 Software Release Bulletin (SRB) is to be used with the NOS/VE Installation and Upgrade Manual (60463913) for installing Control Data Corporation systems. Control Data recommends that the SRB be read in its entirety prior to software installation.

1.1 SCOPE

The SRB is the vehicle to document any changes to the Installation and Upgrade Manual after it has gone to print and any system deficiencies. It is necessary to install NOS 2.2 Level 602 and Level 587 of the Common Products before installing NOS/VE.

1.2 CONTENT

This document contains information to be used in installing NOS/VE Cycle 18 with the Level 1V Product Set. Sections of this document include 1) installing the system, 2) operating system notes and features including undocumented commands and known system deficiencies, 3) product set notes and features, and 4) an appendix containing NOS/VE Peripheral Maintenance Support information.

2.0 FEATURES AND PRODUCTS

2.0 FEATURES AND PRODUCTS

A description of Operating System and Product Set features may be found in the NOS/VE 1.0 Software Availability Bulletin (SAB). Please refer to that document for NOS/VE Cycle 18 system feature information.

All references in the NOS/VE 1.0 SAB to the CYBER 170-835 and CYBER 170-855 should be updated to include the CYBER 170-815, CYBER 170-825, and CYBER 170-845. All references in the NOS/VE 1.0 SAB to NOS 2.1 Level 588 should be modified to NOS 2.2 Operating System Level 596 and Common Product Level 602.

The release materials for NOS/VE Cycle 18 include the following software products.

Product Name	External Identification
• NOS/VE Deadstart Generation and Modification Package	• NOS/VE - 01
• NOS/VE System Core and Job Template Memory Image Maintenance	• Maintenance
• COBOL for NOS/VE	• NOS/VE - 02
• SORT/MERGE Utility	• NOS/VE - 03
• Required Products	• NOS/VE - 04
• File Management Utility	• NOS/VE - 05
• FORTRAN for NOS/VE	• NOS/VE - 06
• CYBER Implementation Language	• NOS/VE - 07
• NOS/VE On Line Manuals	• NOS/VE - 08

2.0 FEATURES AND PRODUCTS

The list of NOS/VE Reference Manuals includes the following:

Operating System Manuals	Publication Number
System Command Language for NOS/VE Language Definition Usage	60464013
System Command Language for NOS/VE System Interface Usage	60464014
System Command Language for NOS/VE + online Quick Reference	60464018
Source Code Utility for NOS/VE + online Usage	60464313
Object Code Management for NOS/VE + online Usage	60464413

CYBIL Manuals	Publication Number
CYBIL for NOS/VE Language Definition Usage	60464113
CYBIL for NOS/VE File Interface Usage	60464114
CYBIL for NOS/VE System Interface Usage	60464115

FORTRAN Manuals	Publication Number
FORTRAN for NOS/VE + online Usage	60485913

COBOL Manual	Publication Number
COBOL for NOS/VE + online Usage	60486013

Language-Independent Tools/Utilities Manuals	Publication Number
File Management Utility for NOS/VE Usage	60486413
Sort/Merge Utility for NOS/VE + online Usage	60486113

2.0 FEATURES AND PRODUCTS

Math Library for NOS/VE Usage	60486513
Family Administration for NOS/VE Usage	60464513

Migration Aid	Publication Number
Migration Guide for NOS to NOS/VE + online	60489503

Error Messages	Publication Number
Diagnostic Messages for NOS/VE + online	60464613

Site Manuals	Publication Number
NOS/VE Installation and Upgrade Usage	60463913
NOS/VE Operations Usage	60463914

3.0 INSIALLAIION_AND_OPERAIIONS_NOIES

3.1 INSIALLAIION

3.1.1 NOS INSTALLATION

1. The NOS User Name creation described in the NOS/VE Installation and Upgrade Manual implies that the default MODVAL DS value is adequate. DS=5 is the appropriate value.
   PSR NV00011

2. PASSON and IRHF both run out of the SYSTEMX account. Because of this, the password must be set to SYSTEMX and the resource validations must be set to their maximum.
   PSR NV00002

3. The "UN=* parameter on the *SETVE* command must have its resource validations set to the maximum. Failure to do this can cause the deadstart job to fail with 'PRU LIMIT' or 'SRU LIMIT'.
   PSR NV00002

3.1.2 CONTROLWARE INSTALLATION

1. To install the 7155 Disk controlware the following process must be followed:

   a. Copy the Controlware card deck to direct access file FMDBIN in a NOS user number.

   b. Create indirect access file CWFMD containing the line

      *COMMENTbPPU/FMDb885bFIRMWAREbMA721-A09(PN67185849)bYY/MM/DD
      where b represents a space and YY/MM/DD should be filled in with the appropriate date.

   c. Return both files.

   d. Create file GENFMD containing the following procedure:

      .PROC,GENFMD.
      GET,CWFMD.
      ATTACH,INHOLD=FMDBIN.
      REWIND,INHOLD.
      RFL,50000.
      RBR(INHOLD,FMD) (the second argument here must match the PPU

```
name in the comment card)
WBR(LGO)
LIBEDIT(P=LGO,B=O,I=CWFMD,L=O,N=FMD)
```

e. Run this procedure from the NOS terminal. Note that this sequence of commands must be run as a procedure and will not work if typed in as individual commands at the terminal.

f. The resultant file FMD may be saved or written to tape and made part of the NOS Deadstart tape via Libedit or Sysedit.

3.1.3 INSTALLATION OF LARGE SECTOR FMD DIAGNOSTICS

Installation of the CML L160 can be done the same way CML L155 was installed with the addition of the 885LS parameter. The installation process is described in Section 1 of the CML Reference Manual publication 60455980, Rev.K. The parameter 885LS for large sector 885 disk drives is described below. The device code for 885 large sector is 16 octal.

(This item goes with Table 1-3 Hardware Configuration Parameters sheet 3 of 3 on page 1-19 of the above mentioned CML Reference Manual.

Parameter Name	Used in Job	Function
885LS	CML2,CML5	Specifies that large sector 885 disk drives are included in the system configuration. Causes MALET diagnostics FLD and DL8 to be loaded and saved. Will also cause FMC and FMU to be loaded and saved if not done through the inclusion of other parameters. (7155/885/DEMH).

3.1.4 NOS/VE INSTALLATION

1. The description of Release Materials in section 1 of the Installation Handbook for NOS/VE 1.0.2 is incorrect.

   The tape described as NOS/VE-04 is actually labelled PRODUCTS , but still contains the required NOS/VE products.

   The tape labelled NOS/VE-04 contains the Online Manuals and is an optional tape. It is a 180 tape written by the Backup_Permanent_Files utility and contains 15 files which comprise the $SYSTEM product subcatalog. They are loaded via the Restore_Permanent_Files utility.

2. Several operational problems have been encountered by sites when maintaining their :$SYSTEM.$SYSTEM files using the BACKUP_PERMANENT_FILE and RESTORE_PERMANENT_FILE utilities. These problems occur due to various files which are created during NOS/VE deadstart. The problems which arise, and the appropriate remedial action are described below:

   a. Circumstances arise where an INSTALLATION deadstart is necessary followed by restoration of the permanent file base using the RESTORE_PERMANENT_FILE utility. When deadstart has completed, the following permanent files may have been created:

      uuu.$FAMILY_USERS - This file is created during processing of the Initialization prolog. The content of this file typically does not match the current validation contained on a site's archived permanent file base. Remedial action: See the next item in this section describing the required remedial action.
      PSR NV00118

      SYSTEM_PROLOG and SYSTEM_EPILOG - These files are also created during Initialization prolog processing. Corresponding SUPPLEMENT files are created as well. Any changes a site may have made to these files which reside in catalog $SYSTEM.PROLOGS_AND_EPILOGS will be lost. Remedial action: Until adequate provisions are made for maintaining and upgrading these files, it is recommended that the family user administrator provide user prolog and user epilog files which reflect the processing performed during user LOGIN and LOGOUT. These user prolog and epilog files are associated with family users by the SET_USER_PROLOG and SET_USER_EPILOG commands of the ADMINISTER_USER utility.
      PSR NV00024

      System Libraries - These files are created from files loaded from the NOS/VE deadstart file whenever the system core commands INITIALIZE_DEADSTART_DEVICE or SET_SYSTEM_ATTRIBUTE LOADFILES 1 are processed. They include the following files:

      :$SYSTEM.$SYSTEM.OSF$OPERATOR_LIBRARY.$NEXT
      :$SYSTEM.$SYSTEM.OSF$OPERATOR_COMMAND_LIBRARY.$NEXT
      :$SYSTEM.$SYSTEM.OSF$COMMAND_LIBRARY.$NEXT (Boot Version)
      :$SYSTEM.$SYSTEM.OSF :$SYSTEM.$SYSTEM_LIBRARY.$NEXT
      :$SYSTEM.$SYSTEM.OSF$SITE_COMMAND_LIBRARY.$NEXT
      :$SYSTEM.$SYSTEM.CYBIL.CYF$RUN_TIME_LIBRARY.$NEXT
      :$SYSTEM.$SYSTEM.OCU.BOUND_PRODUCT.$NEXT (Boot Version)

      Any changes to these files must also be made to NOS/VE deadstart file, otherwise they will be lost when NOS/VE is

deadstarted. Operational problems arise with the files identified above as "Boot Version" files. These "Boot Version" files contain skeletal versions of the system command library and Object_Code_Utilities which are required for deadstart to complete normally. The fully installed version of these files is required for normal NOS/VE usage. Since these files are created during deadstart, they will not be restored during subsequent execution of the RESTORE_PERMANENT_FILE utility. Remedial action: Either rerun the RESTORE_PERMANENT_FILE utility and specify the RESTORE_EXISTING_FILES directive for these files; or specify SET_SYSTEM_ATTRIBUTE LOADFILES 1 and SET_SYSTEM_ATTRIBUTES MULTD 1 during a CONTINUATION deadstart, and reload the NOS/VE required products tape when the NEXTDSTAPE response is requested.

b. When any of the system libraries have changed on the NOS/VE deadstart file, it is necessary to specify SET_SYSTEM_ATTRIBUTE LOADFILES 1 during a CONTINUATION deadstart. Again, the "Boot Version" files are created which are inappropriate for a production workload. Remedial action: Either delete cycle $HIGH of the "Boot Version" files, or specify SET_SYSTEM_ATTRIBUTE MULTD 1 and reload the required products tape when the NEXTDSTAPE response is requested during deadstart.

The problems associated with "Boot Version" files will be corrected in a future release.
PSR NV00077

3. Automated validation performed during an INSTALLATION deadstart creates a $FAMILY_USERS file for the NVE family. If you subsequently reload permanent files to family NVE from a BACKUP_PERMANENT_FILE file, catalogs which were not defined in the automated validation cannot be accessed. Likewise, any catalog information which has not been updated in the automated validation file is incorrect for the catalogs restored using RESTORE_PERMANENT_FILE. The recommended sequence of events in which an installation deadstart can be performed, and the family NVE permanent files restored from a previous level is as follows:

   a. BACKUP_PERMANENT_FILE all NVE catalogs and files to tape in a task that runs in ring 3 from a job other than the system job (console) that runs under $SYSTEM. This is done by bracketing the commands associated with the backup with TASK, TASKEND commands. The ring number is established by using the RING parameter of the TASK command. For example, type from the console:

```
JOB,BACKUP
  REQMT,DUMP,..
    T=MT9/46250,RING=ON,EVSN=(/*01/*)
```

```
TASK,RING=3
  BACPF,BF=DUMP
    BACAF
  QUIT
TASKEND
JOBEND
```

NOTE: The task command can be specified only in an interactive job or in a batch job running under $SYSTEM.

b. Perform an Installation Deadstart of NOS/VE, but enter the command GO rather than AUTO to terminate System Core command processing. An additional pause for operator intervention occurs just prior to the loading of system libraries from the NOS/VE deadstart file. Enter the command DETF VALIDATE/OFAMILY/ONVE prior to entering the NEXTDSTAPE or GO response which causes deadstart processing to resume. This command will disable automated validation.

c. Restore the $FAMILY_USERS file that was backed up in step a. (If you failed to disable validation prior to this step, you must LOGIN as the NVE family user administrator, and change the name of the $FAMILY_USERS file using Change_Catalog_Entry.)

d. Restore the remaining NVE family files which were backed up in step 1.

e. Execute the Create_Family command from the operator console

PSR NV00079

4. The procedure Verify_Installed_Software which is documented in the NOS/VE Installation and Upgrade manual (Publication number 60463913 A) is not available at this time.

5. Multiple NOS/VE upgrades using the CREATE_JOB_TEMPLATE command result in a large number of files being created on the deadstart disk. Each successful execution of the CREATE_JOB_TEMPLATE command creates a directory file and several uniquely named segment files. The directory file name is specified by the TEMPLATE_NAME parameter, and contains the name of every segment file which constitutes the job template. Whenever a job template is no longer being used, the DELETE_JOB_TEMPLATE command should be used to remove the directory file from the deadstart disk. Segment files are removed by this command whenever the directory file is removed. The console entry for this command is:

   K.DELETE/OJOB/OTEMPLATE TN=name

   PSR NV00023

6. Online manuals are released on a NOS/VE BACKUP_PERMANENT_FILE tape at this release. In a future release of NOS/VE, all products will be released in this format. To install the online manuals at this release level, you must reload the files on this tape into the

$SYSTEM.MANUALS subcatalogs of the $SYSTEM family. Create the following NOS/VE job to accomplish this task:

```
JOB,RESTORE
request_magnetic_tape file=backup type=mt9$1600 ..
 ring=no evsn=nve004

WHEN any_fault DO
 display_value osv$status output=$response
 request_operator_action message=' RESTORE FAILED - CHECK TAPE'
 LOGOUT
WHENED

RESTORE_PERMANENT_FILE
 restore_all_files backup_file=backup
QUIT

JOBEND
```

The above job may be created as a NOS or NOS/VE text file. This text file is then accessed from the operator console, using the ATTACH_FILE or GET_FILE command, and the INCLUDE_FILE command is executed to submit this job as a $SYSTEM family batch job.

7. Remote installation and maintenance of the operating system requires assistance from the operator's console. The installation and maintenance procedures are being modified to reduce the need for assistance at the console. Until Remote Deadstart is available, this need will not be totally eliminated.

8. In cycle 16 and earlier, the user_prolog and user_epilog entries for validated users contain the following:

```
user_prolog='include_file $system.prologs_and_epilogs.user_prolog'
user_epilog='include_file $system.prologs_and_epilogs.user_epilog'
```

In cycle 17 and later, the user_prolog and user_epilog entries must now contain the following:

```
user_prolog = '$user.prolog'
user_epilog = '$user.epilog'
```

Users validated in cycles below 17 must have the entries changed to the above mentioned information by the Family Administrator.

The VALIDATE_USER procedure has been modified to accomodate the change. Those users created using this procedure in cycle 17 or later will have the correct information for the user_prolog and user_epilog entries.

There has been a change to the type of value supplied as parameter to the SET_USER_PROLOG and SET_USER_EPILOG subcommands of ADMINISTER_USER. The change is as follows:

```
set_user_prolog prolog='<file_reference>'
set_user_epilog epilog='<file_reference>'
```

9. Recall from the Installation and Upgrade manual that to advance to a new NOS/VE system and have the NOS/VE permanent file base intact, it is necessary to install the system and recovery job template from the new system using the old system. After the templates have been installed, the terminate_system command is used to bring the system down. Recall also that as soon as the system has terminated, another deadstart of NOS/VE (recovery deadstart) occurs automatically. It is essential that this recovery deadstart be allowed to completed for the old system because this recovery deadstart is the checkpoint of the previously running system (i.e., permanent files are updated to be consistent with memory contents at termination and deadstart termination status is recorded).

   An experienced installation analyst may realize that if the recovery deadstart following termination is not allowed to complete, a recovery deadstart will be automatically initiated when the next continuation deadstart is attempted. If the recovery deadstart following termination for the old system is not allowed to complete, the new system (cycle 18) will be unable to deadstart and the old system will have to be deadstarted to allow recovery to properly complete.

3.2 NOS/VE_EXECUTIVE

1. NOS/VE will not support more than 10 tasks concurrently defined per job. The system does not enforce this limit; users must be careful not to exceed this limit or they may cause the system to fail.

3.3 OPERATOR_COMMANDS

1. Restarting IRHF and IFEXC on the 180 Side

   IRHF should automatically restart if it aborts. There is a constant check to see if it has aborted. To bring up IRHF manually, the operator must first bring down IRHF, then bring it up again. The same holds true for IFEXEC.

To bring down IRHF, type in at the 180 NOS/VE operator's console:

```
K.DEACTIVATE/OSYSTEM/OTASKS TASK/ONAMES=RHINPUT
K.DEACTIVATE/OSYSTEM/OTASKS TASK/ONAMES=RHOUTPUT
```

Then to bring IRHF back up, type in at the 180 NOS/VE operator's console:

```
K.ACTIVATE/OSYSTEM/OTASKS TASK/ONAMES=RHINPUT
K.ACTIVATE/OSYSTEM/OTASKS TASK/ONAMES=RHOUTPUT
```

To bring down IFEXEC, type in at the 180 NOS/VE operator's console:

```
K.DEACTIVATE/OSYSTEM/OTASKS TASK/ONAMES=IFEXEC
```

Then to bring IFEXC back up, type in at the 180 NOS/VE operator's console:

```
K.ACTIVATE/OSYSTEM/OTASKS TASK/ONAME=IFEXEC
```

PSR NV00032

2. Restarting IRHF and PASSON on the 170 Side

To make provisions for restarting IRHF and PASSON, a site analyst or an operator should create two procedures and save each one on an indirect permanent file that is located under user index 377777B.

Procedure 1. The file name should be MSSIRHF and should look like this:

```
.PROC,MSSIRHF.
COMMON,SYSTEM.
GTR,SYSTEM,NVELIB,U,ULIB/NVELIB
UNLOAD,SYSTEM.
LIBRARY,NVELIB.
RUNJOBS(IRHF)
```

Procedure 2. The file name should be MSSPASS and should look like this:

```
.PROC,MSSPASS.
COMMON,SYSTEM.
GTR,SYSTEM,NVELIB,U,ULIB/NVELIB
LIBRARY,NVELIB.
RUNJOBS(PASSON)
```

Once these are in user index 377777B, IRHF is restarted by simply

typing the following in at the 170 operator's console:

MSSIRHF.

PASSON is restarted by typing in at the 170 operator's console:

MSSPASS.

PSR NV00016 and PSR NV0E293

3. The NOS application (VEIAF) that connects NOS/VE jobs to NAM will sometimes produce the flashing console message:

"PASSON ABNORMAL 18"

This message means PASSON is discarding an unsolicited message generated between VEIAF and the first read request from NOS/VE.

This should be ignored. Clear the message with 'GO,jsn.'

3.4 CONFIGURATION MANAGEMENT

1. The INITIALIZE_DEADSTART_DEVICE system-core command and the INITIALIZE_MS_VOLUME Logical Configuration Utility subcommand do not warn the operator that the volume has been previously initialized. This is of particular concern in INITIALIZE_MS_VOLUME because the operator may misspell the element_name of the device and inadvertently destroy permanent files.

2. If the physical configuration consists of one channel, one mass storage controller, and one mass storage unit, one can describe this initially using system-core commands. However, this configuration needs to be redescribed using Physical Configuration Utility and Logical Configuration Utility subcommands in order to get through an installation deadstart.

3. For optimal system performance all NOS/VE disk units should be defined during the install deadstart. This is important so that all of the logical configuration is available for allocating product set files.

4. During recovery when operator intervention has been specified, the logical configuration utility (LCU) is invoked both before and after the configuration transition. The first invocation allows the operator to change which elements in the physical configuration are logically configured. The second invocation allows the operator to initialize and add volumes to the system

set. This is useful if the first invocation of the LCU happens to add one or more volumes to the logical configuration.

5. The ability to invoke the LCU during a continuation deadstart as described in the previous bullet has been added.

6. The physical configuration utility subcommand, INSTALL_PHYSICAL_CONFIGURATION has a new parameter, PHYSICAL_CONFIGURATION. This parameter specifies the name of a local file that will receive information about the physical configuration being debugged. Use of this parameter will cause the configuration to NOT be installed. The file produced by this command can be used as input to the INSTALL_LOGICAL_CONFIGURATION subcommand of the logical configuration utility. The abbreviation for PHYSICAL_CONFIGURATION is PC.

7. The logical configuration utility subcommand, INSTALL_LOGICAL_CONFIGURATION has a new parameter, PHYSICAL_CONFIGURATION. This parameter specifies the name of a local file that will provide information about the physical configuration being debugged. Use of this parameter will cause the configuration to NOT be installed. The abbreviation for PHYSICAL_CONFIGURATION is PC.

8. A new keyword has been added to the command table for the SET_SYSTEM_ATTRIBUTES command, SUSPENDED_JOB_TIMEOUT_LIMIT. It provides the capability to change the timeout limit for detached interactive jobs via the SETSA command by entering:

   SET_SYSTEM_ATTRIBUTES N=SUSPENDED_JOB_TIMEOUT_LIMIT V=nnnnnnn

   where nnnnnnn represents milliseconds and has a valid range of 600000 - 10800000 (10 minutes - 3 hours).
   NV0E528

3.5 DEVICE_MANAGEMENT/RECOVERY

1. If any disk volume being used by NOS/VE becomes full, the following message will periodically appear on the MDD screen:

   AAAAAA - volume out of space
   (AAAAAA is the vsn of the volume)

   Any task that is requesting space on a full volume will hang waiting for space. Some space may be obtained by asking users (if they are able) to delete permanent files and detach local files. If the disk full condition persists, the NOS/VE system should be taken down and brought back up. This action will release most of

the temporary file space that was in use. If the disk full was caused by permanent files, then the disk will be nearly full after the recovery, and some action--such as backing up permanent files and deleting some files--must be taken immediately.

It is possible that a disk full situation will occur that cannot be recovered. This will have happened if the "volume out of space" message appears during a deadstart before the system is up. In this case case permanent file volumes must be initiatized and reloaded from a previous backup dump.

2. If a continuation deadstart fails after disk full with a 180 CPU monitor fault, it is possible that the failure is due to unprinted files in the output queue, one or more of which are no longer printable. The failure is a job mode software failure and does not leave the MCR set in monitor mode. The message indicating this failure is "HR - MONITOR FAULT".

   Corrective action:

   On the next attempt at a continuation deadstart, enter the following command when the system asks for operator supplied core commands:

   SET_SYSTEM_ATTRIBUTE HALTRING 0

   This command will allow deadstart to complete. If, at this time, you find that task RHOUTPUT has terminated (by looking at the system job log by doing a DISSL), and files exist in the job output queue (DISPLAY_CATALOG $SYSTEM.$JOB_OUTPUT_QUEUE) you can be reasonably sure that you have encountered this problem.

   Delete the contents of the job output queue (DELETE CATALOG_CONTENTS $SYSTEM.$JOB_OUTPUT_QUEUE), terminate NOS/VE, and do another continuation deadstart.

3. When recovering without a memory image, the EOI for a file is set to the highest allocation-unit which has been initialized on the device. This is probably not what the actual end-of-information (EOI) shows in the memory image. The file's owner is not informed that a different EOI has been chosen. This means that the user cannot use BAM to read the file. The file needs to be recreated. PSR NV0E524

3.6 INTERACTIVE

1. When using X25 to access NOS/VE, a terminal may intermittently hang allowing no input or output. TIP commands break (such as

escape-CH) out of this problem. This is a CCP problem that has been PSRd and corrected in CCP.
PSR CC5A230

3.7 PERMANENT_FILE_UTILITIES

1. There is a timing problem/data sensitive bug that may occassionally cause the RESTORE_PERMANENT_FILE command to abort. If this occurs, the operator should restart RESTORE with the failing reel. If the problem does not re-occur, only one file is lost. If the problem still occurs, restart RESTORE again but with the reel that follows the failing reel. In the second case, all files on the tape following the point where the failure occurred are lost.

4.0 OPERATING SYSTEM NOTES AND CAUTIONS

4.1 SYSTEM COMMAND LANGUAGE

1. A command that depends on the "path name" for a file is affected if the file has been explicitly attached (attach_file, create_file commands).

   The "path name" that such a command will encounter is the "local path name" rather than the path specified when the command was called. For example:

   ```
   ATTACH_FILE $USER.X LFN=Y
   REPLACE_FILE $USER.X
   ```

   will result in a file called Y being referenced in real state rather than X. Users should not mix implicit and explicit attaching of a file.
   PSR NVOD753

2. The $PROGRAM function does not return the open position designator for the LOAD_MAP file.
   PSR NVOD483

3. The SET_SENSE_SWITCH (SETSS) command will not accept a "user supplied job name" but will only work when given a "system supplied job name".
   PSR NVOD871, PSR NVOD888

4. The constants $MIN_INTEGER and $MAX_INTEGER have values of -(2**48-1) and 2**48-1, respectively, rather than -(2**63) and 2**63-1.
   PSR NVOD609

5. A new display_option has been added to the display_catalog and display_catalog_entry commands. For the former command this display option is called CONTENT(S) or C; and for the latter command it is called CYCLE(S) or C. Also, a parameter called DEPTH has been added (following the output) parameter) to both commands that works in conjunction with the added display options. These options show the amount of space occupied by catalogs, files, and cycles. The depth parameter determines the level of detail that is displayed.

   DEPTH=1 gives a 1 line summary of the catalog (or file). DEPTH=2 (the default) gives a breakdown of files and subcatalogs contained directly in a catalog (or the cycles of a file). A DEPTH greater

than 2 is only applicable to catalogs and specifies a further breakdown of the catalogs and files contained within the subcatalogs of the catalog. To get complete information about all subcatalogs, files and cycles contained in a catalog, specify DEPTH=ALL. These changes did not make it into the manuals.

6. The ACCEPT_LINE command now has an optional LINE_COUNT (LC) parameter which can be used to specify an SCL integer variable to receive a count of the lines read by the command.

   The $FILE function now supports the GLOBAL_FILE_POSITION (GFP) attribute. The values for this attribute are described under the DISPLAY_FILE_ATTRIBUTES command. A GLOBAL_FILE_POSITION attribute value is returned by the function as a string with all letters in upper case.

   These facilities can be used to detect EOI (End-Of-Information) when reading a file by means of the ACCEPT_LINE command.

7. Any files created by the program interface cannot be displayed with DISC $LOCAL. For example, the default files created in FORTRAN do not appear when subsequently entering DISC $LOCAL.
   PSRs NVEA052, NVEA055, NVEA056

8. JOB/JOBEND does not work properly when it encloses commands or text containing semicolons. All semicolons appearing at end-of-line are wrongly deleted; embedded semicolons are retained however. This problem means that JOB/JOBEND cannot be used, for example, to compile a CYBIL program, because the source code (assumed to be within a COLLECT_TEXT) will have its semicolon delimiters stripped away; the compiler then issues hundreds of error messages. Use LOGIN/LOGOUT to avoid the problem.
   PSR NV0F060

4.1.1 ADDITIONAL COMMANDS (NOT IN MANUALS)

4.1.1.1 DISPLAY_COMMAND_PARAMETER(S) (DISCP)

This command displays information about the parameters for a command. The information includes the names of the parameters, their types (including allowed keyword values) and their default values (or an indication that the parameter is required).

This command is not intended to be a replacement for information in on-line manuals, but rather a "memory jogger" for any command. It will work for any command that could be called at the point where this command is called (including system supplied commands, SCL procedures, user programs and utility subcommands).

4.0 OPERATING SYSTEM NOTES AND CAUTIONS
4.1.1.1 DISPLAY_COMMAND_PARAMETER(S) (DISCP)

This command can be used to obtain information about the parameters for any command that uses SCL services to parse its parameters.

```
display_command_parameters command=<command>
      [output=<file reference>]
      [status=<status variable>]
```

command : c: This parameter specifies the command for which information about parameters is sought.

output : o: This parameter specifies the file to which the parameter information is written. Omission will cause $OUTPUT to be used.

status: This is the standard status parameter.

4.1.2 ADDITIONAL FUNCTIONS (NOT IN MANUALS)

4.1.2.1 $COMMAND_SOURCE

This function is used to determine where the processor for the requesting command was found. The source (file or catalog) of the command is returned as a string.

By its nature, this function isn't particularly useful when used in the expression for a parameter to a command, since, in that case, it will return the source of that command. Therefore, this function is normally used in an assignment statement.

```
<$command_source> ::= $COMMAND_SOURCE [<(> <)>]
```

```
Example: "The following proc resides on an"
         "object library in some catalog."
         PROC sample_command
           cs = $command_source
           cat = $path($fname(cs), catalog)
           execute_task $fname(cat//'.sample_program)
         PROCEND
```

4.1.2.2 $PREVIOUS_STATUS

This function is used to obtain the completion status of the previous command.

```
<$previous_status> ::= $PREVIOUS_STATUS [<(> <)>]
```

```
Example: collect_text display_status
           PROC display_status, diss (
                 status: status = $previous_status)
```

```
        display_value $value(status)
      PROCEND display_status
    **
    create_variable s kind=status
    create_variable x status=s
    display_status
    NORMAL STATUS
    create_variable x status=s
    display_status
    --ERROR-- X is already declared as a variable.
```

4.1.2.3 $QUOTE

This function is used to quote a string.

```
<$quote> ::= $QUOTE <(> <string expr> <)>
```

```
Example: s = 'ABC''DEF'
         q = $quote(s)
         display_value s
         ABC'DEF
         display_value q
         'ABC''DEF'
```

4.1.2.4 $SCAN_ANY

This function is used to search a string for any one of a set of characters and return the index in the string of the found character. If no character from the set appears in the string, zero is returned.

```
<$scan_any> ::= $SCAN_ANY <(> <char set> <,isp>
                              <string expr> <)>

<char set> ::= <string expr>
```

```
Example: digits = '0123456789'
         s = 'TEMP_32'
         display_value $scan_any(digits, s)
         6
```

4.1.2.5 $SCAN_ANOTANY

This function is used to search a string for any character that is not in a set of characters and return the index in the string of the found character. If only characters from the set appear in the string, zero is returned.

```
<$scan_notany> ::= $SCAN_NOTANY <(> <char set> <,¦sp>
                                   <string expr> <)>

<char set> ::= <string expr>

Example: digits = '0123456789'
         s = 'TEMP_32'
         display_value $scan_notany(digits, s)
         1
```

4.1.2.6 $SCAN_STRING

This function is used to search a string for another string, called the pattern, and return the index of the first character of the pattern in the string. If the pattern is the null string, one is returned. If the pattern is not found in the string, zero is returned.

```
<$scan_string> ::= $SCAN_STRING <(> <pattern> <,¦sp>
                                  <string expr> <)>

<pattern> ::= <string expr>

Example: s = '123_abc_9'
         p = 'abc'
         display_value $scan_string(p, s)
         5
```

4.1.2.7 $TRANSLATE

This function is used to change characters in a string according to a translation table. The translation table is a string of 256 characters which is utilized according to the following algorithm.

```
         result = ''
         for i = 1 to $strlen(string) do
           j = $ord($substr(string, i))
           result = result // $substr(table, j+1)
         forend
```

Two standard translation tables are provided. Use of LOWER_TO_UPPER (LTU) produces a string with all lower case letters translated to their upper case counterparts. Use of UPPER_TO_LOWER (UTL) produces a string with all upper case letters translated to their lower case counterparts.

```
<$translate> ::= $TRANSLATE <(> <translation table> <,¦sp>
                                <string expr> <)>

<translation table> ::= <string expr>
```

```
                        ! LOWER_TO_UPPER ! LTU
                        ! UPPER_TO_LOWER ! UTL

Example: display_value $translate(lower_to_upper, '123_abc')
         123_ABC
```

4.1.2.8 $TRIM

This function is used to remove trailing space characters from a string.

```
<$trim> ::= $TRIM <(> <string expr> <)>

Example: s = 'STRING '
         display_value '<'//s//'>'
         <STRING>
         display_value '<'//$trim(s)//'>'
         <STRING>
```

4.1.3 ADDITIONAL CONTROL STATEMENTS (NOT IN MANUALS)

4.1.3.1 PUSH_COMMANDS

```
<push commands> ::= PUSH_COMMANDS
```

The purpose of this statement is to cause the source (file or catalog) of the issuing command to be pushed onto the top of the "dynamic command list". The entries in the dynamic command list are searched after the commands belonging to any active utilities and before the command list entries manipulated by the SET_COMMAND_LIST command. The effect of this statement is removed (popped) when the issuing command terminates.

4.2 PERMANENT_FILES

1. The DISPLAY_CATALOG and DISPLAY_CATALOG_ENTRY commands, when used on a catalog or file owned by another user, should only provide information for catalogs and/or files that the requesting user is permitted to access. This restriction is not enforced. The information erroneously provided, however, does not include passwords and only contains log entries and permit entries that apply to the requesting user.
   NV0E251
   NV08133

4.3 REMOTE HOST

1. When the NOS/170 IRHF is started via a DIS job (for debugging IRHF), it does not transmit queue files to NOS/VE. RUNJOBS is unable to acquire the queue file from NOS/170. See Section 3.3 of the SR8 for a method of restarting IRHF.

1. When a bad LOGIN command is present in a batch job, the job disappears and printout with the error is received by the user to report the problem. This occurs when sending a job from NOS to NOS/VE via ROUTE,xx,DC=LP,FC=RH.
   NVOE491

2. Changing passwords on NOS and NOS/VE.

   In NOS 2.2, there are two passwords for each user. One is the NOS interactive job password and the other is the NOS batch job password. The NOS interactive job password is used while working on a terminal. It is also used for NOS/VE login processing - when logging into NOS/VE on an interactive terminal, there is no check made of the NOS/VE password for validation purposes, just the NOS interactive job password. Changing your NOS/VE will affect only your NOS batch password. Your NOS/VE password must match your NOS batch password otherwise Remote Host will not work since it submits batch jobs to NOS.

   Note : When using the SET_LINK_ATTRIBUTES command, the NOS/VE password specified must match the NOS batch job password.

   Note : The selection of passwords for NOS and NOS/VE must conform to the definition as given in the NOS Reference Manual for the NOS password and the NOS/VE Command Interface Reference Manual for the NOS/VE password. You have to take the common parts of both definitions to make a password that works for both NOS and NOS/VE.

   To change your NOS/VE password, enter the SET_PASSWORD command on the 180 side :

   ```
   Setpw oldpw newpw
   ```

   where oldpw is your current NOS/VE password and newpw is the new password.

   To change your NOS batch job password, you will have to submit a NOS batch job. Login to the 170 side and create a file to submit this job. the file should contain the following:

   ```
   JOBCARD.
   USER(usernam,userpw)
   ```

```
CHARGE,*
PASSWORD(userpw,newpw)
END
```

where usernam is your username, userpw is your current NOS batch password, and newpw is your new NOS batch password. Your original NOS batch password is set to your original NOS interactive job passwords. All of the above commands in the file created must be typed in capital letters.

To submit this job, type in :

```
submit,ifn,b.
```

where ifn is the file you created.

In order to change your NOS interactive job password, type in on the 170 side :

```
passwor,oldpw,newpw.
```

where oldpw is your current NOS interactive password and newpw is your new interactive job password.

4.4 PROGRAM_MANAGEMENT

1. Loading files into multiple rings may cause improper ring attributes to be assigned.
   PSR NV0C789

2. The file position specifier is ignored when mentioned on SETPA. For example, specifying SETPA DEBUG_OUTPUT= $LIST.$EOI will cause debug to overwrite the $LIST file from its BOI position (not EOI as expected).
   PSR NVEA036

4.5 PHYSICAL_I/O_(TAPE)

1. Tapes are generally available for use in a single or multi-volume, unlabelled file environment. Copy file of multiple files to a single tape produces undefined results, however.

4.6 BASIC_ACCESS_METHOD

1. The SET_WORKING_CATALOG (SETWC) command has no effect upon files whose names are created within applications. For example:

```
SETWC ajl
execute_task
     task generates file name a
```

   The path name of the file is not .ajl.a; it is $local.a. Only actual parameters of a command are subject to SETWC direction.
   PSR NVEA027

2. The preset_value file attribute is not supported.
   PSR NV0B204

3. File attribute MIN_RECORD_LENGTH is not set properly for record_type F files. MIN_RECORD_LENGTH should be set to MAX_RECORD_LENGTH if it is allowed to default, but it is currently defaulting to 0. This affects SORT/MERGE when the sort key is omitted and the default is taken.
   PSR NV0D398

4. If more than 100 instances of OPEN exist concurrently within a single task, an ame$local_file_limit message is issued. The message indicates erroneously that 162025 local files is the limit which was exceeded. The message should say that only 100 instances of open may exist concurrently in a task.
   PSR NV0D856
   PSR NV0D632

5. Skipping forward/backward by records and partitions does not always report the encounter with boundary conditions with the correct exception condition.
   PSR NV0E044
   PSR NV0D424

6. If a FAP returns abnormal status to AMP$OPEN during an OPEN (new), the file name is still registered within the job and implicitly created permanent files are not purged. Recommend explicitly purging the permanent file.
   PSR NV0E150

7. COPY_FILE does not work on index_sequential files when executed from a task.
   PSR NV0E588

8. File attributes that can be changed cannot be reassigned a NIL value.

PSR NVEA003

9. DETACH_FILE returns FATAL abnormal status when a file is not known and causes a batch job to terminate. Use of the STATUS parameter to ignore the error is recommended.
   PSR NV0D618

10. The SET_FILE_ATTRIBUTE command for an existing file accepts the specification of a preserved attribute even if it conflicts with the present one. No error will be seen and the preserved attribute will not be changed.
    PSR NV0C839
    PSR NV0D617
    PSR NV0D549
    PSR NV0D536
    PSR NVEA036

11. When processing a COBOL READ REVERSE statement, BAM\ fails to set the file position to BOI and as a result the COBOL statement will loop indefinitely.
    PSR NV0E044

12. COPY_FILE does not detect a copy to itself via circular file connections.
    PSR NV0E286

13. Amp$put_partial, with term_option = amc$terminate, at file_position BOI, gives 'unrecovered write error' on a new file. Recommend using amp$put_partial, with term_option = amc$start.
    PSR NV0D666

14. Files implicitly attached are not implicitly detached.
    PSR NV0E322
    PSR NV0E296

15. The CONTAINS_DATA flag delivered by AMP$GET_FILE_ATTRIBUTES is incorrect for a newly created index sequential file. This prevents AAM from working properly when multiple simultaneous OPENs are issued on a file. The work around is to CLOSE and re-OPEN the file immediately after doing the first OPEN/NEW.
    PSR NV0E901

16. An intermittent range error occurs if BAM$SYS_BLK_VARIABLE_REC_FAP during a COPY_FILE command.
    PSR NV0D706

17. Attributes in the NOS/VE file label cannot be altered after creation. This could affect some migrating applications using indexed sequential files because FILE_LIMIT could be updated on C170 but not on C180.

PSR NVEA034

18. The file position specifier on a MERGE directive in the SORT_MERGE directive file isn't ignored.
PSR NV0E415

19. In the DISPLAY_FILE_ATTRIBUTE printout, OPEN_POSITION, MESSAGE_CONTROL, and ERROR_LIMIT do not reflect the values supplied by a previous AMP$FILE call.
PSR NV0E515

20. The attribute array given to AMP$FETCH_ACCESS_INFORMATION will not currently allow AMC$NULL_ATTRIBUTE.
PSR NV0E716

4.7 LOADER

1. If a task adds a library to the job's object library list dynamic loads within that task will be unable to satisfy externals from the newly added library. This especially affects the usability of commands packaged in the job monitor task because libraries added to the object library list by the system prolog cannot be used to satisfy externals of these commands.
PSR NV0E175

2. The default for the program attribute LOAD_MAP_OPTION is set to NONE. This means a load map will not be generated by EXECUTE_TASK unless the LOAD_MAP_OPTION parameter has been previously specified by a SET_PROGRAM_ ATTRIBUTE command or explicitly specified on the EXECUTE_TASK command.

3. The length of large data segments and files cannot exceed 100,000,000 bytes.

4. A job which attempts to use COPY_FILE on an indexed sequential file will abort with an error "unable to find FAP AMP$ADVANCED_ACCESS_METHODS".
PSR NV0C941

4.8 INTERACTIVE

1. If a user's terminal should become disconnected from NOS/VE, the job can be resumed by logging in to the system and entering the command ATTACH_JOB (ATTJ). The terminal will be reconnected to the job, and the job will be put into a pause break state. The TERMINATE_COMMAND and RESUME_COMMAND commands can be used at this

time with the same results as during a pause break. A DETACH_JOB (DETJ) command has also been added to allow a user to detach a job by command. The DISPLAY_JOB_STATUS command will display disconnected and detached jobs. Detached jobs will be terminated after a system defined time has elapsed.

2. In order to use a CDC 721 terminal on NOS/VE, the following command should be entered:

   SET_TERMINAL_ATTRIBUTES TC=T4010

3. Access to NOS/VE (VEIAF) via X25 connection will cause the terminal (and its NOS/VE job) to hang. The hang state can be cleared by entering any TIP command, such as escape-CH. Code to fix the problem in CCP is available and is associated with PSR number CC5A230.

4. NOS/VE R1.0.2 does not support an END-OF-INFORMATION (EOI) facility for indicating to an interactive program that the user wishes to terminate interactive input of data. The input must be ended by a count or by detection of a particular input value.
   PSR NV0E484

5. If an interactive job is using a display type terminal (CRT) and the page width terminal attribute is not set to exactly the page width of the terminal, then overprinting of some lines will occur. To remedy this either set the page width correctly for the particular terminal or set the output_device attribute to printer.

4.9 JOB_MANAGEMENT

If the job reaches time limit, the WHEN clause is repeatedly activated with normal status. If the when contains a CONTINUE, then processing continues in an endless loop.
PSR NV0E368

4.10 INTERSTATE_COMMUNICATION

Changing passwords on NOS and NOS/VE

For NOS 2.2, your NOS batch password has to match your NOS/VE password otherwise Interstate Communications will not work since it submits batch jobs to NOS. Refer to Section 4.3 of the SRB on how to change your passwords.

4.11 DISPLAY_BINARY_LOG

4.11.1 DISPLAY_BINARY_LOG

DISPLAY_BINARY_LOG is a command utility that processes the contents of the global binary logs (statistics, accounting and engineering) into human- readable reports and graphs. There are no pre-packaged reports built into DISPLAY_BINARY_LOG; instead, user sub-commands specify the statistics to be extracted from the log and the types of analysis to be performed. Input to the program may be either an active system log, or a permanent file copy of a log as produced by the Terminate_Log command. The information to be extracted from the log, and the type and format of the reports produced, is specified by sub-commands supplied by the user.

4.11.1.1 The DISPLAY_BINARY_LOG Command

```
PDT disbl_pdt (
   input, i: LIST OF FILE
   type, t: KEY statistic, statistics, account,...
     accounting, engineering = statistic
   output, o: FILE = $OUTPUT
   status )
```

This command, which may be abbrievlated DISBL, enters the user into the command utility environment. Subsequent sub-commands will be prompted for by the string "DISBL/".

The binary log to be processed may be either a currently-active system log, or a copy of a terminated log. If an active log is to be used, the TYPE parameter specifies the desired system global log, and the INPUT parameter is omitted. If a terminated log is to be used, the INPUT parameter specifies the file name, and the TYPE parameter is omitted. If a list of file names is provided for the INPUT parameter, the data will be processed as if the contents of these files are concatenated in the order that the files are specified.

The OUTPUT parameter specifies the file on which the reports generated by DISPLAY_BINARY_LOG are written. All current reports are formatted for a page width of 132 columns.

The first output page generated by DISPLAY_BINARY_LOG lists the input files processed and the beginning and ending dates and times for each file.

4.11.1.2 Definition_Sub-Commands

The user extracts data from the binary log by first specifying one or more groups of statistics, and then specifying one or more metrics (numbers expressing a measurement) to be derived from each group. For example, one group might consist of the end-of-job statistic for each job run on the system, and an associated metric might be the CP time consumed for each job.

Final reports are generated through the use of a set of DISPLAY sub-commands. The particular DISPLAY sub-command used specifies the form of the report (histogram, statistical summary, etc.), and parameters to the sub- command specify the metric or metrics to be displayed, and additional information about the form of the report.

To economize on execution time, DISPLAY_BINARY_LOG processes all sub-commands, up to the QUIT sub-command, before generating any reports. The binary log need be scanned only once, and all reports are generated in parallel. They appear on the final output file, however, in the order in which their sub-commands were issued.

As each group and metric is defined, it is given a user-chosen name, which is subsequently used to refer to it. These names may be up to 31 characters in length, and follow the usual SCL conventions for names. The group or metric identified by a given name need not be defined before the name is used, but must be defined before the QUIT sub-command that terminates the sub-command set.

4.11.1.2.1 THE DEFINE_GROUP SUB-COMMAND

```
PDT defg_pdt (
   group,g: NAME = $REQUIRED
   statistic, s: NAME
   time, t: RANGE OF time_value
   date, d: RANGE OF date_value
   job_predecessor, jp: NAME
   task_predecessor, tp: NAME
   descriptive_data, dd: STRING
   between, b: LIST 2 OF NAME
     )
```

This sub-command defines a statistic group. The GROUP parameter supplies a name for this group, and is required. The other parameters are all optional, and specify conditions which each statistic must satisfy before it will be part of the group. If two or more conditions are specified, a statistic must satisfy all conditions specified before it will be included in the group.

4.0 OPERATING SYSTEM NOTES AND CAUTIONS
4.11.1.2.1 THE DEFINE_GROUP SUB-COMMAND

The STATISTIC parameter specifies a statistic identifier. Statistic identifiers are of the form AANNNNN, where AA is two alphabetic characters, and NNNNN is a decimal number. For example, AV260007 is a statistic emitted by the accounting and validation code at end-of-job.

The TIME and DATE parameters can be used to limit the time period within which statistics will be recognized. The default range of time is from 00:00:00 to 23:59:59. The default range of date is from 00/01/01 to 12/31/99. If only one value is specified for a parameter, instead of a range, then the provided value will be taken as the lower limit of the range, and the default used for the upper limit.

The DESCRIPTIVE_DATA parameter supplies a string which is matched against the contents of the descriptive data field in each statistic. Matching is performed character-by-character to the end of the parameter string. If the descriptive data field in the statistic is longer than the parameter string, the excess characters are ignored.

The TASK_PREDECESSOR parameter is used in combination with other parameters on the DEFINE_GROUP sub-command to relate two or more statistics issued by the same task. The TASK_PREDECESSOR condition is satisfied if the task issuing the statistic also previously issued another statistic, which is a member of the group specified for the TASK_PREDECESSOR parameter.

For example:

```
Define_group compile_tasks statistic = PM230002, ..
  descriptive_data = 'FCP$COMPILE_FORTRAN_SOURCE'
Define_group compile_end_of_task statistic = PM230003, ..
  task_predecessor = compile_tasks t=08:00:00..16:00:00, ..
  d=01/01/83..01/01/83
```

These commands define a group called compile_end_of_task that contains the end of task statistics for all FORTRAN compilation tasks.

The JOB_PREDECESSOR parameter is used in combination with other parameters on the DEFINE_GROUP sub-command to relate two or more statistics issued by the same job. The JOB_PREDECESSOR condition is satisfied if the job issuing the statistic also previously issued another statistic, which is a member of the group specified for the JOB_PREDECESSOR parameter.

The BETWEEN parameter is a future feature. It is currently accepted, but ignored.

4.11.1.2.2 DEFINE_METRIC SUB-COMMAND

4.0 OPERATING SYSTEM NOTES AND CAUTIONS
4.11.1.2.2 DEFINE_METRIC SUB-COMMAND

```
PDT defm_pdt (
   metric, m: NAME = $REQUIRED
   group, g: NAME = $REQUIRED
   scale_factor, sf: INTEGER
   unit, u: STRING
   counter, c: INTEGER 1..63
   expression, e: STRING
      )
```

This sub-command defines a metric. The METRIC parameter specifies a name for this metric, and is required. The GROUP parameter is also required, and specifies the group from whose members this metric is to be derived.

The SCALE_FACTOR parameter is used to adjust the values of the resulting metric in order to make them easier to understand or use. Each metric element is divided by SCALE_FACTOR before being used to generate reports. IF this parameter is not specified on the DEFINE_METRIC sub-command, a value of 1 is used for the scale factor.

UNIT is an optional parameter that can be used to provide a string identifying the measurement unit associated with the values of this metric. If a unit is supplied, it is used to label that metric in all output reports. If the UNIT parameter is not used, it defaults to blanks.

The other parameters specify different ways in which a metric can be derived from the members of the group. One and only one of these parameters can be used in a given DEFINE_METRIC sub-command.

COUNTER specifies the index of one of the 63 possible counters included in a statistic. The contents of the specified counter for each statistic in the group make up the resulting metric.

EXPRESSION specifies an arithmetic expression that is evaluated to yield the value of each metric element. At present, the only expression that is accepted by DISPLAY_BINARY_LOG is '1'. This expression is used to count the number of elements in a metric.

4.11.1.3 Display_Sub-Commands

This group of commands is used to specify reports to be written to the output file. As noted above, all commands are processed, up to the QUIT command, before any reports are actually generated. The reports appear on the final output file in the order in which their sub-commands were issued.

4.11.1.3.1 DISPLAY_SUMMARY SUB-COMMAND

```
PDT diss_pdt (
   metric, m: NAME = $REQUIRED
   title, t: STRING
   )
```

This sub-command requests a statistical summary report. This report includes such information as the mean, maximum and minimum, and total number of elements of the specified metric.

The METRIC parameter specifies the metric to be reported, and is required.

The TITLE parameter may be used to provide a title to be placed at the top of the report page. If omitted, the metric name is used.

4.11.1.3.2 DISPLAY_DISTRIBUTION SUB-COMMAND

```
PDT disd (
   metric, m: NAME = $REQUIRED
   title, t : STRING
   limits, limit, l: (( l1, l2 ), ( l3, l4 )): INTEGER
   display_option, do: KEY max_min_bound, first_max_centered,
                          second_max_centered = max_min_bound
   x_interval, xi: KEY self_adjust, large, medium, small =
                          self_adjust
   )
```

This sub-command requests a distribution plot, which indicates the number of metric elements with each value within the observed range. The report consists of a frame indicating X and Y axises, with the possible values of the metric on the X axis, and the element count on the Y axis. DISPLAY_BINARY_LOG places asterisks within the frame to indicate how many metric elements had a particular value. The report also displays the number of elements within the frame and the number of elements which exceed the frame limits .

The functions of the METRIC and TITLE parameters are the same as in the DISPLAY_SUMMARY sub-command.

The LIMITS parameter can be used to define the frame boundaries. The defaults on the X-axis are the minimum and maximum of the contents in the counter. On the Y-axis, the defaults are the minimum and maximum number of elements among X- axis intervals. First value set defines the X boundaries. Second set defines the Y boundaries. First value in each set defines the lower boundary. '-1' can be used to specify a default boundary.

EXAMPLES:

```
I=(( 50, 1000 )) X lower bound = 50, upper bound = 1000
                  Y boundaries are default.

I=(( -1, 1000 ), X lower bound = default, upper bound = 1000
 ( 10, -1 )) Y lower bound = 10, upper bound = default
```

The DISPLAY_OPTION parameter can be used to select first maximum or second maximum (on Y-axis) portion of the graph. FIRST_MAX_CENTERED, SECOND_MAX_CENTERED and MAX_MIN_BOUND are the keywords. The default of this parameter is MAX_MIN_BOUND.

The X_INTERVAL parameter can be used to determine the number of intervals into which the X-asis of the frame is divided. 3 different sizes are available:

LARGE - X-axis is divided into 11 intervals;
MEDIUM - X-axis is divided into 21 intervals;
SMALL - X-axis is divided into 101 intervals.

The default of this parameter is SELF_ADJUST. When default is used, the X- axis division will be depend on the X-axis limits alone. The rules are:

```
X_upper_bound - X_lower_bound <= 10      11 intervals
X_upper_bound - X_lower_bound <= 500     21 intervals
X_upper_bound - X_lower_bound > 500     101 intervals
```

NOTES:

1. The first and the last intervals on the X axis are one-half the width of the other X axis intervals.

2. The X-axis upper limit as displayed may be slightly higher than the value specified in the subcommand. The rules are:

```
 11 intervals        ( X_upper - X_lower ) modulus 10 = 0
 21 intervals        ( X_upper - X_lower ) modulus 20 = 0
101 intervals        ( X_upper - X_lower ) modulus 100 = 0
```

4.11.1.3.3 DISPLAY_TIME_DISTRIBUTION SUB-COMMAND

4.0 OPERATING SYSTEM NOTES AND CAUTIONS
4.11.1.3.3 DISPLAY_TIME_DISTRIBUTION SUB-COMMAND

```
PDT distd_pdt(
  metric, m: NAME = $REQUIRED
  metric_limits, ml: LIST 2..2 of INTEGER = $REQUIRED
  title, t: STRING
  )
```

This subcommand generates a graph which plots metric values against time. The X-axis represents the time intervals and the Y-axis represents the element counts of the metric. The asterisks placed within each time interval represent the range of element values seen in that interval. The report also generates the number of elements within the frame and the number of elements which exceed the frame limits.

The METRIC parameter specifies the metric name to be reported.

The METRIC_LIMITS parameter specifies the limits to be used for the Y-axis. The limits for the X-axis will come from the date and time parameters given on the DEFINE_GROUP subcommand.

```
example: ml=(0,500)        Y lower bound = 0
                           Y upper bound = 500
```

The TITLE parameter is used to provide a title to be placed on the top of the report page. If omitted, the metric name will be used.

4.11.1.3.4 DISPLAY_DESCRIPTIVE_DATA SUB-COMMAND

```
PDT disdd_pdt (
  group, g:  NAME = $REQUIRED
  title, t:  STRING
  )
```

This sub_command processes the members of a group, rather than a metric. It displays values (up to 256 characters long) for the descriptive data field of each statistic in the group, together with the number of times each value was encountered.

The GROUP parameter specifies the group to be scanned, and is required.

The TITLE parameter may be used to specify a character string to be placed at the top of the first page of the report. If it is not included in the sub-command, then the name of the group is used instead.

4.11.1.3.5 DUMP_GROUP SUB-COMMAND

```
PDT dumg_pdt (
  group, g:  NAME = $REQUIRED
  counter_format, cf: LIST 1..63, 1..2 range of INTEGER 1..63
  title, t:  STRING
    )
```

This sub-command processes the members of a group rather than a metric. Each statistic within the specified group is listed in full. This group dump is particularly useful for debugging group specifications and statistic definitions.

The GROUP parameter specifies the group to be dumped, and is required.

The COUNTER_FORMAT parameter specifies in what base the counters will be displayed. The default base is 10.

```
example: cf=((1..8,10),(9..63,8))
  Counters 1 - 8 will be displayed in base 10.
  Counters 9 - 63 will be displayed in base 8.
```

The TITLE parameter may be used to specify a character string to be placed at the top of the first page of the dump. If it is not included in the DUMP_GROUP sub-command, the name of the group is used instead.

4.11.1.4 GENERATE_GROUP_FILE_Sub-Command

```
PDT gengf_pdt (
  group, g:   NAME = $REQUIRED
  output, o:   FILE = $OUTPUT
  permanent, p: BOOLEAN = FALSE
    )
```

This sub_command selects a group of statistics defined by sub_command DEFINE_GROUP, and writes the statistics to a legible file in a format suited for machine processing.

The GROUP parameter specifies the group to be selected, and is required.

The OUTPUT parameter specifies the local file name which contains the selected statistics. If it is not included in the sub_command, the name $OUTPUT is used as the default file name. The attributes of the output file will be:

```
  FILE_CONTENTS = LEGIBLE
  FILE_ORGANIZATION = SEQUENTIAL
  FILE_PROCESSOR = UNKNOWN
  PAGE_FORMAT = CONTINUOUS
  PAGE_WIDTH = 80
```

(Except for the file $OUTPUT, where the default attributes of $OUTPUT are used.)

If the same file name is used for OUTPUT parameter in several GENGF sub_commands, the order of the group statistics will be the same as the order of the GENGF sub_commands.

The PERMANENT parameter is a boolean value. If PERMANENT = TRUE, then a permanent file with the same name specified in parameter OUTPUT will be created under the catalog $USER. If the permanent file is an existing file, then a new cycle (one higher) will be created. The default value of this parameter is FALSE.

There are some restrictions when parameter PERMANENT = TRUE is specified.

1. Parameter OUTPUT can not be the default value or $OUTPUT. The conflict of the file attributes will cause the sub_command to be rejected.

2. Parameter OUTPUT can not be the same as in any previous GENERATE_GROUP_FILE sub_commands. The local file name conflict will cause the sub_command to be rejected.

3. Parameter OUTPUT can not be the same as an existing local file name. The local file name conflict will cause the DISPLAY_BINARY_LOG to be aborted.

The PERMANENT parameter is not consistent with NOS/VE philosophy, and may be removed in a future update. This will not deprive users of any functionality.

Examples:

```
  CASE 1 ( illegal usage )
    defg g=open_input s=ev1000
    GENGF g=open_input PERMANENT=TRUE

  CASE 2 ( illegal usage )
    defg g=open_input s=ev1000
    defg g=close_input s=ev1001
    GENGF g=open_input OUTPUT=INPUT_GROUP
    GENGF g=close_input OUTPUT=INPUT_GROUP PERMANENT=TRUE
```

```
CASE 3 ( legitimate usage )
  defg g=open_input s=ev1000
  defg g=close_input s=ev1001
  GENGF g=open_input OUTPUT=INPUT_GROUP PERMANENT=TRUE
  GENGF g=close_input OUTPUT=INPUT_GROUP
```

Each statistic is output as one or more text lines. The first line contains the header information, including the number of counters and the size of the descriptive data string. The counters are output next, occupying as many lines as necessary. The descriptive data string is output last, on a single line. The format of each statistic in the output file is shown below:

Line	from Ch.	to Ch.	Contents
1	2	3	statistic_identifier
1	4	9	statistic_code
1	11	17	Julian date(yyyyddd)
1	19	30	time(hh:mm:ss.millisecond)
1	32	36	job_name
1	38	45	global_task_id (nnnn-mmm)
1	47	68	condensing_frequency
1	70	71	number_of_counters
1	73	75	descriptive_data_size
2	1	20	counter 1 (right justified,
2	21	40	counter 2 blank filled)
2	41	60	counter 3
2	61	80	counter 4
3	1	20	counter 5
:	:	:	:
:	:	:	:
n-1	:	:	:
n	2	80	descriptive data

4.11.1.5 The QUIT Sub-Command

This sub-command has no parameters. It ends acceptance of sub-commands, and begins processing of the prior sub-commands to generate reports.

4.11.2 THE ACTIVATE_STATISTICS COMMAND

This command is used to establish and enable one or more statistics to a given global binary log. It may be abbreviated as "ACTS". The PDT is:

```
PDT acts_command (
  statistic, statistics, s:  LIST OF NAME = $REQUIRED
  type, t:  KEY statistic, statistics, accounting, account, ...
    engineering = statistic
  STATUS )
```

The STATISTIC parameter specifies one or more statistics to be established and enabled. Each statistic is specified as two alphabetic characters followed by a decimal number; for example, AV260007. A list of statistics, enclosed in parenthesis as per SCL syntax, may be specified.

The TYPE parameter specifies the global binary log to which the specified statistics are to be enabled. Only one log may be specified in each ACTIVATE_STATISTIC command. If it is desired to send a particular statistic to more than one log, multiple ACTIVATE_STATISTIC commands may be used.

4.11.3 THE DEACTIVATE_STATISTIC COMMAND

This command is used to disable one or more statistics that had previously been enabled to a particular log. All statistics remain established, and if they are enabled to another log, that connection remains undisturbed. The command name may be abbreviated as DEAS.

```
PDT deas_command (
  statistic, statistics, s:  LIST OF NAME = $REQUIRED
  type, t:  KEY statistic, statistics, accounting, account, ...
    engineering = statistic
  STATUS )
```

The STATISTIC parameter specifies one or more statistics to be disabled. Each statistic is specified as two alphabetic characters followed by a decimal number; for example, AV260007. A list of statistics, enclosed in parenthesis as per SCL syntax, may be specified.

The TYPE parameter specifies the global binary log to which the specified statistics are to be disabled. Only one log may be specified in each DEACTIVATE_STATISTIC command. If it is desired to disable a particular statistic from more than one log, multiple DEACTIVATE_STATISTIC commands may be used.

All statistics specified in the DEACTIVATE_STATISTIC command must have previously been enabled to the specified log, either by an ACTIVATE_STATISTIC command or by a program interface call.

4.11.4 ACTIVATE_INTERVAL_STATISTIC

The ACTIVATE_INTERVAL_STATISTIC (ACTIS) command is used to establish and and enable those statistics which are emitted on an interval basis to a given global binary log. This command can only be issued from the system console.

```
PDT actis_command (
  jms_interval, ji: integer 1..999 = 1
  pms_interval, pi: integer 1..999 = 5
  status )
```

The STATISTIC parameter specifies the statistics to be emitted. The only valid statistics of this type currently is the OS210000 and OS210001 statistics.

The JMS_INTERVAL parameter specifies at what time interval the jms statistic (OS210000) should be emitted. The default value is 1 minute.

The PMS_INTERVAL parameter specifies at what time interval the PMS statistic (OS210001) should be emitted. The default value is 5 minutes.

4.11.5 DEACTIVATE_INTERVAL_STATISTIC

The DEACTIVATE_INTERVAL_STATISTIC (DEAIS) command is used to disable interval statistics that have previously been enabled to a particular log. This command only disables the OS210000 and OS210001 statistics. This command can only be issued from the system console.

```
PDT deais_command (
  status )
```

4.11.5 STATISTICS AVAILABLE IN NOS/VE

This is a list of the current statistics available on NOS/VE.

STATISTIC	NAME	COUNTER/DESCRIPTIVE DATA
AV260000	Accounting CP time	C1: Total CP increment C2: Job-mode CP time incr. C3: Monitor-mode CP time incr.
AV260001	Accounting Page Faults	C1: Total page fault incr.

4.0 OPERATING SYSTEM NOTES AND CAUTIONS
4.11.5 STATISTICS AVAILABLE IN NOS/VE

AV260002	Accounting working set	C1: Current W.S. size
AV260003	Accounting Ready Tasks	C1: Current ready task count
AV260006	Accounting Begin Accnt.	DD: 1-31 - account name 32-33 - ', ' 34-64 - project name
AV260007	Accounting End Account	C1: Total job SRU's C2: Total job CP time
AV260008	Accounting SRU's	C1: Total SRU increment
CL170000	Display message command	DD: String specified in command
CL170001	Command names	DD: Name of command; upper case, 31 characters
JM180000	User name	DD: User name; 31 characters
JM180001	Job name	DD: User's job name; 31 chars
JM180002	Job mode	DD: 'INTERACTIVE' or 'BATCH'; 11 characters
JM180003	Job end	C1: CP time - job mode C2: CP time - monitor mode C3: Page faults C4: Page-ins C5: Page-reclaims C6: Page-assigns
PM230000	Task begin	None
PM230001	Task Starting Procedure	DD: Starting procedure name; 31 characters
PM230002	Task name	DD: Task name; 31 characters
PM230003	Task end	C1: CP time - job mode C2: CP time - monitor mode C3: Page faults C4: Page-ins C5: Page-reclaims C6: Page-assigns
PM230004	Loader begin	none

PM230005	Loader end	C1: CP time - job mode C2: CP time - monitor mode C3: Page faults C4: Page-ins C5: Page-reclaims C6: Page-assigns
FC580000	Fortran Compilation	C1: Task time in compiler C2: Monitor time C3: No. of lines compiled C4: No. of statements C5: No. of program units
FC580001	Fortran Program Unit	DD: Program unit name C1: Task time for unit C2: Monitor time for unit C3: No. of lines in unit C4: No. of statements
OS210000	JMS-interval (job-memory interval)	C1: free pages C2: available pages C3: available modified pages C4: wired pages C5: shared pages C6: fixed pages C7: IO error pages C8: job shared pages C9: job working set pages C10: swapped jobs C11: ready tasks C12: interactive class C13: batch class C14: site class 0 C15: site class 1 C16: site class 2 C17: site class 3 C18: maintenance class
OS210001	PMS-interval (paging-monitor interval)	C1: pf available C2: pf available modified C3: pf disk read C4: pf new page C5: pf locked C6: pf IO reject C7: force aggresive aging C8: aggresive aging shared queue C9: aggresive aging job queue C10: aggresive aging failed C11: write aged page C12: mr cycle

C13: mr cycle average duration
C14: mr delay
C15: mr delay average duration
C16: mr wait
C17: mr wait average duration
C18: mr write modified pages
C19: mr write modified pages average duration

4.12 ACCOUNTING/VALIDATION

1. The ADMINISTER_USER utility can be run from a batch job or SCL procedure contrary to what the Family Administrator for NOS/VE Usage Manual says.
   PSRs NVEA045 and NVEA046

2. Great care should be taken using the DELETE_USER subcommand. If more than 20 users are deleted since installation of the system, there is a good chance all users will be lost.
   PSRs NV0D096, NV0E061, NV0E062

5.0 PRODUCT SET NOTES AND CAUTIONS

5.1 CYBIL

1. When using DEBUG to breakpoint on a multi-line statement, the line number supplied to DEBUG must be the last line number of the statement.
   CILA498

2. Run-time error messages, such as RANGE ERROR, don't display the bad value.
   CILA575

3. A large string variable is miscompiled with the wrong size.
   CILA774

4. CYBIL aborts with CG1 error 101, 51, 52, and 38 with no listings.
   PSRs CILA727, CILA785, CILA787, CILA679

5. CYBIL aborts with "ring number zero".
   PSR CILA768

6. CYBIL aborted without any diagnostic.
   PSR CILA755

5.2 COBOL

1. If a USE declarative isn't present, then an I/O error occurring during file input or output causes the job to abort with CB 569260. The text of that message reports "I/O error...", but it doesn't show the underlying reason (for example, "parity error"). Hence, the user should be watchful to include USE declarations for any file where full error messages are desired.

2. The COBOL READ REVERSE statement does not work. If used, it will loop indefinitely on the first record and never take the AT END path. This is the result of BAM failing to set the file position to BOI.
   PSR NV0E044

3. Two consecutive instances of OPEN OUTPUT on the same indexed sequential file will not cause a message warning that the data has been evicted by the second OPEN. The user should refrain from issuing another OPEN OUTPUT once the file has been created.
   PSR AA8A106

5.3 FILE_MANAGEMENT_UTILITY

1. Do not use the file name INPUT in batch mode. It is always a null (empty) file with no data in it. Choose a different name. This affects migration of jobs from NOS, where the name INPUT was the job's input deck. On NOS/VE use the COLLECT_TEXT command to create the input data.

2. When using FMU and Inside Interstate Communication Facility, do not do a CLEAR on the EXECUTE_INTERSTATE COMMAND. This will clear FMU checkpoint files and render FMU inoperable.

3. If the 180 Interstate Communication Facility job aborts, the corresponding 170 FMSLAVE job becomes hung and does not return previously attached files. These FMSLAVE jobs remain at control points and continue to be CPU and memory overhead. Corrective action should include for the user or operator to drop the "hung jobs". They can be identified at a control point as "FMSLAVE,INITIAL", "FMSLAVE,NORMAL", or "FMSLAVE,ABNORM".
   PSR NV0E735

5.4 EQBIBAN

1. When using READ (* ...) in batch jobs, a file containing the desired input must be connected to $INPUT by CREATE_FILE_CONNECTION or equated to INPUT on the FTN program statement or LGO statement.

2. If INPUT or OUTPUT is opened using an OPEN statement, the OPEN statement must be changed to use $INPUT or $OUTPUT.

3. To insure correct listings and improve compilation performance, it is necessary to prevent the listing file and the errors file from being the same file. In batch jobs, both would default to $OUTPUT. To avoid this, set the ERROR parameter on the FORTRAN command to a file reference diffeent from that of the LIST parameter.

4. Utility subprograms CALL CONNEC and CALL DISCON do not work. Their use will cause your program to abort in execution.
   PSR FN8A900

5. RUNTIME_CHECKS=NONE should be specified in addition to OPT=HIGH in the FORTRAN command options when the objective is to achieve the fastest possible execution time for a FORTRAN program. The

compiler generates runtime checking code by default at all OPT levels, unlike FTN5 on the 170 which generates checking code only if explicitly requested. Checking code can have a considerable impact on execution time, since every array element reference with a variable subscript generates a call to a library routine. ONE_TRIP_DO=ON should also be selected to improve execution time if all DO loops in the program have an iteration count of at least one.

6. When compiling at OPT=HIGH, the FORCED_SAVE=ON option should be used for any subprogram which relies on the values of its local variables or arrays being saved between calls to the subprogram but does not use a SAVE statement to retain their definition status. At OPT=HIGH local variables and arrays which are not saved are normally allocated on the runtime stack, so that their values are lost after the execution of a RETURN or END statement in the subprogram. At OPT=LOW/DEBUG local variables and arrays are always allocated to a static memory segment so that saving is always in effect, as it is on the 170 for non-overlayed FTN5 programs compiled at any OPT level.

5.5 ADVANCED_ACCESS_METHODS

A job using an indexed sequential file will fail due to entry point errors from the loader.

The following command executed earlier in the job will correct this:

```
SET_PROGRAM_ATTRIBUTES TERMINATION_ERROR_LEVEL=FATAL
or SETPA TEL=F
```

PSR NV0E276

5.6 SOURCE_CODE_UTILITY

1. The SCU editor command SEARCH_BACKWARD will hang if asked to locate multiple occurrences of a range of text. A terminate break will stop the hung command and allow the user to continue editing. PSR SC8A027

2. An anomaly can occur when some modifications are excluded when a deck is expanded. An example is below:
   Line 1 version 1.
   is introduced by modification A. Modification B replaces this line with
   Line 1 version 2.

Then modification C in turn replaces this line with
Line 1 version 3.
If one expands the deck with modification B excluded both the lines introduced by modifications A and C will appear in the COMPILE file.

5.7 PRODUCT SET = DEBUG

1. When an unselected even occurs in a ring lower than the ring in which it is running, DEBUG incorrectly informs the user that the error occurred in the procedure which called the faulty procedure in the lower ring.

2. DEBUG and CYBIL do not have a run-time interface. Therefore, a program which aborts due to "CYBIL run time error" cannot be properly trapped by DEBUG. Instead, the DEBUG trap occurs in the CYBIL error processor; all dynamic variables (at the point of actual error) have been freed by the time DEBUG gets control.

5.8 ON_LINE_MANUALS

1. The HELP abbreviation, HEL, is broken.

2. On line manuals will be documented in the NOS/VE Installation and Upgrade Reference Manual for Release 1.1.1. Until that time, the installation process has been described earlier in this document in section 3.1.2.6 and the maintenance process will be described in this section.

   Installed into the $SYSTEM_MANUALS subcatalog will be the following:

FORTRAN	(Reference Manual)
FORTRAN_T	(Tutorial)
SORT	(Reference Manual)
SORT_T	(Tutorial)
MIGRATE	(NOS to NOS/VE Migration)
NOS/VE	(Default Manual for EXPLAIN)
COBOL	(Reference Manual)
COBOL_T	(Tutorial)
OCM	(Object Code Mgmt Reference Manual)
SCL	(System Command Language Reference Manual)
SCU	(Source Code Utility Reference Manual)
CONTEXT	(On Line Manual Author's Guide)
MESSAGES	(NOS/VE Diagnostic Messages)

and a MAINTENANCE subcatalog which contains the SCU source library of all the manuals and the binding procedure for creating new "bound" manuals.

The source is being distributed to allow customers the capability of tailoring the manuals for their site. Source libraries contain modifications reflecting the revision level documentation changes. This not only gives the benefit of a "revision packet" to identify new features and changes, but also aids the update process for te site version of manuals.

To tailor a manual for a site:

a. Create a new source library or additional cycle of $system.manuals.maintenance.source_library reflecting the modifications. (Note that each manual is contained in a deck. Decks are grouped by product.)

b. Execute $system.manuals.maintenance.binding_procedure within the $system user or maintenance user number to create new manuals.

```
PROC bind_manuals, binm, bind_manual
  manual, manuals, m: LIST OF NAME or KEY all = all
  base, b: FILE = $SYSTEM.MANUALS.MAINTENANCE.SOURCE_LIBRARY
  catalog, c: FILE = $USER.MANUALS
  status)

where,
  manual = scu deckname(s) of manual(s) desired to be
  created
  base = source library used for compilation
  catalog = resultant catalog for new manual(s)
```

6.0 NOS R2.2

6.0 NOS_R2.2

The BATCHIO printer driver does not correctly handle long lines (over 135 characters). The extra characters are either dropped, overprinted, or cause the page to be spaced erratically.
NS2C121

7.0 FCA LEVELS

7.0 FCA_LEVELS

The folowing tables show the microcode and controlware levels that were used to validate the NOS/VE system on the specified hardware systems supported. These tables show the latest official FCA levels of the hardware and microcode and the necessary modifications (deviations) required to support the NOS/VE system. Tables 1, 2, 3, and 4 correspond to the C170-815, C170-825, C170-835, C170-845, and C170-855 systems, respectively.

A1.0 DEVICE CHARACTERISTICS

A1.0 DEVICE CHARACTERISTICS

A1.1 DISK DEVICES

A1.1.1 PHYSICAL CHARACTERISTICS

	844-4x	885-1x
Spindles/Cabinet	1	2
Cylinders/Spindle	823	843
Tracks/Cylinder	19	40
Sectors/Track	24	32
Bytes (8 bit)/Sector	483	516

A1.1.2 LOGICAL CHARACTERISTICS

A1.1.2.1 Terminology

- Allocation_unit - The quantum of assignment of mass storage space to a file. An allocation_unit is a power-of-two multiple of consecutive DAUs. At present, the allocation_unit is not selectable on a file by file basis; all end-user files default to 16384 byte allocation.

- Device_allocation_unit (DAU) - The quantum of allocation of a mass storage volume (spindle). NOS/VE views a device as an array of DAUs which individually or as a contiguous group (called an allocation_unit) may be assigned to a mass storage file. A DAU consists of an integral number of MAUs; the ratio is device dependent and dependent upon the granularity of allocation NOS/VE can afford to provide for a particular device. A DAU may span tracks but never spans cylinders of a device.

- Large_sector - A controlware capability provided by the 7155 class of mass storage controller which logically groups four physical sectors into one for the purpose of decreasing device driver overhead. The term 'MAU' is preferable to 'large sector' since not all controllers may provide this capability to NOS/VE. Note that a large_sector on an 885 device is composed of four physical sectors each of which contains 516 bytes. The last 33 of these bytes are not transferred when a physical sector is read in small sector mode. Therefore, it is possible that NOS/VE will encounter additional bad-spots on a device previously only accessed in small sector mode. The DL8 diagnostic is provided to verify the data field of a large sector. The controlware and NOS/VE consistently report failure status in terms of physical (small) sector address. The large sector diagnostics such as DL8 and FLD, however, display

large sector address. When using the latter diagnostics, it is recommended that one rely on the sector address in the detailed status.

- Minimum_addressable_unit (MAU) - The quantum of data transfer supported by a PP driver. The MAU is a software concept which is used to normalize the various device sector sizes for the Central Processor. When the CP prepares requests for mass storage devices it thinks in terms of a number of MAUs. The PP driver then breaks down the MAU into physical address and sectoring considerations. If a mass storage device has a sector size which is not a power of two bytes in length (such as the 844-4x), then the MAU will begin at the start of a sector and finish in the midst of the last sector; the number of sectors spanned is device dependent. An MAU may span tracks but never spans cylinders of a device.

- Page_size - A power of two number of bytes ranging from 512 bytes to 65536 bytes. NOS/VE is presently constrained to supporting page sizes of 2048, 4096, 8192 and 16384 bytes. The constraint is a file system constraint related to a)not wanting more than one page per MAU and b) not wanting a page to span tracks.

A1.1.2.2 NOS/VE_Disk_Volume_Allocation

	844-4x	885-1x
Sectors/MAU	5	4
MAU/Track	4.8	8
DAU/Track	2.4	4
DAU/Cylinder	44	160
DAU/Spindle	37035	134880
Bytes/MAU	2048	2048
Bytes/Track (avg.)	9485.4	16384
Bytes/Cylinder	180224	655360
10**6 Bytes/Spindle	151.6	552.46
10**6 Bytes/Segment	2147.5	2147.5

A1.1.2.3 NOS/VE_File_Allocation

Allocation Unit Size: (Consecutive DAUs)	844-4x (bytes)	885-1x (bytes)
1	4096	4096
2	8192	8192
4	16384	16384
8	32768	32768
16	65536	65536
32	131072	131072

A1.0 DEVICE CHARACTERISTICS
A1.1.2.3 NOS/VE File Allocation

64	180224	262144
128	180224	524288
256	180224	655360

A1.2 TAPE_DEVICES

A1.2.1 PHYSICAL CHARACTERISTICS

A1.2.2 LOGICAL CHARACTERISTICS

A2.0 MASS STORAGE VOLUME MANAGEMENT

A2.0 MASS_STORAGE_VOLUME_MANAGEMENT

A2.1 SYSTEM_INSTALLATION_PREPARATION

A2.1.1 PREPARING 844 AND 885 DEVICES

All 844 and 885 volumes are formatted at the factory. Neither 844 nor 885 volumes need to be reformatted to be used by NOS/VE. Volumes used by NOS are interchangeable with NOS/VE without re-formatting. Utility Map processing is identical in all aspects. However, with the exception of the Factory Map and the Utility Map, the data recorded on a volume by NOS cannot be interchanged with NOS/VE and vice versa.

NOS/VE does not honor the CTI deadstart sector content. Therefore it does no good to install CTI or CML on a NOS/VE device nor can one expect this information to remain intact across mixed system use of the device, e.g. first NOS, then NOS/VE and then NOS again. CTI and CML should only be installed on the NOS deadstart device or an alternate device used only by NOS.

Prior to attempting NOS/VE system installation, the following procedures should be performed either on-line using MALET on NOS or off-line using DEMOT. The objective of these procedures is to ensure the correctness of the Utility Map maintained on each device. The INITIALIZE_MS_VOLUME process described later uses the Utility Map to avoid defects (bad-spots) on the disk surface.

A2.1.1.1 System_Deadstart_Device

a. For the NOS/VE system deadstart device it is recommended that the SCAN DISPLAY FLAWS module of the FMU diagnostic be used to display all the sectors which have track or sector flaws set. This display must then be compared to the display of the Utility Map. Any flawed sectors not in the Utility Map must be added to ensure that the installation of the NOS/VE system will go smoothly. This procedure should be expected to take at least 10 minutes. Refer to the discussion of defect management below for further insight as to why the above procedure is recommended.
b. Use the DL8 diagnostic to write/read large sectors on Cylinder 0, Track 0 of an 885 deadstart device. This accomplishes two things: 1) it proves that track 0 can later be written by INITIALIZE_MS_VOLUME (INIMV) 2) it ensures that fictitious unrecovered checksum failures are not encountered by INIMV when performing NOS/VE label searching. Note that it is totally unnecessary to write large sectors on the whole device because NOS/VE never reads a sector that it has not first written (except during INIMV label searching).

A2.1.1.2 Volumes_Other_Than_the_System_Deadstart_Device

The above procedure is strongly recommended for each mass storage device to be used by NOS/VE.

A2.2 INITIALIZE_MS_VOLUME

A2.2.1 INITIALIZATION OF 844 AND 885 VOLUMES

The INITIALIZE_MS_VOLUME (INIMV) subcommand of the Logical Configuration Utility (LCU) is used during the NOS/VE installation process to prepare mass storage volumes for use by NOS/VE. The initialization includes writing a NOS/VE software label on the volume and processing the Utility Map located on the volume itself. Before proceeding with the initialization of the volume, INIMV attempts to read a NOS/VE label from one of three possible locations on the volume: MAUs 1, 2 and 3 are tried in succession. This is done as a protection against operator error which could destroy files on the volume if re-initialized.

If an 885 volume has not been previously initialized by NOS/VE or has not been written in large sector mode by diagnostic software, then an uncorrected checkword failure will be reported for each of the three attempts made by INIMV to read a NOS/VE label. Under the previously stated conditions, the reporting of these 3 failures is expected and should not be a source of concern. The 3 locations read by INIMV, in the order attempted, are:

1. Cylinder 0, Track 0, Sector 0
2. Cylinder 0, Track 0, Sector 4
3. Cylinder 0, Track 0, Sector 8 (10 octal)

Note that for an 844 device, checkword failures are not expected during INIMV label checking.

Once convinced that it is not destroying files by mistake, INIMV proceeds to write a NOS/VE label on the volume. Next it processes the Utility Map recorded on the volume. The location of the Utility Map is device dependent as is the content of the map. However the manner in which the Utility Map is read by the NOS/VE driver is device independent as follows:

The 844 and 885 Utility Maps are written in small sector format by the factory and by diagnostic utilities. The Utility Maps fall somewhere within a NOS/VE MAU (logical sector of 2048 bytes). The driver is given an MAU ordinal by INIMV and the driver attempts to deliver an MAU. This involves reading either 4 or 5 small sectors depending upon the device. Because some of the sectors have been

write-protected by the controller and some have not the driver attempts to read each small sector using the following sequence of commands:

1. Read protected sector (34)
2. Read (4)
3. Read factory data (30)
4. Read utility map (31)

If an address error is returned for a particular sector the next read function is tried for the same sector until all 4 generic read functions have been tried or the read is successful, whichever occurs first. Intervening address error failures are not reported, only the last of the four. Any failure other than an address error detected after any of the four generic reads causes request termination and is reported.

A2.2.1.1 844 Utility Map Processing

The INTIALIZE_MS_VOLUME subcommand uses the Utility Map located at Cylinder 822, Track 0, Sector 2 to flaw defects which have been detected at the factory or by the CE and recorded in the Utility Map.

The Utility Map is an array of 0 .. 161 flaw entries. Each entry documents either a track flaw or a sector flaw. Sector flaws are recorded in the physical (small) sector numbering scheme.

INIMV logically flaws the DAU affected by a sector flaw and the 3 DAUs affected by a track flaw. Note that because of the data mapping of the 844 device a track flaw will also logically flaw from 0 to 4 small sectors on the next track.

To support diagnostic use of the device, INIMV also flaws cylinders 820 .. 822.

A2.2.1.2 885 Utility Map Processing

The INITIALIZE_MS_VOLUME subcommand uses the Utility Map located at Cylinder 841, Track 1, Sector 1 to flaw defects which have been recorded there by the factory or by a CE.

The Utility Map is an array of 0 .. 161 flaw entries. Each entry documents a track flaw.

INIMV logically flaws the 4 DAUs which are affected by a flawed track.

To support diagnostic use of the device, INIMV flaws cylinders 841..842.

A2.3 VOLUME_DEFECT_MANAGEMENT

A2.3.1 RECOMMENDATIONS FOR 844 AND 885 DEVICES

Once NOS/VE has been installed and all of its mass storage volumes have been initialized, it is possible that additional defects will appear on the surface of a volume.

The symptom of this defect is likely to initially be a recovered read/write failure. If the same sector is reported repeatedly, one may wish to take action to avoid the loss of performance incurred by repeated disk driver recovery attempts. The SAVE DATA and RESTORE DATA capabilities of the FMU diagnostic utility may be used to relocate the failing sector. This capability only applies to 885 devices as these devices provide two spare sectors per track for this purpose. The procedure must be performed off-line to NOS/VE. Refer to the discussion of off-line flawing for further information.

If the symptom of the defect is an unrecovered read failure, the recommended procedure is to delete the file to which the failing sector was allocated. The deletion of the file should only be done after repeated attempts to read the file have all failed. It is possible that an unrecovered read failure may actually be caused by an intermittent problem with the mass storage controller or storage device logic. Therefore, repeated read attempts may eventually succeed depending on the cause and duration of the problem. If the file is deleted, NOS/VE will ensure that the failing DAU is not allocated to another file. Refer to the on-line flawing discussion for further information.

If the symptom of the defect is an unrecovered write failure, the recommended procedure is to make a copy of the file. NOS/VE keeps unwritable modified pages of a file in memory until the file is detached by all jobs which may be sharing the file. Therefore it is possible to make a copy of the file without re-encountering the original defect. Once a copy of the file has been made, delete the original file and NOS/VE will ensure that the failing DAU is not allocated to another file.

If an unrecovered read/write failure occurs on a file which is critical to continued NOS/VE system execution, the only recourse is to take NOS/VE off-line, flaw the sector or the track, re-initialize the volume on which the flaw was set and perform an installation deadstart. It will be necessary to restore all permanent files.

A2.4 ONLINE_FLAWING

Once a volume has been initialized using the INITIALIZE_MS_VOLUME

subcommand, NOS/VE pays no further attention to the Utility Map on the volume, unless of course the volume should again be subject to initialization.

NOS/VE does not update the Utility Map under any circumstance. Only the diagnostic utility, FMU, is used for this purpose. Refer to the discussion of off-line flawing below.

A2.4.1 NOS/VE SOFTWARE FLAWING OF UNRECOVERED MEDIA FAILURES

NOS/VE dynamically flaws defects detected on failing read or write operations. Defects are logically flawed in the software allocation tables stored on the individual devices; this is referred to as software-flawing.

When a DAU is software-flawed the file to which the DAU is assigned can continue to be accessed. If the cause of the defect is an intermittent problem subsequent reads of the DAU may not re-encounter the failure. Therefore, NOS/VE's philosophy is to not set sector or track flaws in the Utility Map for unrecovered media failures. However, if the read failure persists and the file is later deleted from the system the bad DAU will not be re-assigned to another file. The same is true for DAUs which are the object of failing writes except the write is not re-attempted.

A2.5 OFF-LINE FLAWING

Only a maintenance person (CE) should perform physical flawing. It is not recommended to set either a track or sector flaw on an 844 device or a track flaw on an 885 device unless the defect prevents continued NOS/VE system execution. Setting such flaws on a device used by NOS/VE will destroy data recorded in the sector or track. If the defect is not critical to NOS/VE system execution then on-line flawing is sufficient. Refer to the discussion of defect management above.

If a particular sector is causing system performance degradation due to repeated successful failure recovery, one may decide to relocate the data; this is an option only on an 885 device.

Each track on an 885 device has two spare sectors. The Utility Map on the device has a defective sector list. If there are already two defective sectors reported for the affected track, no data relocation is possible. If there is a spare sector, the FMU diagnostic utility may be used to relocate the data in the failing sector to the spare sector. The 7155 controlware automatically skips defective sectors during a read or write. The FMU diagnostic utility provides modules to SAVE DATA, FLAW SECTOR and RESTORE DATA. The SAVE DATA writes the track with the

suspicious sector to a maintenance track. Next the FLAW SECTOR module is used to set a sector flaw. In addition to flawing the suspicious sector, FLAW SECTOR automatically rewrites the address field in all higher numbered, non-defective sectors to increase the logical sector number stored in each address field by one. Refer to the discussion of setting sector flaws below. Finally, the RESTORE DATA module is used to copy the original track contents from the maintenance track to the original track.

Flawing can be performed only when NOS/VE is off-line. However, flawing may be performed on-line on NOS using MALET or off-line using DEMOT.

Failures reported to the MDD console have cylinder, track and sector reported in decimal. Refer to the section on MDD failure reporting to determine how to find the octal representation, if necessary.

The FMU package is used to perform flawing. FMU is capable of accepting the flaw address in either octal or decimal. Octal values for cylinder, track and sector are the default. A post radix of D indicates the specified value is in decimal.

A2.5.1 SETTING SECTOR FLAWS ON THE 885-1X DEVICE

The failure data displayed to the MDD console or logging in NOS/VE's Engineering Log provides the address of a defect in terms of cylinder, track and sector. The sector address is logical and not physical.

This distinction can best be explained by an example:

Suppose that track N has no flawed sectors. Then a defect is reported by NOS/VE on this track at sector 0. In this case sector 0 is both the logical and physical address of the defect. Some time later another defect is reported at the same address, i.e. sector 0. This time the physical sector number is 1 because physical sector 0 has already been flawed.

Because the logical address is reported in the failure data, the maintenance person must use FMU to read the Utility Map map to determine the location of any existing defective sectors on the track. Then the logical sector address must be converted to a physical sector address and the sector flaw set based on the physical address. The mapping from logical to physical is as follows:

P=L + D

A2.0 MASS STORAGE VOLUME MANAGEMENT
A2.5.1 SETTING SECTOR FLAWS ON THE 885-1X DEVICE

Where:

P is the physical sector address to be given to FLAW SECTOR module of FMU

L is the logical sector address reported

D is the number of defective sectors already set whose address is less than or equal to L

It is better to set sector flaws than track flaws because the 885-1x does have the two spare sectors per track for this purpose. Therefore up to two sectors per track may be flawed without losing the full track capacity. Once a track flaw is set then NOS/VE will not use the track.

A3.0 DUAL STATE DEVICE MANAGEMENT

A3.0 DUAL_STATE_DEVICE_MANAGEMENT

A4.0 MDD CONSOLE FAILURE REPORTING

A4.0 MDD_CONSOLE_FAILURE_REPORTING

The NOS/VE monitor will format and display failure data on the MDD console for every unrecovered disk failure regardless of the impact the failure has on the system. If the system can continue executing CP tasks, the failure data will also be put in a binary engineering log.

If the system cannot sustain task execution, the failure data will only be displayed on the MDD console; no error logging can occur as no other logging media (save real memory) is available to the monitor.

The following information will be displayed for unrecovered disk failures:

a. Channel number of disk controller (in decimal)
b. Equipment number of disk controller
c. Disk unit number (in decimal)
   844-4x unit numbers are 0 .. 7
   885-1x unit numbers are 32 .. 47
d. Logical operation performed at time of failure
   READ
   WRITE
   WRITE INITIALIZE
   READ FLAW MAP
   DRIVER INITIALIZATION
e. English description of failure symptom if it can be determined
   SOFTWARE FAILURE - (CP/PP interface failure)
   INDETERMINATE - (channel or controller or unit)
   INPUT CHANNEL PARITY
   OUTPUT CHANNEL PARITY
   CONTROLLER FAILURE
   UNIT FAILURE
   FUNCTION TIMEOUT (not RAM-parity induced)
   UNIT RESERVED
   CONTROLLER RESERVED
   SEEK FAILURE
   ERROR IN CHECKWORD
   CONTROLLER RAM PARITY
   INCOMPLETE SECTOR TRANSFER
   UNIT NOT READY
   UNIT OFFLINE OR NOT CABLED (844)
   UNIT READ ONLY SWITCH ON (885)
   CHAN ENABLE SWITCH OFF/UNIT NOT CABLED (885)
   FLAWED TRACK
   FLAWED SECTOR (844)
   SECTOR ADDRESS MISCOMPARE
   CYLINDER ADDRESS MISCOMPARE
   LOST CONTROL WORD

A4.0 MDD CONSOLE FAILURE REPORTING

IOU OUTPUT PARITY
INDETERMINATE OUTPUT PARITY

f. Cylinder, Track and Sector of media failure (in decimal)
g. Octal display of general status (last failure status)
h. Octal display of detailed status (last failure status)

A4.0 MDD CONSOLE FAILURE REPORTING

Note that if the presentation of unit number, cylinder, track and sector in decimal is a problem, the detailed status does contain these values in an octal representation, as follows:

```
844-4x          1110 9 8 7 6 5 4 3 2 1 0
    word 4                 U U U U U U U _(U= unit number)
         5      C C C C C C C C C T T T
         6      T T S S S S S             (Cylinder,Track,Sector)


885-1x          1110 9 8 7 6 5 4 3 2 1 0
    word 4                   U U U U U U _(U= unit number)
         5                   C C C C C C
         6      C C C C T T T T T T T T  (Cylinder,Track,Sector)
         7      S S S S S S S S
```

A5.0 FAILURE LOGGING

A5.0 FAILURE_LOGGING

NOS/VE logs failure data in the ENGINEERING_LOG. Refer to the sections below for specific information on the log entry content.

Each item in the ENGINEERING_LOG is called a statistic. Information follows concerning identification of statistics in this log.

If problems with mass storage or tape peripherals are suspected, the DISPLAY_BINARY_LOG utility may be used to extract a subset of the ENGINEERING_LOG pertinent to the source of the problem, depending upon the degree of problem isolation already done.

A good habit for the site to follow is to use DISPLAY_BINARY_LOG at the time that the ENGINEERING_LOG is terminated via the TERMINATE_LOG command to list the mass storage and tape statistics for the day. IF an intermittent failure should develop, a picture of how the peripheral has performed in the recent past may be invaluable in debugging the problem.

A5.1 STATISTIC_IDENTIFIER

The configuration management statistics will have 'CM' as the identifier.

A5.2 STATISTIC_CODE

The configuration management statistics will be identified by the codes 350000..359999.

A5.3 RELATION_TO_OTHER_STATISTICS

The sfp$emit_system_statistic interface captures some information from the emitting job and task environment and places this in the emitted statistic. The Julian date (yyyyddd), time (hh.mm:ss.ms), job name (system-supplied) and global-task-id (system-supplied) are provided.

At present all the disk-related statistics are emitted from a 'system' job. Therefore, the job name and global-task-id data are of no value. However, the tape-related statistics are emitted from within the environment of the job using the tape hardware elements. Therefore one may use the JM180000 statistic to determine the user name of the job owner and the JM180001 statistic to determine the user-supplied job name of the job which emitted tape-related statistics.

For example, the following command sequence extracts all the 'JM180000' entries in the STATISTIC log to the file LISTING. Then the EDIT_FILE command is used to peruse the output to find a particular entry.

```
DISPLAY_BINARY_LOG T=STATISTIC O=LISTING
DEFINE_GROUP G=USER_NAME S=JM180000
DUMP_GROUP G=USER_NAME
QUIT
EDIT_FILE LISTING
```

A5.4 SAMPLE PROCEDURES FOR DISPLAYING FAILURE DATA

The following SCL procedures are provided as an example of how to use DISPLAY_BINARY_LOG to obtain failure data.

A5.4.1 DISPLAY_DISK_FAILURE_DATA

```
PROC display_disk_failure_data ,disdfd ( ..
    log,i: file =$local.$engineering_log; ..
    output,o: file = $output)
display_binary_log i=$value(log) o=$value(output)
define_group system_action statistic=CM350000
define_group software_failure statistic=CM350001
define_group $7154_failure statistic=CM350002
define_group $7155_failure statistic=CM350003
dump_group group=system_action cf=((1..4,8),(5..7,10))
dump_group group=software_failure cf=((1..4,8),(5..8,10))
dump_group group=$7154_failure ..
  cf=((1..4,8),(5..10,10),(11..16,8),(17,10),(18..60,8))
dump_group group=$7155_failure ..
  cf=((1..4,8),(5..10,10),(11..16,8),(17,10),(18..60,8))
quit

PROCEND display_disk_failure_data
```

A5.4.2 DISPLAY_TAPE_FAILURE_DATA

```
PROC display_tape_failure_data ,distfd ( ..
    log,i: file =$local.$engineering_log; ..
    output,o: file = $output)

display_binary_log i=$value(log) o=$value(output)
define_group group=$7021_failure statistic=CM351002
dump_group group=$7021_failure ..
    cf=((1..4,8),(6..9,10),(10..25,8),..
    (26..28,10),(29,16),(30,10),(31..62,8))
quit

PROCEND display_tape_failure_data
```

A5.5 DISK-RELATED STATISTICS

The statistic codes 350000 .. 350999 will be set aside for disk related configuration management statistics.

The following codes will be initially defined:

350000 - System Action. Documents significant events/actions related to the disk configuration.

350001 - Software Failure Data. Documents software failures detected by NOS/VE software.

350002 - 7154 Disk Controller Failure Data. Documents hardware failures detected by NOS/VE software.

350003 - 7155-1 and 7155-1x Disk Controller Failure Data. Documents hardware failures detected by NOS/VE software.

A5.5.1 DESCRIPTIVE_DATA PORTION OF STATISTIC

The descriptive-data portion of the disk related statistics have the following contents:

'<mf>.<pp>.<channel>.<controller>.<unit>*<vsn>*<severity>*<symptom>'

where <mf> is the name of the mainframe specified when the physical configuration was installed.

where <pp> is the string 'PPn' where n is the PP number in decimal of the PP which performed the i/o operation which is being reported.

where <channel> is the string 'CHANNELn' where n is the channel number over which the i/o request was processed.

where <controller> is the element name of the disk controller used in the failing request.

where <unit> is the element name of the failing disk storage device used in the failing request.

where <vsn> is the recorded-vsn of the disk volume which was the object of the failing request.

where <severity> is the string 'UF' or 'RF' for unrecovered and recovered failure severity, respectively.

where <symptom> is the symptom/action statement provided by the system.

A5.5.2 COUNTERS PORTION OF STATISTIC

For recovered disk failures the initial failure data is logged. For unrecovered disk failures both the initial and final failure data is

logged.

The COUNTERS portion of the disk-related statistics have the following contents:

Note that bit 0 (left-most bit) of the counter word indicates the presence or absence of information in the remainder of the word. If bit 0 is not set, then the contents of the word contain a bonafide value as documented below.

A5.5.2.1 System_Action_=_350000

1. PP number
2. Channel Number of Controller
3. Equipment Number of Controller
4. Physical Unit Number (if action related to a unit)
5. Unit-type (identifies the kind of unit, i.e. product id)
   - 1 - 844-4x
   - 2 - 885-1x
6. Controller-type (identifies the kind of controller)
   - 1 - 7154
   - 2 - 7155-1x
7. System Action Code (tells what the system did)
   - 1 - CONTROLWARE RELOADED
   - 2 - CONTROLLER DOWNED
   - 3 - UNIT DOWNED

A5.5.2.2 Software_Failure_=_350001

1. PP number
2. Channel Number of Controller
3. Equipment Number of Controller
4. Physical Unit Number (if failure related to unit)
5. Unit-type (identifies the kind of unit, i.e. product id)
   - 1 - 844-4x
   - 2 - 885-1x
6. Logical Operation Code (tells what the system was trying to do)
7. Failure Severity
   - 0 - Recovered Failure
   - 1 - Unrecovered Failure
8. Failure Symptom Code (tells what the system thinks is wrong)
   - 1 - Address of PP_Communication_Area not on word boundary.
   - 2 - Reserved field of the PP_Communication_Buffer descriptor is non-zero
   - 3 - Reserved field of the PP_Request_Queue descriptor is non_zero
   - 4 - Logical unit number in Unit_Descriptor not in sequence or out of range
   - 5 - Address of Unit_Interface_Table not on word boundary.

6 - Unit_Descriptor contains invalid channel number
7 - Communication_Buffer length not a multiple of words or not long enough
8 - Reserved field after number_of_units component of the PP_Interface_Table is non-zero
9 - Invalid physical-unit-number in Unit_Descriptor
10 - Logical_unit_number of Unit_Interface_Table not equal to value in Unit_Descriptor
11 - Address of Unit_Communication_Buffer in Unit_Interface_Table not on word boundary
12 - Reserved field of Unit_Communication_Buffer is non-zero
13 - Reserved field of Unit_Request_Queue descriptor is non_zero
14 - Invalid unit-type in Unit_Interface_Table
15 - Unit_Communication_Buffer length not word multiple
16 - Unit_Communication_Buffer length too small
17 - Invalid secondary address in I/O request header
18 - Invalid I/O request command code
19 - Invalid length field in indirect address list
20 - Invalid command sequence

A5.5.2.3 7154_Failure_=_350002

1. PP number
2. Channel Number of Controller
3. Equipment Number of Controller
4. Physical Unit Number (if failure related to unit)
5. Unit-type (identifies the kind of unit, i.e. product id)
   1 - 844-4x
   2 - 885-1x (7155 controller only)
6. Logical Operation Code (tells what the system was trying to do)
   1 - read
   2 - write
   3 - write_initialize (sets unwritten areas to preset_value)
   4 - read_flaw_map
   5 - disk_driver_initialization
7. Failure Severity
   0 - Recovered Failure
   1 - Unrecovered Failure
8. Failure Symptom Code (tells what the system thinks is wrong)
   1 - INDETERMINATE (channel or controller orunit)
   2 - INPUT CHANNEL PARITY - On an input from controller to PP the channel-error-flag was set.
   3 - OUTPUT CHANNEL PARITY - On an output from PP to controller the controller reported a parity error in detailed status but the channel-error-flag was not set.
   4 - CONTROLLER FAILURE - (reported by controller)
   5 - UNIT FAILURE
   6 - FUNCTION TIMEOUT - (controller not responding)

A5.0 FAILURE LOGGING
A5.5.2.3 7154 Failure - 350002

7 - UNIT RESERVED - (unit reserved to opposite access)
During i/o request processing the PP driver attempts to connect to the target unit. If it cannot connect to the unit, this statistic is generated. The driver will terminate the i/o request as a result.
8 - CONTROLLER RESERVED - (to another channel)
During i/o request processing the PP driver attempts to obtain the coupler reservation. If not successful for 10 seconds, this statistic is generated. The driver will continue to try to obtain the coupler reservation until successful.
9 - SEEK FAILURE
10 - ERROR IN CHECKWORD
11 - CONTROLLER RAM PARITY
12 - INCOMPLETE SECTOR TRANSFER
15 - UNIT NOT READY - START switch off, unit not spun up or dropped ready
16 - UNIT OFFLINE OR NOT CABLED - 844 unit has switch on back of unit in offline position or unit not cabled to controller.
17 - UNIT READ ONLY SWITCH ON - 885 unit has write-inhibit switch set
18 - CHAN ENABLE SWITCH OFF/UNIT NOT CABLED - 885 unit has CHAN ENABLE switch off or unit is not cabled to controller
19 - FLAWED TRACK - a sector with a track flaw bit set has been read/written. This may indicate that the Utility Map is wrong or there is a NOS/VE software problem.
20 - FLAWED SECTOR - a flawed sector on an 844 unit has been read/written. This may indicate that the Utility MAP is wrong or there is a NOS/VE software problem.
21 - SECTOR ADDRESS MISCOMPARE
22 - CYLINDER ADDRESS MISCOMPARE
23 - LOST CONTROL WORD
24 - IOU OUTPUT PARITY - On an output from PP to controller both the channel-error-flag and the controller's detailed status indicated a parity error occurred.
25 - INDETERMINATE OUTPUT PARITY - On an output from PP to controller the channel-error-flag was set but there was no error reported by the controller. This may mean there is a problem in the IOU and/or the channel and/or the controller.

9. Request Retry Count
The number of times the PP driver retried the i/o request from the beginning.
10. Sector Retry Count
The number of retries that the PP driver performed on the failing sector on the last attempt to retry the i/o request.
11. Cylinder number of initial seek
12. Track number of initial seek

A5.0 FAILURE LOGGING
A5.5.2.3 7154 Failure - 350002

13. Sector number of initial seek
14. Cylinder number of failure
15. Track number of failure
16. Sector number of failure
17. Residual byte count on incomplete channel transfer
18. Failing Function
    The function that caused the initial recovery attempt. The value is extracted from the initial detailed status if the controller provides status after the failure. On a function timeout, the function reported is the one which was outstanding when the controller hung.
19. General Status of Initial Failure
20. Detailed Status of Initial Failure- Word 1
.
.
.
39. Detailed Status of Initial Failure- Word 20

The following counter words are only present for unrecovered disk failures. They represent the status from the last unsuccessful attempt at recovery.

40. General Status of Last Failure
41. Detailed Status of Last Failure - Word 1
.
.
.
60. Detailed Status of Last Failure - Word 20

A5.5.2.4 7155-1x Failure - 350003

Refer to 7154 for specific content.

A5.6 TAPE-RELATED STATISTICS

The statistic codes 351000 .. 351999 will be set aside for tape related configuration management statistics.
The following codes will be initially defined:

351000 - System Action. Documents significant events/actions related to the tape configuration. This will be supported in NOS/VE R1.1.

351002 - 7021-3x Tape Controller Failure Data. Documents hardware failures detected by NOS/VE software.

The statistics defined above have the following general format:

A5.6.1 DESCRIPTIVE_DATA PORTION OF STATISTIC

The descriptive-data portion of the tape related statistics have the following contents:

'<mf>.<pp>.<channel>.<controller>.<unit>*<vsn>*<severity>*<symptom>'

where <mf> is the name of the mainframe specified when the physical configuration was installed.

where <pp> is the string 'PPn' where n is the PP number in decimal of the PP which performed the i/o operation which is being reported.

where <channel> is the string 'CHANNELn' where n is the channel number over which the i/o request was processed.

where <controller> is the element name of the tape controller used in the failing request.

where <unit> is the element name of the failing tape storage device used in the failing request.

where <vsn> is the external-vsn of the tape volume which was the object of the failing request.

where <severity> is the string 'UF' or 'RF' for unrecovered and recovered failure severity, respectively.

where <symptom> is the symptom/action statement provided by the system.

A5.6.2 COUNTERS PORTION OF STATISTIC

In NOS/VE R1.0.2, only the last failure encountered during an unrecovered i/o operation will be logged. In NOS/VE R1.0.3, the format of the following statistic will change to report the initial failure data for recovered failures and both the initial and final failure data for unrecovered failures.

The Historical and Current Block_Id Windows are only appended if the Failure Symptom Code is 'TAPE MEDIUM FAILURE'.

The general format of the COUNTERS portion of the statistic is as follows:

Note that bit 0 (left-most bit) of the counter word indicates the

presence or absence of information in the remainder of the word. If bit 0 is not set, then the contents of the word contain a bonafide value as documented below.

A5.6.2.1 7021-3x Failures = 351002

1. PP number
2. Channel Number of Controller
3. Equipment Number of Controller
4. Physical Unit Number (if failure related to unit)
5. Unit-type (identifies the kind of unit, i.e. product id)
   - 1 - 679-2
   - 2 - 679-3
   - 3 - 679-4
   - 4 - 679-5
   - 5 - 679-6
   - 6 - 679-7
6. Logical Operation Code (tells what the system was trying to do)
   - 1 - read
   - 2 - write
   - 3 - rewind
   - 4 - unload
   - 5 - clear_reserve
   - 6 - write_tapemark
   - 7 - data_security_erase
   - 8 - forespace
   - 9 - backspace
   - 10 - forespace_tapemark(s)
   - 11 - backspace_tapemark(s)
   - 12 - get_status
   - 13 - TCU_loopback
   - 14 - Unit_loopback_I
   - 15 - Unit_loopback_II
7. Failure Severity
   - 0 - Recovered Failure
   - 1 - Unrecovered Failure
8. Failure Symptom Code (tells what the system thinks is wrong)
   - 1 - INDETERMINATE (channel or controller or unit)
   - 2 - INPUT CHANNEL PARITY - On an input from controller to PP the channel-error-flag was set.
   - 3 - OUTPUT CHANNEL PARITY - On an output from PP to controller the controller reported a parity error in detailed status but the channel-error-flag was not set.
   - 4 - CONTROLLER FAILURE - (reported by controller)
   - 5 - UNIT FAILURE
   - 6 - FUNCTION TIMEOUT - (controller not responding)
   - 7 - UNIT RESERVED - (unit reserved to opposite access)
   - 13 - TAPE MEDIUM FAILURE
   - 14 - ERASE LIMIT EXCEEDED

A5.0 FAILURE LOGGING
A5.6.2.1 7021-3x Failures - 351002

24 - IOU OUTPUT PARITY - On an output from PP to controller both the channel-error-flag and the controller's detailed status indicated a parity error occurred.
25 - INDETERMINATE OUTPUT PARITY - On an output from PP to controller the channel-error-flag was set but there was no error reported by the controller. This may mean there is a problem in the IOU and/or the channel and/or the controller.

9. Retry Count (tells how many times the driver performed retry)
10. General Status - Word 1
11. General Status - Word 2
12. Detailed Status - Word 3
.
.
.
25. Detailed Status - Word 16
26. Block Count From Beginning of Tape - This indicates the displacement from the beginning of the tape at the time the failure occurred.
27. Total Blocks Read - This indicates the total number of blocks read by a job since the tape volume was mounted on this tape transport.
28. Total Blocks Written - This indicates the total number of blocks written by a job since the tape volume was mounted on this tape transport.
29. Requested Format (Describes options selected by driver)
30. Density of Tape Used
31. Historical Block-id Window - Word 1 (latest)
.
.
.
46. Historical Block-id Window - Word 16 (oldest)
47. Current Block-id Window - Word 1 (latest)
.
.
.
62. Current Block-id Window - Word 16 (oldest)

A6.0 FAILURE ANALYSIS

A6.0 FAILURE_ANALYSIS

A7.0 RECONFIGURATION

A7.0 RECONFIGURATION

A7.1 ASSUMPTIONS

1. The reader is already familiar with the NOS/VE installation and deadstart process but is not familiar with what to do in particular failure situations.
2. The system analyst will look at the MDD display for failure information in the event of a system failure. If the system has not failed but peripheral failures are suspected, the system analyst will look at the Engineering Log for failure information.
3. The site has generated a deadstart tape with the specified physical and logical configurations as depicted below. Unless otherwise specified, assume the site has installed NOS/VE, has been running the system, has permanent files on disk and has backup tapes of the permanent files.

A7.2 FAILURES NOT REQUIRING RECONFIGURATION

If the performance of the system is degrading and there is an MDD console, see if unrecovered disk failure(s) have been reported. Alternatively, use the DISPLAY_BINARY_LOG command to determine whether unrecovered disk/tape failures have been reported in the Engineering Log.

The first alternative to consider when a peripheral failure is suspected is to terminate NOS/VE (if still running) and revert to NOS (single-state execution). When NOS/VE is terminated, all devices remain DOWN in the NOS configuration. Therefore MALET may be used on NOS to isolate and repair peripheral hardware problems. If the problem is a hardware logic failure and is repaired, one should be able to perform a continuation deadstart and resume normal operation. If this alternative is not chosen and the failing element is the system deadstart controller or disk drive, any modified data in NOS/VE real memory at the time of the failure will be lost.

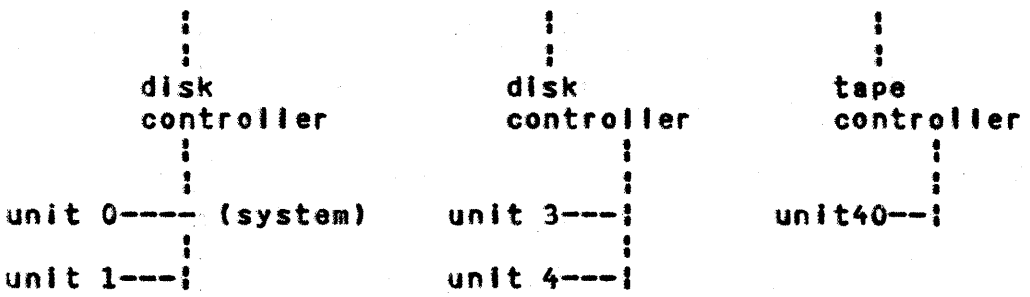
If repair will take too long or files will be lost as a result of the failure, the discussion below is pertinent.

A7.3 CONFIGURATION #1 - SINGLE CONTROLLER

Note that the disk and tape units are only accessible from one controller. This is not a recommended configuration from the standpoints of fault tolerance and system performance. Compare the following scenarios with those of configuraton #2.

channel 0 channel 2 channel 6

A7.0 RECONFIGURATION
A7.3 CONFIGURATION #1 - SINGLE CONTROLLER



A7.3.1 DISK RECONFIGURATION SCENARIOS

Listed below are scenarios for possible disk failures which require a NOS/VE deadstart or which occur while deadstarting NOS/VE. Assume that in each case the desired recovery action is to change the configuration and resume operations as soon as possible. In the discussion below a 'bad HDA' refers to the disk pack mounted on a disk drive; failures such as head-crash or a bad head may cause data to be recorded improperly or to be incapable of being read.

1. Assume 1) a bad HDA on disk #1 (not system device) or 2) a failure in disk drive #1 which will take too long to repair and no spare disk drive. The desired result is to delete disk #1 and continue degraded operation with only 1 disk (#0) on channel #0.
   a. Rebuild the NOS/VE deadstart tape with a new configuration prolog. The new prolog should have the failing device in the physical configuration but not in the logical configuration, i.e. the INSTALL_LOGICAL_CONFIGURATION subcommand of the LOGICAL_CONFIGURATION_UTILITY (LCU) should 'exclude' the failing element. In addition the configuration prolog may need to be changed if it was set up to INITIALIZE_MS_VOLUME and ADD_VOLUME_TO_SET for the failing element; these subcommands must be deleted.
   b. Perform an 'installation deadstart' of NOS/VE.
   c. Reload permanent files. Since you now have only 3 disks, you may need to reload files for a subset of the users and tell the rest of your users to wait for the fourth disk to be repaired.
   d. Resume operation in a degraded mode.
   e. When the failing disk has been repaired, execute the LCU to re-install the logical configuration to 'include' the repaired element. Note that the newly installed logical configuration will not become active until the next continuation deadstart.
   f. When convenient do a TERMINATE_SYSTEM and then a continuation deadstart. This will activate the logical configuration installed above.

g. Execute the LCU to INITIALIZE_MS_VOLUME and ADD_VOLUME_TO_SET for the repaired element.
h. Reload any permanent files not previously reloaded.
i. Continue with normal system operation.

2. Assume 1) a bad HDA on disk #0 (system device) or 2) a failure in disk drive #0 which will take too long to repair and no spare disk drive. The desired result is to delete disk #0 and continue degraded operation with only 1 disk (#1) on channel #0.
   a. Rebuild the NOS/VE deadstart tape on the NOS system. The SET_DEADSTART_DEVICE system-core command must be changed to reflect the type of unit and the unit-number (1) for the new system disk. The configuration prolog must also be rebuilt. The new prolog should have the failing device (0) in the physical configuration but not in the logical configuration, i.e. the INSTALL_LOGICAL_CONFIGURATION subcommand of the LCU should 'exclude' the failing element. In addition the configuration prolog may need to be changed if it was set up to INITIALIZE_MS_VOLUME and ADD_VOLUME_TO_SET for the new system device; these subcommands must be deleted.
   b. Perform an 'installation deadstart' of NOS/VE.
   c. Reload permanent files. Since you now have only 3 disks, you may need to reload files for a subset of the users and tell the rest of your users to wait for the fourth disk to be repaired.
   d. When the failing disk has been repaired, execute the LCU to re-install the logical configuration to 'include' the repaired element. 8Note that the newly installed logical configuration will not become active until the next continuation deadstart.
   e. When convenient do a TERMINATE_SYSTEM and then a continuation deadstart. This will activate the logical configuration installed above.
   f. Execute the LCU to INTITIALIZE_MS_VOLUME and ADD_VOLUME_TO_SET for the repaired element.
   g. Reload any permanent files not previously reloaded.
   h. Continue with normal system operation.
3. Assume 1) a bad HDA on disk #3 (not system device) or 2) a failure in disk drive #3 which will take too long to repair and no spare disk drive. The desired result is to delete disk #3 and continue degraded operation with only 1 disk (#4) on channel #2.
   a. Rebuild the NOS/VE deadstart tape on the NOS system. The configuration prolog will need to be changed. The new prolog should have the failing device in the physical configuration but not in the logical configuration, i.e. the INSTALL_LOGICAL_CONFIGURATION subcommand of the LOGICAL_CONFIGURATION_UTILITY (LCU) should 'exclude' the failing element. In addition the initialization prolog may need to be changed if it was set up to INITIALIZE_MS_VOLUME and ADD_VOLUME_TO_SET for the failing device; these

subcommands must be deleted.

b. Perform an 'installation deadstart' of NOS/VE
c. Reload permanent files. Since you now have only 3 disks, you may need to reload files for a subset of the users and tell the rest of your users to wait for the fourth disk to be repaired.
d. When the failing disk has been repaired, execute the LCU to 'include' the failing element in the installed logical configuration. Note that the newly installed logical configuration will not become active until the next continuation deadstart.
e. When convenient do a TERMINATE_SYSTEM and then a continuation deadstart. This will activate the logical configuration installed above.
f. Execute the LCU to INITIALIZE_MS_VOLUME and ADD_VOLUME_TO_SET for the repaired element.
g. Reload any permanent files not previously reloaded.
h. Continue with normal system operation.

4. Assume the controller on channel #0 requires repair and that the repair will take too long. The desired result is to delete the failing controller from the configuration and continue degraded operation with the single controller on channel #2.
   a. Have a Customer Engineer recable units 0 and 1 to the controller on channel 2.
   b. Rebuild the NOS/VE deadstart tape on the NOS system. A new configuration prolog is required. The new prolog should show units 0 and 1 being connected to the controller on channel 2. The configuration prolog should also be changed to 'exclude' channel 0 from the installed logical configuration. The SET_DEADSTART_CONTROLLER system-core command may need to be changed if the controller on channel #2 has a different product_identification than the one on channel #0.
   c. Do a SETVE to change the deadstart channel number from 0 to 2.
   d. Perform an 'installation deadstart' of NOS/VE.
   e. Reload permanent files.
   f. Continue degraded system operation.
   g. When the failing controller has been repaired, use the INSTALL_PHYSICAL_CONFIGURATION subcommand of the PHYSICAL_CONFIGURATION_UTILITY to re-install the original physical configuration. Note that the newly installed physical configuration will not become active until the next continuation deadstart.
   h. When convenient do a TERMINATE_SYSTEM. Recabling cannot occur while the NOS/VE system is executing.
   i. Recable to match the installed physical configuration.
   j. Perform a continuation deadstart.
   k. Continue with normal system operation.

5. Assume the controller on channel #2 requires repair and the repair will take too long. The desired result is to delete the failing controller from the configuration and continue with the remaining controller on channel #0.
   a. Have a Customer Engineer recable units 3 and 4 to the controller on channel #0.
   b. Rebuild the NOS/VE deadstart tape on the NOS system. A new configuration prolog is required. The new prolog should show units 3 and 4 being connected to the controller on channel #0. The configuration prolog should also be changed to 'exclude' channel 2 from the installed logical configuration.
   c. Perform an 'installation deadstart' of NOS/VE.
   d. Reload permanent files.
   e. Continue degraded system operation.
   f. When the failing controller has been repaired, use the INSTALL_PHYSICAL_CONFIGURATION subcommand of the PHYSICAL_CONFIGURATION_UTILITY to re-install the original physical configuration. Note that the newly installed physical configuration does not become active until the next continuation deadstart.
   g. When convenient do a TERMINATE_SYSTEM. Recabling cannot occur while the NOS/VE system is executing.
   h. Recable to match the installed physical configuration.
   i. Perform a continuation deadstart. This will activate the physical configuration installed above.
   j. Continue with normal system operation.
6. Assume 1) the HDA on disk #1 (not system device) is not damaged and 2) the failure in disk drive #1 will take too long to repair and 3) a spare disk drive. The desired result is to delete the failing device, to move the disk pack from the failing drive to the spare drive, and to continue normal operation.
   a. This assumes that a spare disk drive has been physically configured and that the disk pack (HDA) normally mounted on the drive is not currently used by NOS/VE, i.e. the disk drive is defined in the installed physical configuration but not in the installed logical configuration. If these assumptions are not valid, then follow one of the earlier scenarios concerning a failing disk drive.
   b. Remove the disk pack (HDA) from drive #1 and mount it on the spare drive.
   c. Perform a continuation deadstart and request operator intervention.
   d. From within the LCU use the INSTALL_LOGICAL_CONFIGURATION subcommand to 'exclude' drive #1 and 'include' the spare drive.
   e. Continue normal system operation.
7. Assume 1) the HDA on disk #0 (system device) is not damaged and 2) the failure is in disk drive #0 which will take too long to repair

and 3) a spare disk drive. The desired result is to delete the failing device, to move the disk pack (HDA) from the failing device to the spare device and to continue normal operation. Note that 885-1x (FMD) HDAs can be removed by a CE and moved to a spare drive. Operators can move 844-4x drives themselves.

a. This assumes that a spare disk drive has been physically configured and that the HDA normally mounted on the storage device is not currently used by NOS/VE, i.e. the disk drive is defined in the installed physical configuration but not in the installed logical configuration. If these assumptions are not valid, then follow one of the earlier scenarios concerning a failing disk storage element.
b. Remove the disk pack (HDA) from drive #0 and mount the pack on the spare drive.
c. If the disk pack was moved to channel #2 use the SETVE command to change the deadstart channel from 0 to 2.
d. Perform a continuation deadstart and request operator intervention.
e. If necessary change the SET_DEADSTART_CONTROLLER system-core command to specify the product-identification of the controller.
f. Change the SET_DEADSTART_DEVICE system-core command to identify the unit number of the spare drive.
g. From within the LCU use the INSTALL_LOGICAL_CONFIGURATION subcommand to 'exclude' drive #0 and 'include' the spare drive.
h. Continue normal system operation.

A7.3.2 TAPE RECONFIGURATION SCENARIOS

Listed below are scenarios for possible hardware failures which may occur (or be detected) while using tape peripherals as configured in configuration #1.

a. Assume a failure in the tape controller has caused a NOS/VE interruption. The desired result is to remove the failing tape controller from the NOS/VE configuration to allow diagnosis and repair to occur in the NOS state concurrent with NOS/VE execution.
    a. Perform a continuation deadstart and request operator intervention.
    b. From within the LCU, re-install the logical configuration and 'exclude' channel 6.
    c. Continue operation without the tape controller.
    d. When the tape controller has been repaired, execute the LCU to re-install the logical configuration and 'include' channel #6. Note that the newly installed logical configuration will not become active until the next continuation deadstart.
    e. When convenient do a TERMINATE_SYSTEM and perform a

continuation deadstart to activate the newly installed logical configuration.

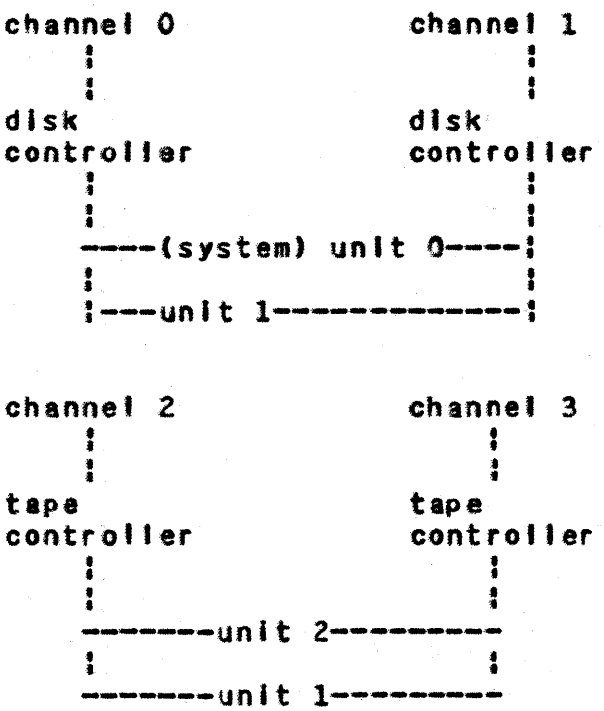
f. Continue with normal operation.

b. Assume a failure in the tape controller which does not cause a NOS/VE interruption. The desired result is to remove the failing controller from the NOS/VE configuration to allow diagnosis and repair of the element to occur in the NOS state concurrent with NOS/VE execution.
   a. Execute the LCU to re-install the logical configuration and 'exclude' channel #6. Note that the newly installed logical configuration does not become effective until the next continuation deadstart.
   b. When convenient do a TERMINATE_SYSTEM and perform a continuation deadstart.
   c. Continue operation without the tape controller.
   d. When the tape controller has been repaired, repeat the process above to 'include' channel #6.

c. Assume that tape unit #40 has failed. Also assume that this failure has not caused a NOS/VE interruption. The desired result is to replace unit #40 with a different tape unit from the NOS configuration, use the NOS state for diagnosis and repair of the failing element and continue NOS/VE execution.
   a. Use the PCU to re-install the physical configuration omitting the failing tape unit and adding the replacement tape unit. Note that the newly installed physical configuration will not become active until the next continuation deadstart.
   b. Use the LCU to re-install the logical configuration. Note this is necessary because the LCU captures some of the information in the installed physical configuration which changed above.
   c. When convenient do a TERMINATE_SYSTEM and a continuation deadstart.
   d. Continue with normal system operation.

d. Assume that tape unit #40 has failed and the failure caused a NOS/VE interruption. The desired result is to replace unit #40 with a different tape unit from the NOS configuration, use the NOS state for diagnosis and repair of the failing element and continue NOS/VE execution.
   a. Perform a continuation deadstart.
   b. After deadstart run the PCU to install a new physical configuration which omits the failing tape unit and defines the replacement.
   c. Do a TERMINATE_SYSTEM and perform another continuation deadstart to activate the newly installed physical configuration.
   d. Continue normal system operation.

A7.4 CONFIGURATION #2 - MULTIPLE CONTROLLERS



Listed below are senarios for possible hardware failures that may require a NOS/VE deadstart or may occur while performing a NOS/VE deadstart. The following scenarios can also be applied to the case where the failure did not cause a NOS/VE interruption. In this case one can execute the LCU to 'exclude' a failing controller, do a TERMINATE_SYSTEM when convenient and then follow the steps below omitting the operator intervention and consequent LCU execution.

A7.4.1 DISK RECONFIGURATION SCENARIOS

The following describes the recommended approach for dealing with the failure of one of the two disk controllers in configuration #2 above.

1. Assume the site has configured to channel #0 unit #0 for the deadstart device and the controller on channel #0 fails. The desired result is to return the controller to the NOS state for diagnosis and repair and to continue degraded operation minus the failing controller.
   a. Execute SETVE command to change the deadstart channel number from 0 to 1.
   b. Perform a continuation deadstart and request operator intervention.

c. Change the SET_DEADSTART_CONTROLLER system-core command if the product_identification of the controller on channel #1 is different than that of the controller on channel #0.
d. From within the LCU use the INSTALL_LOGICAL_CONFIGURATION subcommand to 'exclude' channel #0.
e. Continue degraded system operation.
f. When the failing controller has been repaired, use the LCU to re-install the logical configuration and 'include' channel #0. Note that the newly installed logical configuration will not become active until the next continuation deadstart.
g. When convenient do a TERMINATE_SYSTEM and perform a continuation deadstart to activate the newly installed logical configuration.
h. Continue normal system operation.

2. Assume the controller on channel #1 fails. The desired result is to return the failing controller to the NOS configuration for diagnosis and repair and to continue with degraded system operation minus the failing controller.
   a. Perform a continuation deadstart and ask for operator intervention.
   b. From within the LCU use the INSTALL_LOGICAL_CONFIGURATION subcommand to 'exclude' channel #1.
   c. Continue degraded system operation.
   d. When the failing controller has been repaired, use the LCU to re-install the logical configuration and 'include' channel #1. Note that the newly installed logical configuration will not become active until the next continuation deadstart.
   e. When convenient do a TERMINATE_SYSTEM and perform a continuation deadstart to activate the newly installed logical configuration.

A7.4.2 TAPE RECONFIGURATION SCENARIOS

The following is the recommended approach for dealing with the failure of one of two tape controllers in configuration #2 above.

1. Assume one of the controllers on either channel #2 or channel #3 fails. The desired result is to return the failing tape controller to the NOS state for diagnosis and repair and to continue degraded operation minus the failing controller.
   a. Perform a continuation deadstart and ask for operator intervention.
   b. From within the LCU use the INSTALL_LOGICAL_CONFIGURATION subcommand to 'exclude' the channel with the failing controller.
   c. When the failing controller has been repaired, use the LCU to re-install the logical configuration and 'include' the

channel with the failing controller. Note that the newly installed logical configuration will not become active until the next continuation deadstart.

d. When convenient do a TERMINATE_SYSTEM and perform a continuation deadstart to activate the newly installed logical configuration.

B1.0 PERMANENT FILE MANAGEMENT

B1.0 PERMANENT_FILE_MANAGEMENT

B1.1 BACKUP_PERMANENT_FILE_1_BACPE

The purpose of this command is to initiate execution of the BACKUP_PERMANENT_FILE command utility. This utility creates backup copies of permanent files and catalogs. All catalog information (e.g. file descriptions, access control lists, usage log) and file contents are copied. The RESTORE_PERMANENT_FILE utility restores files and catalogs from their backup copy.

In general, use of the utility is according to the same access rules as other Permanent File Management commands. Users can only backup files for which they are permitted an access_mode that includes read and a share_mode that does not include (append, shorten or modify). The System Operator can backup all files regardless of access permission. A backup will not be done for files that are busy. If a file is busy the file will be skipped. A file is busy if access_mode of (read) and share_mode of (read, execute) cannot be obtained. This means that exclusive access of append, shorten and modify must be obtained.

Files that have been accessed with an implicit attach prior to a backup may be still attached in a mode that causes the file to be skipped. A implicitly attached file should be explicitly detached or the user should logout and login to insure that none of the files are busy that are to be backed up.

In general if errors occur on a subcommand, (such as file busy) a warning message is returned to the user. The user should inspect the list file to determine what errors have occurred.

Utility subcommands are used to direct the backup operation. Capabilities include backup of specific file cycles; backup of all cycles for a file; backup of all files and catalogs under a given catalog; backup of all files and catalogs on the system.

Utility subcommands are provided to allow the installation to do a partial backup of the permanent file base. A partial backup allows the installation to only backup daily files that have been modified on that day, or since the last full backup.

Utility subcommands are provided to allow the installation to backup a subset of all the users. This allows the backup of the permanent file base to be divided into a serveral sets of magnetic tapes. This also allows concurrent backup to occur to more then one tape device.

Utility subcommands are provided to allow the system operator to

B1.0 PERMANENT FILE MANAGEMENT
B1.1 BACKUP_PERMANENT_FILE ¦ BACPF

decrease the size of the permanent file base by backup and/or deletion of files.

Backup commands that use magnetic tape may not executed as part of the system console job because of conflicts in tape assignment.

```
backup_permanent_file backup_file=<file>
                      [list=<file reference>]
                      [status=<status variable>]
```

backup_file ¦ bf: This parameter specifies the file to which backup information will be copied. Backup information resulting from each subcommand issued during this instance of utility execution is copied to this file. It is initially positioned to beginning-of-information.

list ¦ l: This parameter specifies the file to which a summary of the results of executing the backup utility are written. This includes a list of the names of all catalogs and files for which a backup copy was produced. Omission will cause $LIST to be used. The content of the list file may be specified using the SET_LIST_OPTIONS subcommand prior to the backup subcommand. If the SET_LIST_OPTIONS subcommand is ommitted then modification date and time, and size will be displayed.

status: See ERROR HANDLING.

B1.2 BACKUP_PERMANENT_FILE_UTILITY_SUBCOMMANDS

The following subcommands are processed by the BACKUP_PERMANENT_FILE utility.

B1.2.1 BACKUP_ALL_FILES ¦ BACAF

The purpose of this subcommand is to create a complete backup copy of every catalog and permanent file defined in a NOS/VE system. The file specified in the BACKUP_PERMANENT_FILE command to contain the backup information must be associated with magnetic tape. Only a system operator can perform this function.

A previous INCLUDE_USERS directive may be used to select a subrange of users.

Any catalog, file, or cycle specified on a previous EXCLUDE_CATALOG , EXCLUDE_FILE, or EXCLUDE_HIGHEST_CYCLES directive is not backed up.

Previous INCLUDE_CYCLES commands may be used to create a partial backup. A partial backup includes all catalog information and all

B1.0 PERMANENT FILE MANAGEMENT
B1.2.1 BACKUP_ALL_FILES ! BACAF

permanent file cycle data modified after the date specified on the INCLUDE_CYCLES subcommand. The catalog data is used during a restore to select file cycle data from previous partial or full backup operations.

Previous INCLUDE_CYCLES, INCLUDE_VOLUMES, INCLUDE_LARGE_CYCLES, EXCLUDE_HIGHEST_CYCLES, and INCLUDE_EMPTY_CATALOGS along with subsequent DELETE_ALL_FILES subcommands may be used to reduce the files maintained online.

backup_all_files [status=<status variable>]

status: See ERROR HANDLING.

Example: to backup all files. Note that the commands submit a job to avoid console conflicts with tape assignment.

```
Job backup
  request_magnetic_tape file=backup evsn=('BACK1', 'BACK2')  ..
    ring=on t=mt9$6250
  backup_permanent_files backup_file=backup
    backup_all_files
  quit
Jobend
```

B1.2.2 BACKUP_CATALOG ! BACC

This subcommand is intended to be used by the user. The purpose of this subcommand is to create a backup copy of each file cycle and catalog registered relative to a specified catalog. Starting at the specified catalog the complete catalog hierarchy is followed to obtain a backup copy of each file and its associated catalog information.

A system operator is permitted access to all files in a catalog. The owner of the specified catalog and a alternate user may obtain a backup for all files (and associated catalogs) to which they are permitted read access and not required to share the file for (modify, shorten, or append). The system operator or the owner of a catalog may backup a file which contains a password.

The utility skips a file cycle if it is busy, that is if it cannot access the file with an access mode of read and a share mode of (read, execute). Exclusive access of (modify,shorten and append) is required.

Previous EXCLUDE_CATALOG and EXCLUDE_FILE commands allow the user to exclude catalogs and files from the backup.

INCLUDE_CYCLES, INCLUDE_VOLUMES, INCLUDE_LARGE_CYCLES and EXCLUDE_HIGHEST_CYCLES subcommands if executed previous to this subcommand, may have an undesirable effect on the command.

B1.0 PERMANENT FILE MANAGEMENT
B1.2.2 BACKUP_CATALOG ! BACC

```
backup_catalog catalog=<catalog>
               [status=<status variable>]
```

catalog ! c: This parameter specifies the catalog to be backed up. This parameter is required.

status: See ERROR HANDLING.

For example: For a user to backup all files

```
request_magnetic_tape file=backup ring=on evsn='MYDUMP'
backup_permanent_file backup_file=backup list=output
  backup_catalog catalog=$user
quit
```

B1.2.3 BACKUP_FILE ! BACF

This subcommand is intended to be used by a user. The purpose of this subcommand is to create a backup copy of a specified permanent file. If the file name includes a cycle reference only that cycle is copied. If a cycle reference is omitted all cycles of the file are copied. The backup is only done if the user making the request has been permitted read access to the file and not required to share for (modify, shorten, or append).

A permanent file cycle will be skipped if it is busy, that is if it cannot be accessed with a access_mode of (read) and a share_mode of (read, execute). Note: exclusive access for (modify, shorten, and append).

A EXCLUDE_FILE subcommand specifying a cycle number when executed previous to this subcommand may be used to exclude specific cycles from the backup.

INCLUDE_CYCLES, INCLUDE_VOLUMES, INCLUDE_LARGE_CYCLES and EXCLUDE_HIGHEST_CYCLES subcommands if executed previous to this subcommand, may have a undesirable effect on the subcommand.

```
backup_file file=<file>
            [password=<name>]
            [status=<status variable>]
```

file ! f: This parameter specifies the permanent file or permanent file cycle for which a backup copy of each cycle is to be obtained.

password ! pw: This parameter specifies the file password. It must match the password saved for the file. Omission will cause no password to be used.

B1.0 PERMANENT FILE MANAGEMENT
B1.2.3 BACKUP_FILE ! BACF

status: See ERROR HANDLING.

B1.2.4 DELETE_ALL_FILES

The purpose of this subcommand is to delete every permanent file created in a NOS/VE system. Previous INCLUDE_USERS, EXCLUDE_CATALOG, EXCLUDE_FILE, INCLUDE_LARGE_CYCLES, INCLUDE_VOLUMES, EXCLUDE_HIGHEST_CYCLES, and INCLUDE_CYCLES, subcommands may be used to specify a subset of the permanent files to delete. This subcommand may only performed by the system operator.

Empty catalogs may also be deleted depending upon previous INCLUDE_EMPTY_CATALOGS subcommand.

This command is intended to be used following a BACKUP_ALL_FILES subcommand to reduce the size of the permanent file base. The listing generated by the BACKUP_ALL_FILES subcommand should be reviewed to insure that the files selected could be backed up before executing the DELETE_ALL_FILES subcommand.

The file will not be deleted if acess_mode of (read) and a share_mode of (read, execute) cannot be obtained. This means exclusive access for (modify, shorten, and append) is required. The intent of this checking is to insure that a file cannot be deleted that could not be backed up by a previous BACKUP_ALL_FILES subcommand. If these operations are performed in a busy system, changes in file access may result in files being deleted that have not been previously backed up.

delete_all_files [status=<status_variable>]

status: See ERROR HANDLING.

B1.2.5 DELETE_CATALOG_CONTENTS ! DELETE_CATALOG_CONTENT ! DELCC

The purpose of this subcommand is to delete all files in a catalog.

This command is intended to be used by the user. Only the owner of a catalog may successfully use this request to delete a catalog, and to delete files with non_null passwords.

The owner of a catalog and alternate users may use this request to delete all files to which they have been granted control and read access permission, and are not required to share for modify, shorten, and append. That is a share_mode of (read, execute)

If a permanent file cycle is busy the file will be skipped. Busy implies the file could not be accessed with an access_mode of (read) and a share_mode of (read, execute). Note: exclusive access required for

B1.0 PERMANENT FILE MANAGEMENT
B1.2.5 DELETE_CATALOG_CONTENTS ! DELETE_CATALOG_CONTENT ! DELCC

(modify, shorten, and append).

If a file cycle is in use at the time of this subcommand the actual delete is not done until the last user detaches the file.

If a master catalog is specified on this subcommand, the master catalog will not be deleted.

Previous EXCLUDE_CATALOG, EXCLUDE_FILE, EXCLUDE_HIGHEST_CYCLES INCLUDE_LARGE_CYCLES, INCLUDE_CYCLES and INCLUDE_EMPTY_CATALOGS subcommands may be used to specify a subset of the permanent files to be deleted.

This subcommand is useful for the user to manage the users permanent file base. If files are being backed up prior to the DELETE_CATALOG_CONTENTS, the list file should be reviewed to insure the selected files were accessible.

```
delete_catalog_contents catalog=<catalog>
                        [status=<status_variable]
```

catalog ! c: this parameter specifies the catalog name.

status: See ERROR HANDLING.

B1.2.6 DELETE_FILE_CONTENTS ! DELETE_FILE_CONTENT ! DELFC

The purpose of this subcommand is to delete all cycles of a file.

This subcommand is intended to be used by the user. The user must be permitted an access_mode of (control, read) and a share requirement that does not include (modify, append, or shorten).

If a permanent file cycle is busy it will be skipped. Busy implies a access_mode of read and a share_mode of (read, execute) may not be obtained. Note exclusive access for (modify, shorten,and append) is required.

If the file is in use at the time of the delete the actual delete is not done until the last user detaches the file.

Previous EXCLUDE_FILE, EXCLUDE_HIGHEST_CYCLES, INCLUDE_LARGE_CYCLES, and INCLUDE_CYCLES subcommands may be used to specify a subset of the cycles to delete.

```
delete_file_contents file=<permanent_file>
                     [password=<name>]
                     [status=<status variable>]
```

B1.0 PERMANENT FILE MANAGEMENT
B1.2.6 DELETE_FILE_CONTENTS ! DELETE_FILE_CONTENT ! DELFC

file ! f: This parameter specifies the permanent file name. The cycle number, if specified, is ignored.

password ! pw: This parameter specifies the file password of the permanent file to be deleted. This must match the password registered with the file. Omission will cause no password to be used.

status: See ERROR HANDLING.

B1.2.7 INCLUDE_USERS ! INCLUDE_USER ! INCU

This subcommand is provided for the system operator. The purpose of this subcommand is to select a subrange of users or families to be backed up on subsequent BACKUP_ALL_FILES subcommand. This allows the installation to organize their backup magnetic tapes onto several sets of tapes. This also allows multiple backups to occur simultaniously to improve peformance when multiple tape drives and controllers are available. If this subcommand is not used prior to the backup all files of all users will be backed up.

Another benefit of smaller sets of tapes is restartability. If the backup must be restarted for some reason, fewer tapes may need to be rewritten.

The EXCLUDE_CATALOG and EXCLUDE_FILES subcommands take precedence over this command.

```
include_users  users = list range of <catalog> or key all
               [status=<status variable>]
```

users ! user ! u: This parameter specifies an alphabetic subrange of users to be backed up. If only a family name is specified then all users in the family are backed up. Example: :NVE will backup all users in family NVE. If a family name is specified as the low value in a range then all users in the family up to the high value will be backed up. Example: users =:NVE..:NVE.CKB will backup all users up to and including CKB. IF a family name is specified as the high value in a range then all users from low value to the end of the family specified by the high value will be backed up. Up to 20 subranges may be specified. If the keyword All is selected then the effect of any previous INCLUDE_USERS is undone.

status: See ERROR HANDLING.

B1.2.8 SORT_USERS ! SORU

The purpose of this subcommand is to sort family and user names so that the backup will be performed in alphabetical order. This makes it easier to locate user on a backup listing. If this subcommand is not issued users will be backed up in the order they were created.

This subcommand may precede a BACKUP_ALL_FILES or a DELETE_ALL_FILES subcommand.

```
sort_users [alphabetical_order=<boolean>]
           [status=<status variable>]
```

alphabetical_order ! ao: This parameter indicates whether the users are to be sorted or not on the backup. Omission will cause the users to be sorted.

status: See ERROR HANDLING.

B1.2.9 EXCLUDE_CATALOG ! EXCC

The purpose of this subcommand is to direct subsequent backup subcommands to exclude the specified catalog from being backed up or deleted. The catalog will only be excluded if the subsequent backup subcommand is at a higher level in the catalog tree, (closer to the root), thus one may override the effects of this by explicitly backing up the catalog, or subcatalogs below the specified catalog. This subcommand takes precedence over all INCLUDE subcommands.

This subcommand is useful for excluding files that are installed as part of the system initialization process.

```
exclude_catalog catalog=<catalog>
                [status=<status_variable>]
```

catalog ! c: This parameter specifies the permanent file catalog that is to be excluded from being backed up. The catalog must exist.

status: See ERROR HANDLING.

B1.2.10 EXCLUDE_FILE ! EXCF

The purpose of this subcommand is to direct subsequent backup subcommands to exclude the permanent file or permanent file cycle specified from being backed up or deleted. The file or cycle specified will only be excluded if the backup subcommand is at a higher level in

B1.0 PERMANENT FILE MANAGEMENT
B1.2.10 EXCLUDE_FILE ; EXCF

the catalog structure, (closer to the master catalog). The effect of this subcommand may be overridden by explicitly backing up the permanent file or cycle. This subcommand has precedence over all INCLUDE subcommands.

```
exclude_file file=<permanent_file>
             [status=<status_varible>]
```

file ; f: This parameter specifies the permanent file or the permanent file cycle which is to be excluded from being backed up. The file must exist.

status: See ERROR HANDLING.

B1.2.11 INCLUDE_CYCLES ; INCC

The purpose of this subcommand is to direct subsequent backup subcommands to backup or delete cycles based on the creation, last access, last modification, or expiration date of the cycle. All catalog information is still backed up. If a "before" criteria (created_before, accessed_before, modified_before, or expired_before) is selected, then only cycles prior to the given date will be backed up. If an "after" criteria (created_after, accessed_after, modified_after, expired_after) is selected then only cycles after the given date are backed up.

Two INCLUDE_CYCLES directives may be used to specify a window. Thus if one does INCLUDE_CYCLES MODIFIED_AFTER January 1 1983 followed by an INCLUDE_CYCLES MODIFIED_BEFORE January 3 1983 then only cycles modified between the two dates specified will be backed up.

Only one INCLUDE_CYCLES subcommand or two INCLUDE_CYCLES subcommands that specify a window may be used preceding a backup subcommand. CREATED_BEFORE may be used with CREATED_AFTER, ACCESSED_BEFORE may be used with ACCESSED_AFTER, MODIFIED_BEFORE may be used with MODIFIED_AFTER, and EXPIRED_BEFORE may be used with EXPIRED_AFTER. Selection criteria may not be mixed. For example it is not allowable to use CREATED_BEFORE with EXPIRED_BEFORE.

The INCLUDE_CYCLES subcommand is intended to provide a partial backup capability. An INCLUDE_CYCLES subcommand with a MODIFIED_AFTER selection criteria preceding a BACKUP_ALL_FILES subcommand backs up all catalog information. Cycle data is backed up for files that have been modified after the specified date. The INCLUDE_CYCLES MODIFIED_AFTER and MODIFIED_BEFORE subcommands may be used to rebackup a window of partial backup dates.

The INCLUDE_CYCLES with an ACCESSED_BEFORE selection criteria followed by a BACKUP_ALL_FILES and a DELETE_ALL_FILES subcommands may be used to reduce the size of the permanent file base by deleting permanent

B1.0 PERMANENT FILE MANAGEMENT
B1.2.11 INCLUDE_CYCLES ! INCC

file cycles that have not been accessed since the specified date and time.

```
include_cycles selection_criteria=key of created_before,
                                        created_after,
                                        accessed_before, accessed_after,
                                        modified_before, modified_after,
                                        expired_before, expired_after,
                                        all
               month=1 .. 12 ! JANUARY ! FEBRUARY ! MARCH ! APRIL !
                     MAY ! JUNE ! JULY ! AUGUST ! SEPTEMBER !
                     OCTOBER ! NOVEMBER ! DECEMBER
               day=1..31
               year=1983..1999
               [hour:0..23]
               [minute=0..59]
               [second=0..59]
               [millisecond=0..999]
               [status =<status_variable>]
```

SELECTION_CRITERIA ! SC: This parameter specifies the selection, criteria to be used in determining whether the data for a cycle will be backed up. This selection is based on statistics maintained in the permanent file catalog.

created_before: Only cycles created before the given date, time will be backed up/deleted.

created_after: Only cycles created after the given date, time will be backed up/deleted.

accessed_before: Only cycles whose last access was before the given date, time will be backed up/ deleted. accessed_after: Only cycles whose last access was after the given date and time will be backed up/deleted.

modified_before: Only cycles which were last modified prior to the given date and time will be backed up/deleted.

modified_after: Only cycles modified after the given date and time will be backed up/deleted.

expired_before: Only cycles with an expiration date before the given date will be included. If no date is specified the current date is used. The retention period may be selected on the CREATE_FILE or CHANGE_CATALOG_ENTRY commands.

expired_after: Only cycles whose expiration date is after the specified date will be included.

B1.0 PERMANENT FILE MANAGEMENT
B1.2.11 INCLUDE_CYCLES ! INCC

all: This parameter undoes the effect of any previous INCLUDE_CYCLES subcommand. No date or time parameters may be specified with this selection.

month ! m: This parameter specifies the month.

day ! d: This parameter specifies the day of the month.

year ! y: This parameter specifies the year.

The next three parameters specify time. Omission will cause the beginning of the day specified by month, day, and year to be used.

hour ! hr: This parameter specifies hour. Omission will cause the beginning of day to be used.

minute ! min: This specifies minutes. Omission will cause zero to be used.

second ! sec: This specifies seconds. Omission will cause zero to be used.

millisecond ! msec: This specifies milliseconds. Omission will cause zero to be used.

status: See ERROR HANDLING.

Example: INCLUDE_CYCLES MODIFIED_AFTER January 1, 1984

B1.2.12 INCLUDE_EMPTY_CATALOGS ! INCEC

The purpose of this subcommand is to direct subsequent DELETE_ALL_FILES or DELETE_CATALOG_CONTENTS subcommands to delete (or not) any empty catalogs encountered. If this directive is not issued prior to a DELETE_ALL_FILES or DELETE_CATALOG_CONTENTS then empty catalogs will not be deleted.

```
Include_empty_catalogs [delete_catalog= <boolean>]
                       [status=<status_varible>]
```

delete_catalog ! delete_catalogs ! dc: This parameter specifies whether any empty catalog encountered on subsequent DELETE_ALL_FILES or DELETE_CATALOG_CONTENTS should be deleted. Master catalogs will not be deleted. Omission will cause TRUE to be used.

status: See ERROR HANDLING.

B1.2.13 EXCLUDE_HIGHEST_CYCLES ! EXCLUDE_HIGHEST_CYCLE ! EXCHC

The purpose of this subcommand is to exclude the highest cycles of permanent files from being backed up/deleted on subsequent subcommands. This subcommand takes precedence over any INCLUDE_CYCLES subcommand. If this subcommand precede a backup or delete subcommand then the highest cycles of files are excluded.

This command is useful by the installation and the user to select files for backup and deletion.

```
exclude_highest_cycles [number_of_cycles=0..999]
                       [status=<status variable>]
```

number_of_cycles ! noc: This parameter specifies the number of high cycles to exclude. The range of this parameter is 0 .. 999. Omission will cause 3 to be used.

status: See ERROR HANDLING.

B1.2.14 INCLUDE_VOLUMES ! INCLUDE_VOLUME ! INCV

The purpose of this subcommand is to specify that only permanent files residing on the included volumes will be backed up/deleted on subsequent backup subcommands. Any previouly excluded files will not be included regardless of device residence. All volumes specified must be currently active.

This be subcommand may be useful in increasing the amount of available permanent space on a specific volume. Restoring the files will cause the files to be spread over all the volumes.

```
include_volumes recorded_vsns= list of <string> or key all
                [status=<status variable>]
```

recorded_vsns ! recorded_vsn ! rvsn: This parameter specifies the volumes to include. The recorded_vsn specified when the volume was initialized must be provided. The recorded_vsn is a string of up to six alphanumeric characters in uppercase. If keyword ALL is specified, then cycles will not be excluded based on device residence.

status: See ERROR HANDLING.

B1.2.15 INCLUDE_LARGE_CYCLES ! INCLC

The purpose of this subcommand is to specify that subsequent

B1.0 PERMANENT FILE MANAGEMENT
B1.2.15 INCLUDE_LARGE_CYCLES ¦ INCLC

backup/delete subcommands should only include permanent file cycles whose size is greater than or equal to the given size. Any previously excluded cycle will not be backed up/ deleted regardless of cycle size.

This subcommand is useful when trying to reduce the amount of space used on mass storage. Deleting files that are small may cause the user a lot of trouble compared to the amount of space that may be obtained. Using this subcommand the installation may ignore small files when reducing space.

```
include_large_cycles minimum_size=<integer>
                     [status=<status variable>]
```

minimum_size ¦ ms: This parameter specifies the minimum size of files included on subsequent backup/delete subcommands.

status: See ERROR HANDLING.

B1.2.16 SET_LIST_OPTIONS ¦ SET_LIST_OPTION ¦ SETLO

The purpose of this subcommand is to specify the information that will be displayed to the list file on subsequent subcommands. If the length of the display line selected is greater than the page width of the list file, additional lines will be used for the display. This directive does not affect the DISPLAY_BACKUP_FILE subcommand.

```
set_list_options [file_display_options= list of key account, a,
                                        project, p, all, none]
                 [cycle_display_options= list of key
                                         creation_date_time, cdt,
                                         access_date_time, adt,
                                         modification_date_time, mdt,
                                         expiration_date, ed,
                                         access_count, ac, size, s,
                                         recorded_vsn, rvsn,
                                         global_file_name, gfn,
                                         all, none]
                  [display_excluded_items=<boolean>]
                  [status=<status variable>]
```

file_display_options ¦ file_display_option ¦ fdo: This parameter selects data to be displayed with the file name. Data that may be displayed include the account and project. Omission will cause no data to be displayed. The following keywords are used to select displays:

account ¦ a: Display the account name.

project ¦ p: Display the project name.

none: Only the file name is displayed.

all: Both the account and project name are displayed.

cycle_display_options ¦ cycle_display_option ¦ cdo: This parameter selects what information will be displayed for each cycle backed up or restored on subsequent backup and restore subcommands. Additionally the cycle number and if the cycle was excluded will be displayed. The following options are available:

creation_date_time ¦ cdt: The date and time the cycle was created is displayed.

access_date_time ¦ adt: The date and time the cycle was last accessed is displayed.

modification_date_time ¦ mdt: The date and time the cycle was last modified is displayed.

expiration_date ¦ ed: The expiration date of the cycle is displayed.

access_count ¦ ac: The number of accesses to the cycle are displayed.

size ¦ s: The size of the cycle in bytes is displayed.

recorded_vsn ¦ rvsn: The disk volume that the start of the cycle resides on is displayed.

global_file_name ¦ gfn: The internally generated global file name generated when the cycle was created is displayed. This name is not backed up or restored.

none: Only the cycle number is displayed.

all: All options are displayed.

Omission causes modification_date_time and size to be displayed.

display_excluded_item ¦ del: If this is specified as TRUE, then the identification of all catalogs, files or cycles excluded is displayed. If FALSE no excluded items are displayed. Omission causes excluded items to be displayed.

status: See ERROR HANDLING.

B1.0 PERMANENT FILE MANAGEMENT
B1.2.17 QUIT ! QUI

B1.2.17 QUIT ! QUI

The purpose of this subcommand is to terminate the current instance of execution of the backup permanent file utility. Control returns to the processor from which the BACKUP_PERMANENT_FILE command was issued.

quit

B1.3 RESTORE_PERMANENT_FILE ! RESPF

The purpose of this command is to initiate execution of the RESTORE_PERMANENT_FILE command utility. This utility restores permanent files and catalogs from backup copies created by the BACKUP_PERMANENT_FILE utility. The restored information will look as it did when the backup occurred. Utility subcommands are used to direct the restore operation.

```
restore_permanent_file [list=<file reference>]
                       [status=<status variable>]
```

list ! l: This parameter specifies the file to which a summary of the results of executing the restore utility are written. This includes a list of the names of all files and catalogs that were restored. Omission will cause $LIST to be used. Information to be listed may be selected by using the SET_LIST_OPTIONS subcommand. If the SET_LIST_OPTIONS subcommand is not used modification date and time, and size will be displayed for each permanent file cycle.

status: See ERROR HANDLING.

B1.4 RESTORE_PERMANENT_FILE_UTILITY_SUBCOMMANDS

The following subcommands are processed by the RESTORE_PERMANENT_FILE utility.

B1.4.1 RESTORE_ALL_FILES ! RESAF

The purpose of this subcommand is to restore all catalogs and all permanent files on the specified backup file. Backup copies of catalogs and files that do not already exist on the online devices are restored. Catalogs and files that already exist are not altered.

To restore the system when partial backups have been taken, this command is used to restore the last partial backup first. This has the effect of restoring the catalog structure as it was at the time of the

B1.0 PERMANENT FILE MANAGEMENT
B1.4.1 RESTORE_ALL_FILES ! RESAF

last partial backup. File cycle data which is not contained on the last partial backup is restored using the RESTORE_EXCLUDED_FILE_CYCLES subcommand.

```
restore_all_files backup_file=<file>
                  [status=<status variable>]
```

backup_file ! bf: This parameter specifies the file that contains the backup copies of the files and catalogs to be restored. The file is initially positioned to beginning of information.

status: See ERROR HANDLING.

B1.4.2 RESTORE_EXCLUDED_FILE_CYCLES ! RESEFC

The purpose of this subcommand is to restore cycle data for files which are cataloged in the permanent file system but do not have data defined for them. These cycles must have been created by restoring them from a backup file for which cycle data was excluded by the INCLUDE_CYCLES subcommand. The modification date on the backup file must match the modification date in the current catalog for the cycle data to be restored. If a cycle already has data defined, it will not be altered.

This subcommand is to be used in restoring cycle data generated as a result of partial backups. If the permanent file system is backed up by a full backup followed by daily partial backup, then the last partial backup is restored with the RESTORE_ALL_FILES subcommand. RESTORE_ALL_FILES restablishes the catalog structure as it was at the time of the last partial backup. It restores file data that was included on the partial backup file. File data for the rest of the files backed up by the full backup and previous partial backups are restored by using the RESTORE_EXCLUDED_FILE_CYCLES subcommand. This subcommand restores files by matching the modification date in the catalog with the modification date on the backup file. Using the procedure only files that were in existence when the last partial backup was taken are restored.

```
restore_excluded_file_cycles [ file=<permanent_file> !
                               catalog=<catalog>]
                               backup_file=<file>
                               [new_name =
                                    <permanent file> ! <catalog>]
                               [status=<status variable>]
```

The first two parameters specify the catalog or file for which data is to be restored. Only one of the two may be specified. Omission of both will cause all data to be restored. In this case the new name parameter may not be specified.

B1.0 PERMANENT FILE MANAGEMENT
B1.4.1 RESTORE_ALL_FILES ; RESAF

last partial backup. File cycle data which is not contained on the last partial backup is restored using the RESTORE_EXCLUDED_FILE_CYCLES subcommand.

```
restore_all_files backup_file=<file>
                  [status=<status variable>]
```

backup_file ; bf: This parameter specifies the file that contains the backup copies of the files and catalogs to be restored. The file is initially positioned to beginning of information.

status: See ERROR HANDLING.

B1.4.2 RESTORE_EXCLUDED_FILE_CYCLES ; RESEFC

The purpose of this subcommand is to restore cycle data for files which are cataloged in the permanent file system but do not have data defined for them. These cycles must have been created by restoring them from a backup file for which cycle data was excluded by the INCLUDE_CYCLES subcommand. The modification date on the backup file must match the modification date in the current catalog for the cycle data to be restored. If a cycle already has data defined, it will not be altered.

This subcommand is to be used in restoring cycle data generated as a result of partial backups. If the permanent file system is backed up by a full backup followed by daily partial backup, then the last partial backup is restored with the RESTORE_ALL_FILES subcommand. RESTORE_ALL_FILES restablishes the catalog structure as it was at the time of the last partial backup. It restores file data that was included on the partial backup file. File data for the rest of the files backed up by the full backup and previous partial backups are restored by using the RESTORE_EXCLUDED_FILE_CYCLES subcommand. This subcommand restores files by matching the modification date in the catalog with the modification date on the backup file. Using the procedure only files that were in existence when the last partial backup was taken are restored.

```
restore_excluded_file_cycles [ file=<permanent_file> ;
                               catalog=<catalog>]
                               backup_file=<file>
                               [new_name =
                                    <permanent file> ; <catalog>]
                               [status=<status variable>]
```

The first two parameters specify the catalog or file for which data is to be restored. Only one of the two may be specified. Omission of both will cause all data to be restored. In this case the new name parameter may not be specified.

B1.0 PERMANENT FILE MANAGEMENT
B1.4.2 RESTORE_EXCLUDED_FILE_CYCLES ; RESEFC

file ; f: This parameter specifies the permanent file or permanent file cycle on the backup file that is to be restored. If no cycle number is specified then data for all cycles of the file will be restored. Cycle number if specified must be a specific cycle not $HIGH or $LOW.

catalog ; c: This parameter specifies the catalog as identified on the backup file. Data for all cycles in this catalog will be restored if a matching catalog entry exists with no data.

backup_file ; bf: This parameter specifies the file containing the backup information. This file is positioned at the beginning of information.

new_name ; nn ; new_file_name ; nfn ; new_catalog_name ; ncn: This parameter specifies a new name for the catalog file or cycle for which data is to be restored. This may be used if the name on the backup file is different then that in the current permanent file system. Omission will cause the name as it exists on the backup file to be used. If a cycle reference was included on the file parameter but not on the new name parameter then $HIGH is used. In general this parameter should never have to be used by the installation or the user. It intent is to match the parameters on the RESTORE_CATALOG and RESTORE_FILE subcommands allowing change of catalog names to occur when working with partial backups.

B1.4.3 RESTORE_EXISTING_CATALOG ; RESEC

The purpose of this subcommand is to restore an existing online catalog. Backup copies of files and lower level catalogs/files that do not already exist in the specified online catalog are restored. Any item that exists online is not altered.

```
restore_existing_catalog catalog=<catalog>
                         backup_file=<file>
                         [new_catalog_name=<file>]
                         [status=<status variable>]
```

catalog ; c: This parameter specifies the catalog as identified on the backup file that is the base for the restore.

backup_file ; bf: This parameter specifies the file that contains the backup copy of the catalog and its associated files and subcatalogs/files. Information must have previously been written to this file by the BACKUP_FILE, BACKUP_ALL_FILES or BACKUP_CATALOG subcommands. The file is positioned at beginning of information.

B1.0 PERMANENT FILE MANAGEMENT
B1.4.3 RESTORE_EXISTING_CATALOG ¦ RESEC

new_catalog_name ¦ ncn: This parameter specifies the existing online catalog into which the files and subcatalogs on the backup file are restored. Use of this parameter allows the contents of a catalog on the backup file to be restored in an online catalog with a different name. Omission will cause the name as it exists on the backup file to be used.

status: See ERROR HANDLING.

B1.4.4 RESTORE_CATALOG ¦ RESC

The purpose of this subcommand is to restore a catalog that does not currently exist online. Backup copies of the catalog and all its associated files and subcatalogs are restored. Only a system operator or the owner of the resulting online catalog can request this function. The catalog can be restored with the name it had at the time it was backed up or under a new name. The resulting catalog must be unique relative to the catalog in which it is registered. This command cannot be used to restore a user's master catalog.

```
restore_catalog catalog=<catalog>
                backup_file=<file>
                [new_catalog_name=<catalog>]
                [status=<status variable>]
```

catalog ¦ c: This parameter specifies the catalog as identified on the backup file that is the base for the restore.

backup_file ¦ bf: This parameter specifies the file containing backup information. It must have been previously written by the BACKUP_ALL_FILES or BACKUP_CATALOG subcommands. The file is positioned at beginning of information.

new_catalog_name ¦ ncn: This parameter specifies a new name for the catalog being restored. If this new name is not a unique name relative to the catalog in which it is registered an error status is returned. Omission will cause the name as it exists on the backup file to be used.

status: See ERROR HANDLING.

B1.4.5 RESTORE_EXISTING_FILE ¦ RESEF

The purpose of this subcommand is to restore file cycles of an existing online file. All file cycles that exist on the backup file but that do not exist online are restored. Cycles that exist online are not altered. A user with cycle permission to the online file can request this function.

B1.0 PERMANENT FILE MANAGEMENT
B1.4.5 RESTORE_EXISTING_FILE ¦ RESEF

```
restore_existing_file file=<file>
                      backup_file=<file>
                      [password=<name>]
                      [new_file_name=<file>]
                      [status=<status variable>]
```

file ¦ f: This parameter specifies the permanent file as identified on the backup file whose cycles are to be restored. If a cycle reference is included it is ignored.

backup_file ¦ bf: This parameter specifies the file containing backup information. It must have previously been written by the BACKUP_PERMANENT_FILE utility. The file is positioned at beginning of information.

password ¦ pw: This parameter specifies the file password. It must match the existing file password. Omission will cause no password to be used.

new_file_name ¦ nfn: This parameter specifies an existing online file to be restored. Use of this parameter allows a permanent file to be restored under a different file name than exists on the backup. If a cycle reference is included, it is ignored. Omission will cause the name as it exists on the backup file to be used.

status: See ERROR HANDLING.

B1.4.6 RESTORE_FILE ¦ RESF

The purpose of this subcommand is to restore file cycles that do not currently exist online. If the file name includes a cycle reference only that cycle will be restored (at least one file cycle must exist). If a cycle reference is omitted all file cycles will be restored (the file must not already exist). If included it must specify a specific cycle number, $HIGH or $LOW cannot be used. The file can be restored with the name it had at the time it was backed up or under a new name. The resulting name must be unique relative to the catalog in which it is registered. Only a user with cycle permission to the online catalog can restore all file cycles. Only a user with cycle permission to the file can restore an additional file cycle.

```
restore_file file=<file>
             backup_file=<file>
             [password=<name>]
             [new_file_name=<file>]
             [status=<status variable>]
```

file ¦ f: This parameter specifies the permanent file as identified

on the backup file.

backup_file ! bf: This parameter specifies the file containing backup information. It must have been previously written by the BACKUP_ALL_FILES, BACKUP_CATALOG or BACKUP_FILE subcommands. The file is positioned at beginning of information.

password ! pw: This parameter specifies the file password. It must be provided if a specific cycle of an existing online file is being restored. It must match the exsting file password. Omission will cause no password to be used.

new_file_name ! nfn: This parameter specifies a new name for the file being restored. If a cycle reference is not included on this parameter, but was included on the file parameter then $NEXT will be used. If this name isn't a unique name relative to the catalog in which it is registered an error status is returned. Omission will cause the name as it exists on the backup file to be used.

status: See ERROR HANDLING.

B1.4.7 SET_LIST_OPTIONS ! SETLO

This subcommand is available as described in the backup section.

B1.4.8 DISPLAY_BACKUP_FILE ! DISBF

The purpose of this subcommand is to display the contents of a backup file. Information is displayed about catalogs, files, and cycles which have been backed up onto the specified backup file. The caller may optionally select to display header information maintained on the backup file, and may also direct the subcommand to attempt reading of all data on the backup file. The caller may compare this result with the backup listing to verify that everything backed up may be read successfully.

This subcommand is useful to the installation and the user to determine exactly what information is contained on a backup file.

```
display_backup_file backup_file=<file>
                    [display_option= key identifier, i,
                                     descriptor, d, read_data, rd]
                    [number=<integer> or key all]
                    [status=<status variable>]
```

backup_file ! bf: This parameter specifies the file that contains the backup copies of the files and catalogs previously backed

up by a backup utility subcommand.

display_option ! do: This parameter specifies the level of information to be displayed. Valid specifications are:

identifier ! i: Selects a display of the name and type of each entry on the specified backup file.

descriptor ! d: Selects a display that includes: description of the record headers maintained on the backup file, the version of the backup permanent files utilities producing the backup file, The date and time the backup file was written, as well as the backup_permanent_file subcommand producing the backup file, and for each cycle of each file on the backup file the following: cycle number, usage count, creation date and time, last access date and time, last modification date and time, expiration date, and cycle size.

read_data ! rd: In addition to the descriptor display an attempt is made to read all data for each cycle on the backup file. The listing reports whether the data is read without error. No attempt is made to verify the data with the original file backed up.

Omission will cause identifier to be used.

number ! n: This parameter selects the number of catalogs, files, or cycles from the beginning of the backup file, for which information is to be displayed. If this parameter is omitted, or ALL is specified, then all entries on the backup file are displayed.

status: See ERROR HANDLING.

B1.4.9 QUIT ! QUI

The purpose of this subcommand is to terminate the current instance of execution of the restore permanent file utility. Control returns to the processor from which the RESTORE_PERMANENT_FILE command was issued.

quit

Table of Contents

```
PAGE LINE CARD ERRORS/SOURCE AND COMMANDS
  30   20 1474  Err       \*asis
  30   20 1475  Err        ?
  30   20 1475  Err       **PREVIOUS COMMAND UNRECOGNIZED
  31   33 1533  Err       \*asis
  31   33 1534  Err        ?
  31   33 1534  Err       **PREVIOUS COMMAND UNRECOGNIZED
  32   37 1592  Err       \-
  32   37 1593  Err         ?
  32   37 1593  Err       **NEST DOES NOT EXIST
  32   48 1603  Err       \*asis
  32   48 1604  Err        ?
  32   48 1604  Err       **PREVIOUS COMMAND UNRECOGNIZED
  34    3 1656  Err       \*asis
  34    3 1657  Err        ?
  34    3 1657  Err       **PREVIOUS COMMAND UNRECOGNIZED
  34   21 1678  Err       \*asis
  34   21 1679  Err        ?
  34   21 1679  Err       **PREVIOUS COMMAND UNRECOGNIZED
  35    2 1711  Err       \*asis
  35    2 1712  Err        ?
  35    2 1712  Err       **PREVIOUS COMMAND UNRECOGNIZED
  35   19 1732  Err       \*asis
  35   19 1733  Err        ?
  35   19 1733  Err       **PREVIOUS COMMAND UNRECOGNIZED
  65   16 2930  Err       \-
  65   16 2931  Err         ?
  65   16 2931  Err       **NEST DOES NOT EXIST
```