SOFTWARE RELEASE BULLETIN
NOS/VE (CYCLE 17)

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.

LAST UPDATED FEBRUARY 9, 1984

- COMPANY PRIVATE -

1.0 INTRODUCTION

1) The NOS/VE Cycle 17 system consists of the operating system and product set features outlined in the NOS/VE 1.0 Software Availability Bulletin (SAB).

   The NOS/VE Cycle 17 system must be installed using the release materials. (See the NOS/VE 1.0 SAB, pages 11 and 12, for the list of the release materials.) NOS V2.2 Level 596 Operating System and Level 587 for the Common Products must be installed on the computer before installing NOS/VE Cycle 17 to complete the dual state system configuration.

2) It is necessary to understand this document, the NOS/VE 1.0 SAB, the NOS/VE Installation and Upgrade Manual (60463913), and the NOS 2.2 Level 596 Operating System and Level 587 for the Common Products and related documents before installing NOS/VE Cycle 17.

3) After installing NOS V2.2 Level 596 Operating System and Level 587 for the Common Products system and the NOS/VE Cycle 17 system, the NOS and NOS/VE systems may be deadstarted and placed into operation.

4) PSR numbers are included with some of the items in this document. These PSRs are outstanding as of the date of the release.

2.0 FEATURES AND PRODUCTS

A complete description of Operating System and Product Set features may be found in the NOS/VE 1.0 Software Availability Bulletin. Please refer to that document for NOS/VE Cycle 17 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 587.

3.0 INSTALLATION AND OPERATIONS NOTES

3.1 Installation

3.1.1 NOS Installation

1) 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

2) 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 NOS/VE Installation

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.

2) 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 BACPF 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 BACPF. 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:

   1. BACPF all NVE catalogs and files (except uuu.$FAMILY_USERS) to tape (uuu is the FAMILY_USER Administrator).
   2. BACPF the file uuu.$FAMILY_USERS.
      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:

```
TASK RING=3
  COPY_FILE input=alpha output=omega
```

TASKEND
causes the COPY_FILE command to execute in ring 3.
NOTE:
The task command can be specified only in an interactive job or in a batch job running under $SYSTEM.

3. Perform an Installation Deadstart of NOS/VE, but enter the command DETF VALIDATE/OFAMILY/ONVE prior to entering the NEXTDSTAPE response to the request for the first product tape. This command will disable automated validation.
4. Restore the $FAMILY_USERS file that was backed up in step 2. (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.)
5. Restore the remaining NVE family files which were backed up in step 1.
6. Execute the Create_Family command from the operator console PSR NV00079

3) 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.

4) The NOS User Name creation described in the NOS/VE Installation and Upgrade Manual impiles that the default MODVAL DS value is adequate. DS=5 is the appropriate value.

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

6) Online manuals are released on a NOS/VE Backup_Permanent_Files tape at this release. This is an interim process that will be replaced by a process being generated by Publications and Integration. To installed the online manuals, you must reload the files on this tape into the $SYSTEM.MANUALS subcatalogs of the $SYSTEM family. Create the following NOS/VE job to accomplished this task:

```
JOB
request_magnetic_tape file=backup type=mt9$1600 ring=no evsn=nve007

WHEN any_fault DO
```

```
  display_value osv$status output=$response
  request_operator_action message=' RESTORE FAILED - CHECK TAPE'
  logout
WHENEND

RESTORE_PERMANENT_FILES
  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) Upgrade installation specifying LOADFILES (through SET_SYSTEM_ATTRIBUTES) but not MULTD causes the deadstart version of OCU rather than the standard version of OCU to be generated. The $HIGH cycle must be DELF'd after the system is up.

8) The operating system requires assistance at the console for remote installation and remote maintenance. The installation and maintenance procedures are being modified to reduce the need for assistance at the console but until Remote Deadstart is available, this need will not be totally eliminated.

9) Great care should be taken using the DELETE_USER subcommand of the ADMINISTER_USER utility. If more than 20 users are deleted since installation of the system, there is a good chance ALL USERS WILL BE LOST.
PSRs NV0D896, NV0E061, NV0E062

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.

2) To successfully terminate a job that has an action message posted to assign a tape, the sequence must be as follows:

```
K.ASSIGN_DEVICE AI=nnnn STOP
K.TERMINATE_JOB ...
```

Failure to assign the device prior to terminating the job will cause the job to hang.
PSR NV0E337

3.3 Operator Commands

1) Restarting Up 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 PASSON.

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

```
K.DEACTIVATE_SYSTEM_TASKS TASK_NAMES=RHINPUT
K.DEACTIVATE_SYSTEM_TASKS TASK_NAMES=RHOUTPUT
```

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

```
K.ACTIVATE_SYSTEM_TASKS TASK_NAMES=RHINPUT
K.ACTIVATE_SYSTEM_TASKS TASK_NAMES=RHOUTPUT
```

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

```
K.DEACTIVATE_SYSTEM_TASKS TASK_NAMES=IFEXEC
```

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

```
K.ACTIVATE_SYSTEM_TASKS TASK_NAME=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 3777778.

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.

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.

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 NVOE524

3.6 Physical I/O (Tape)

1) Tape support is limited to and intended for use only by the permanent file backup and restore utility. Use of tape I/O for any other purpose may produce undefined

results.

2) When preparing to mount a tape to be read by NOS/VE, the heads on the drive should first be cleaned. This is recommended before each tape mount due to the currently incomplete implementetion of error recovery.

3.7 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

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 NV0D753

2) The $PROGRAM function does not return the open position designator for the LOAD_MAP file.
   PSR NV0D483

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 NV0D871, PSR NV0D888

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 NV0D609

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 defalt) 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 c nges did not make it into the manuals.

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).

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> <,!sp>
                           <string expr> <)>

<char set> ::= <string expr>

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

4.1.2.5 $SCAN_NOTANY

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.
   NVOE251
   NV08133

2) Great care should be taken using the DELETE_USER subcommand of the ADMINISTER_USER utility. If more than 20 users are deleted since installation of the system, there is a good chance ALL USERS WILL BE LOST.
   PSRs NV0D896, NV0E061, NV0E062

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.
   PSR NV0E292

   A method to work around the problem is to start IRHF with an unused "subsystem". An example would be to create the following PROC within the SYSTEMX catalog:

```
.PROC,"subsystem"IRHF
COMMON,SYSTEM.
GTR,SYSTEM,NVELIB,U.ULIB/NVELIB
UNLOAD,SYSTEM.
RUNJOBS(IRHF)
REVERT.
```

   The following command could then be given to DSD.
   "subsystem" IRHF

2) 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

3) Change information is not transmitted from NOS/VE side to the NOS side.
PSR NV00019

4) 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,*.
PASSWOR(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,lfn,b.
```

where lfn 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 NVOC789

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) Tape support is limited to and intended for use only by the permanent file backup and restore utility. Use of tape I/O for any other purpose may produce undefined results.

2) When preparing to mount a tape to be read by NOS/VE, the heads on the drive should first be cleaned. This is recommended before each tape mount due to the currently incomplete implementetion of error recovery.

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 with 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

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 E CUTE_TASK command.

3) The length of large data segments cannot exceed 100,000,000 bytes (i.e., 2**27).

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 e 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 a 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 NV0E387

4.10 Interstate Communication

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 Interstate Communications 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,*.
PASSWOR(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,lfn,b.
```

where lfn 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.

4.11 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 abbrieviated 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.

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

```
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 specified a default boundary.

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

l=(( -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

```
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_STATISTIC 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 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) commmand 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 )
```

4.11.6 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.
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

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

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) Cybil prohibits the user from defining an integer greater than 48 bits. This has been fixed in the Cycle 18 compiler version 83333.
PSR CILA609, PSR CILA515

3) The 256th character of an SCL string variable cannot be set to any character other than a space. This is corrected in CYBIL 83333 for Cycle 18.
PSR CILA634

5.2 COBOL

1) 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

2) 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 TAPE I/O

1) Tape support is limited to and intended for use by the permanent file backup and restore utility. Use of tape I/O for any other purpose may produce undefined results.

2) When preparing to mount a tape to be read by NOS/VE, the heads on the drive should first be cleaned. This is recommended before each tape mount due to the currently incomplete implementetion of error recovery.

5.4 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.

5.5 FORTRAN

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 different from that of the LIST parameter.

5.6 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 NV0C941, NV0E276

5.8 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.

5.9 Product Set = DEBUG

If a job calls a procedure in a lower-numbered ring and that procedure causes an error, then Debug does not give a proper indication of the problem. Debug reports that PM INVALID SEGMENT has occurred in the user's program at the line which originally called the (faulty) procedure. This situation can occur when a CYBIL program calls an OS procedure and the OS procedure does something wrong.

5.10 On Line Manuals

Help (also known as EXPLAIN_MESSAGE) does not exist in the system.

6.0_NOS_B2.2

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

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.

APPENDIX A NOS/VE PERIPHERAL MAINTENANCE SUPPORT

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

A1.0 DEVICE CHARACTERISTICS
A1.1.2.2 NOS/VE Disk Volume Allocation

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
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.

A2.0 MASS STORAGE VOLUME MANAGEMENT
A2.2.1 INITIALIZATION OF 844 AND 885 VOLUMES

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

A2.0 MASS STORAGE VOLUME MANAGEMENT
A2.2.1.1 844 Utility Map Processing

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 t<<<<<<

SOFTWARE RELEASE BULLETIN

84/02/10

NOS/VE (CYCLE 17)

Table of Contents