ADMINISTER_LOCK_FILE 4

ADMINISTER_LOCK_FILE Command

Purpose Begins an ADMINISTER_LOCK_FILE utility session.

Format

ADMINISTER_LOCK_FILE or
ADMLF
STATUS=status variable

Remarks For more information, see the NOS/VE Advanced File Management manual.

CLEAR_LOCK_FILE_CONNECTION ADMLF Subcommand

Purpose Deletes the connection between a lock file and a keyed file.

Format

CLEAR_LOCK_FILE_CONNECTION or
CLELFC
KEYED_FILE=file
STATUS=status variable

Parameters

KEYED_FILE or KF

Keyed file whose connection to the current lock file is to be deleted.

This parameter is required.

Remarks

- After using CLEAR_LOCK_FILE_CONNECTION, you can connect the lock file to another keyed file.
- Use CLEAR_LOCK_FILE_CONNECTION to delete all connections before you use DELETE_LOCK_FILE to delete the lock file.
- After you clear the connection, the keyed file is automatically connected to the default lock file $SYSTEM.AAM.AAF$LOCK_FILE. You do not need to clear this connection before creating another connection.

- For more information, see the NOS/VE Advanced File Management Usage manual.

CREATE_LOCK_FILE ADMLF Subcommand

Purpose Creates a lock file.

Format

CREATE_LOCK_FILE or
CRELF
PRIVATE=boolean
FILE_CLASS=application
INITIAL_VOLUME=name
STATUS=status variable

Parameters

PRIVATE or *P*

Specifies whether the lock file is used for more than one keyed file. TRUE indicates the lock file is used for one keyed file. FALSE indicates the lock file can be used for more than one keyed file.

The default is TRUE.

FILE_CLASS or *FC*

Parameter Attributes: BY_NAME

Class of the file to be assigned. The file class specifies the class of device on which the file will reside. Refer to the REQUEST_MASS_STORAGE command in the NOS/VE System Performance and Maintenance manual, Volume 2, for a complete description of this parameter.

The default is A.

INITIAL_VOLUME or *IV*

Parameter Attributes: BY_NAME

Name specifying the volume serial number (vsn) of the mass storage volume or volumes to which the file is to be assigned. The name is a 1- to 6-character string. The volume you specify must belong to the file class you specified with the FILE_CLASS parameter. Refer to the REQUEST_MASS_STORAGE command in the NOS/VE System Performance and Maintenance manual, volume 2, for a complete description of this parameter.

If omitted, a volume with the appropriate file class is chosen.

Remarks

- The lock file created by CREATE_LOCK_FILE is specified by the USE_LOCK_FILE subcommand.
- You need CONTROL and CYCLE permission for the lock file to use this subcommand.
- The lock file is given the same ring attributes at which you are executing.
- The lock file created, by default, has PUBLIC permission for READ and WRITE.
- You cannot use a file name of AAF$DEPENDENCY_FILE for the lock file.
- If you do not own the catalog where the lock file will reside, the lock file is created but the owner of the catalog owns the lock file.
- For more information, see the NOS/VE Advanced File Management Usage manual.

DELETE_LOCK_FILE ADMLF Subcommand

Purpose Deletes the current lock file.

Format

DELETE_LOCK_FILE or
DELLF
STATUS=status variable

Remarks

- To delete a lock file, you need CONTROL and CYCLE permission for the lock file.
- You need to clear all connections to the lock file with CLEAR_LOCK_FILE_CONNECTION before you delete the lock file. Otherwise, you cannot access the keyed file or files until you recreate the lock file or connect them to a different lock file.
- For more information, see the NOS/VE Advanced File Management Usage manual.

DISPLAY_LOCK_FILE ADMLF Subcommand

Purpose Lists the keyed file or files that are connected to the current lock file.

Format

DISPLAY_LOCK_FILE or
DISLF
OUTPUT=file
STATUS=status variable

Parameters *OUTPUT* or *O*

File where DISPLAY_LOCK_FILE writes the display.

If omitted, the default is $OUTPUT.

Remarks For more information, see the NOS/VE Advanced File Management manual.

DISPLAY_LOCK_FILE_CONNECTION ADMLF Subcommand

Purpose Displays the lock file that is connected to the specified keyed file.

Format

DISPLAY_LOCK_FILE_CONNECTION or
DISLFC
KEYED_FILE=file
OUTPUT=file
STATUS=status variable

Parameters **KEYED_FILE** or **KF**

Keyed file whose lock file connection is displayed.

OUTPUT or *O*

File where DISPLAY_LOCK_FILE_CONNECTION writes the display.

If omitted, the default is $OUTPUT.

Remarks For more information, see the NOS/VE Advanced File Management manual.

ESTABLISH_LOCK_FILE_CONNECTION ADMLF Subcommand

Purpose Establishes the connection between a lock file and a keyed file.

Format

ESTABLISH_LOCK_FILE_CONNECTION or **ESTLFC**
KEYED_FILE=file
STATUS=status variable

Parameters

KEYED_FILE or **KF**

Keyed file to connect to the lock file specified on the USE_LOCK_FILE subcommand.

This parameter is required.

Remarks

- If the keyed file you specify does not exist, the Administer_Lock_File utility creates it. However, you need to set the file attributes of the keyed file before you create it because the default file attributes are sometimes inappropriate for keyed files. Set the file attributes outside a Administer_Lock_File utility session with the SET_FILE_ATTRIBUTES command. For more information on the SET_FILE_ATTRIBUTES command, see the manual NOS/VE Commands and Functions.
- You cannot establish a connection under these circumstances:
  - The lock file is private and the keyed file is temporary.
  - The lock file is private and has an existing connection.
  - The keyed file has an existing connection.
- For more information, see the NOS/VE Advanced File Management Usage manual.

HELP
ADMLF Subcommand

Purpose Provides online help from within the Administer_Lock_File utility.

Format

HELP or
HEL
SUBJECT=string
MANUAL=name
STATUS=status variable

Parameters

SUBJECT or *S*

Topic to be located in the online manual index. The topic must be enclosed in single quotes.

If omitted, HELP displays a list of the available subcommands.

MANUAL or *M*

Online manual file whose index is searched.

AFM

The AFM online manual index is searched.

File

File name of the online manual whose index is searched.

If MANUAL is omitted, the default is AFM. The working catalog is searched for the file and then the $SYSTEM.MANUALS is searched.

Remarks For more information, see the NOS/VE Advanced File Management manual.

QUIT
ADMLF Subcommand

Purpose Ends the Administer_Lock_File utility session.

Format

QUIT or
QUI
STATUS=status variable

Remarks For more information, see the NOS/VE Advanced File Management manual.

USE_LOCK_FILE ADMLF Subcommand

Purpose Specifies the lock file that is used by any subsequent subcommands until you specify another lock file with USE_LOCK_FILE.

Format

USE_LOCK_FILE or
USELF
LOCK_FILE=file
STATUS=status variable

Parameters

LOCK_FILE or **LF**

Lock file to be used by all subsequent subcommands. This parameter is required.

Remarks

- You must use the USE_LOCK_FILE subcommand before any other subcommand, except for HELP, QUIT, or DISPLAY_LOCK_FILE_CONNECTION.
- You can switch lock files during an Administer_Lock_File session by using another USE_LOCK_FILE subcommand.
- USE_LOCK_FILE attaches the lock file with exclusive access.
- The lock file, if it exists, must be a permanent file with a sequential file organization.
- If the lock file does not exist, you can create it with the CREATE_LOCK_FILE subcommand.
- For more information, see the NOS/VE Advanced File Management Usage manual.

ADMINISTER_RECOVERY_LOG 5

ADMINISTER_RECOVERY_LOG Command

Purpose Begins an ADMINISTER_RECOVERY_LOG utility session.

Format

ADMINISTER_RECOVERY_LOG or
ADMRL
STATUS=status variable

Remarks For more information, see the NOS/VE Advanced File Management manual.

Examples The following is the minimal ADMINISTER_RECOVERY_LOG session; it does nothing.

```
/administer_recovery_log
admrl/quit
```

To see a list of available subcommands you can type HELP while in this utility.

BACKUP_LOG ADMRL Subcommand

Purpose Initiates an immediate backup of the log.

Format

BACKUP_LOG or
BACL
STATUS=status variable

Remarks

- This subcommand must be preceded in the session by a USE_LOG subcommand to specify the log to be backed up.
- This subcommand can be performed only on a log that has been configured for log backups. (This is done using the CONFIGURE_LOG_BACKUP subcommand.)

- You should use the BACKUP_LOG subcommand in both of the following situations:
  - Log users are receiving the status AAE$LOG_TEMPORARILY_ FULL, which indicates that an immediate repository switch is needed.
  - A system failure seems imminent.
- For more information see the NOS/VE Advanced File Management Usage manual.

Examples The following session initiates an immediate repository switch and backup for the existing log in $USER.MY_LOG.

```
/administer_recovery_log
admrl/use_log,catalog=$user.my_log
admrl/backup_log
admrl/quit
/
```

CANCEL_LOG_CHANGES ADMRL Subcommand

Purpose Discards the log specifications and any delete requests accumulated in the session.

Format **CANCEL_LOG_CHANGES** or
CANLC
STATUS=status variable

Remarks

- This subcommand discards the accumulated log specifications and delete requests before they are put into effect by the QUIT subcommand.
- The CANCEL_LOG_CHANGES subcommand is appropriate only after a USE_LOG subcommand has been entered.
- You can begin accumulating log specifications again after the CANCEL_LOG_CHANGES subcommand. To do so, you must begin with another USE_LOG subcommand to specify the log to be created or changed.

- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples The following session enters a change for $USER.MY_LOG, but then discards the change so the session does nothing.

```
/administer_recovery_log
admrl/use_log, $user.my_log, ..
admrl../set_performance_option, emphasis=speed
admrl/cancel_log_changes
admrl/quit
/
```

CLEAR_PROBLEM_JOURNAL ADMRL Subcommand

Purpose Clears the problem journal for the log.

Format **CLEAR_PROBLEM_JOURNAL** or **CLEPJ**

STATUS=status variable

Remarks

- The system maintains a problem journal in each log in which it records any problems that occur while using the log.
- You must display the problem journal before clearing it. To do so, use the DISPLAY_PROBLEM_JOURNAL subcommand.
- The log referenced by a CLEAR_PROBLEM_JOURNAL subcommand is the log specified on the USE_LOG subcommand earlier in the session.
- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples The following session prints the contents of the problem journal for $USER.MY_LOG before clearing the problem journal.

```
/administer_recovery_log
admrl/use_log, $user.my_log
admrl/display_problem_journal, output=log_problems
admrl/print_file, log_problems
admrl/clear_problem_journal
admrl/quit
/
```

CONFIGURE_LOG_BACKUP ADMRL Subcommand

Purpose Establishes the backup file pool for the log.

Format

CONFIGURE_LOG_BACKUP or
CONLB
ADD_FILE=file
REMOVE_FILE=file
MEDIA=keyword
EXTERNAL_VSN=list of string
RECORDED_VSN=list of string
TYPE=keyword
VERIFY=boolean
FILE_CLASS=application
INITIAL_VOLUME=name
STATUS=status variable

Parameters *ADD_FILE* or *AF*

File to be added to the pool of backup files for the log. If ADD_FILE is omitted, no backup file is added.

REMOVE_FILE or *RF*

File to be removed from the pool of backup files for the log. If REMOVE_FILE is omitted, no backup file is removed.

MEDIA or *M*

Device class of the file specified by the ADD_FILE parameter.

MAGNETIC_TAPE_DEVICE or MTD

Indicates that the log files are backed up to a labeled tape.

MASS_STORAGE_DEVICE or MSD

Indicates that the log files are backed up to disk. (The next four parameters are not used.)

The default value is MAGNETIC_TAPE_DEVICE.

EXTERNAL_VSN or *EVSN*

List of external VSNs identifying the tape volumes that compose the file specified by the ADD_FILE parameter. The VSNs are specified as strings of from 1 through 6 characters enclosed in apostrophes. This parameter must be specified if MEDIA is set to MAGNETIC_TAPE_DEVICE.

RECORDED_VSN or *RVSN*

List of recorded VSNs of the tape volumes that compose the file specified by the ADD_FILE parameter. The recorded VSN is in the ANSI VOL1 label on the volume. The VSNs are specified as strings of from 1 through 6 characters enclosed in apostrophes. This parameter must be specified if MEDIA is set to MAGNETIC_TAPE_DEVICE.

TYPE or *T*

Tape density written by a tape drive for the file specified by the ADD_FILE parameter. This parameter is used only if MEDIA is set to MAGNETIC_TAPE_DEVICE.

MT9$800

Indicates 800 cpi written by a nine-track tape drive.

MT9$1600

Indicates 1600 cpi written by a nine-track tape drive.

MT9$6250

Indicates 6250 cpi written by a nine-track tape drive.

MT18$38000

Indicates 38000 cpi written by a 16-track tape drive.

The default value is MT18$38000.

VERIFY or *V*

Indicates whether the backup file specified by the ADD_FILE parameter is verified. This parameter is used only if MEDIA is set to MAGNETIC_TAPE_DEVICE.

TRUE or YES or ON

The magnetic tape is mounted; the backup file is opened to verify that it exists and that it has read and write capabilities.

FALSE or NO or OFF

The backup file is not verified.

The default value is TRUE.

FILE_CLASS or *FC*

Specifies the class of the file to be assigned. Refer to the REQUEST_MASS_STORAGE command in the NOS/VE System Performance and Maintenance, Volume 2, Maintenance manual for class assignments and a complete description of this parameter. This parameter is used only if MEDIA is set to MASS_STORAGE_DEVICE.

INITIAL_VOLUME or *IV*

Name specifying the volume serial number (VSN) of the mass storage volume to which the file is to be assigned. The name is specified as a string of from 1 through 6 characters. Refer to the REQUEST_MASS_STORAGE command in the NOS/VE System Performance and Maintenance, Volume 2, Maintenance manual for a complete description of this parameter. This parameter is used only if MEDIA is set to MASS_STORAGE_DEVICE.

Remarks

- A mass storage backup file is specified by its file path. However, any file cycle specification on the file path is ignored. The backup is always written to cycle 1. (Cycle 1 is created if it does not exist and overwritten if it does exist.)

- If any backup files are configured for the log, a backup file must be configured for each log repository. For example, if backup files are configured, a log with five repositories must have five backup files.
- The FILE_CLASS and INITIAL_VOLUME parameters are described in detail as parameters of the REQUEST_MASS_STORAGE command in the NOS/VE System Performance and Maintenance, Volume 2, Maintenance manual.
- For more information, see the NOS/VE Advanced File Management Usage manual.

CONFIGURE_LOG_RESIDENCE ADMRL Subcommand

Purpose Establishes configuration of the log.

Format

CONFIGURE_LOG_RESIDENCE or
CONLR
REPOSITORIES=integer
REPOSITORY_SWITCHING_SIZE=integer
REPOSITORY_SWITCHING_TIME=integer
SWITCH_SUPPRESSION_SIZE=keyword or *integer*
SWITCH_SUPPRESSION_TIME=keyword or *integer*
REPOSITORY_SIZE_LIMIT=integer
FILE_CLASS=application
INITIAL_VOLUME=name
STATUS=status variable

Parameters

REPOSITORIES or *R*

Number of disk-resident repositories for the log (integer from 2 through 4096). The default value is 5.

If a backup account or backup pool is specified for the log, the log must have at least 3 repositories.

REPOSITORY_SWITCHING_SIZE or *RSS*

Repository size threshold for the log (in bytes, from 500,000 through 2,132,483,647 [(2^{31} - 1) - 15,000,000]). The default value is 70,000,000 bytes.

REPOSITORY_SWITCHING_TIME or *RST*

Repository time threshold for the log (in minutes, from 1 through 525,600 [365 days]). The default value is 1440 (24 hours).

SWITCH_SUPPRESSION_SIZE or *SSS*

Specifies the minimum repository size before switching. Options are:

Integer

Minimum repository size required before switching (in bytes, from 500,000 through 2,132,483,647 [$(2^{31} - 1) - 15,000,000$]).

NONE

No minimum repository size is required before switching.

The default is NONE.

SWITCH_SUPPRESSION_TIME or *SST*

Specifies the minimum repository time before switching. Options are:

Integer

Minimum repository time required before switching (in minutes, from 1 through 525,600 [365 days]).

NONE

No minimum repository time is required before switching.

The default is NONE.

REPOSITORY_SIZE_LIMIT or *RSL*

Absolute maximum repository size limit (in bytes, from 15,500,000 through 2,147,483,647 [$2^{31} - 1$]). It must be at least 15,000,000 bytes larger than the REPOSITORY_SWITCHING_SIZE. The default value is 100,000,000 bytes.

FILE_CLASS or *FC*

Specifies the class of the file to be assigned. Refer to the REQUEST_MASS_STORAGE command in the NOS/VE System Performance and Maintenance, Volume 2, Maintenance manual for class assignments and a complete description of this parameter.

INITIAL_VOLUME or *IV*

Name specifying the volume serial number (VSN) of the mass storage volume to which the file is to be assigned. The name is specified as a string of from 1 through 6 characters. Refer to the REQUEST_MASS_STORAGE command in the NOS/VE System Performance and Maintenance, Volume 2, Maintenance manual for a complete description of this parameter.

Remarks

- You cannot modify an existing log while any keyed file that uses the log is being updated. The subcommand notifies you when it cannot get exclusive access to the log. You should then quit the session and try again later.
- This subcommand can be specified only for a new log. The configuration cannot be changed for an existing log.
- During normal log activity, the active repository size should never approach the REPOSITORY_SIZE_LIMIT.
- The FILE_CLASS and INITIAL_VOLUME parameters are described in detail as parameters of the REQUEST_MASS_STORAGE command in the NOS/VE System Performance and Maintenance, Volume 2, Maintenance manual.
- For more information see the NOS/VE Advanced File Management Usage manual.

DELETE_LOG ADMRL Subcommand

Purpose Requests deletion of an existing log.

Format

DELETE_LOG or
DELL
CATALOG=file
RETAIN_CONFIGURATION=boolean
STATUS=status variable

Parameters

CATALOG or C

Catalog path of the log to be deleted. This parameter is required.

RETAIN_CONFIGURATION or RC

Indicates whether the log configuration is kept.

TRUE or YES or ON

Empty the repositories and the log journal, but keep the log configuration.

FALSE or NO or OFF

Delete all files composing the log, including the repositories, the log journal, and mass storage log backup files.

This parameter is required.

Remarks

- The logs specified by DELETE_LOG subcommands are not deleted until the QUIT subcommand is entered for the session. A CANCEL_LOG_CHANGES subcommand clears any pending deletion requests.
- If the log configuration is to be retained, the subcommand deletes all the log data, but the log data on the repositories continues to exist and can continue to be used.

  If the log configuration is not to be retained, the subcommand requests deletion of all files relating to the log in the catalog. The catalog will no longer be usable as a log until a new log is created in it.

  If the subcommand requests deletion of all files in the catalog, the catalog is deleted as well.

- The catalog used is specified on the DELETE_LOG subcommand. Therefore, the subcommand does not reference the log specified by the USE_LOG subcommand. More than one log can be deleted in a session.
- For more information see the NOS/VE Advanced File Management Usage manual.

Examples The following session requests deletion of log $USER.MY_LOG, but then cancels the request:

```
/administer_recovery_log
admrl/delete_log, $user.my_log, retain_configuration=false
admrl/cancel_log_changes
admrl/quit
/
```

DISPLAY_LOG_CONFIGURATION ADMRL Subcommand

Purpose Displays the current log specifications.

Format

DISPLAY_LOG_CONFIGURATION or
DISLC
OUTPUT=file
STATUS=status variable

Parameters *OUTPUT* or *O*

File to which the display is written.

The subcommand positions the file according to the file position ($BOI, $EOI) appended to the file reference or, if no position is specified, according to its OPEN_POSITION attribute value.

If OUTPUT is omitted, the display is written to the standard output file, $OUTPUT.

Remarks

- This subcommand must be preceded in the session by a USE_LOG subcommand to specify the log whose configuration is displayed.
- For more information see the NOS/VE Advanced File Management Usage manual.

DISPLAY_PROBLEM_JOURNAL ADMRL Subcommand

Purpose Displays the problem journal for the log.

Format

DISPLAY_PROBLEM_JOURNAL or
DISPJ
OUTPUT=file
STATUS=status variable

Parameters *OUTPUT* or *O*

File to which the display is written.

The subcommand positions the file according to the file position ($BOI, $EOI) appended to the file reference or, if no position is specified, according to its OPEN_POSITION attribute value.

If OUTPUT is omitted, the display is written to the standard output file, $OUTPUT.

Remarks

- The system records any problems that have occurred while using the log in the problem journal for the log.
- The log referenced by a DISPLAY_PROBLEM_JOURNAL subcommand is the log specified on the USE_LOG subcommand earlier in the session.
- For more information see the NOS/VE Advanced File Management Usage manual.

Examples The following session writes the problem journal for $USER.MY_LOG to file LOG_PROBLEMS and prints it.

```
/administer_recovery_log
admrl/use_log, $user.my_log
admrl/display_problem_journal, ..
admrl../output=log_problems
admrl/print_file, log_problems
admrl/quit
/
```

HELP
ADMRL Subcommand

Purpose Provides access to online information about the utility.

Format

HELP or
HEL
SUBJECT=string
MANUAL=file
STATUS=status variable

Parameters

SUBJECT or *S*

Topic to be found in the index of the online manual. The topic must be enclosed in apostrophes ('topic').

If you omit the SUBJECT parameter, HELP displays a list of the available subcommands and prompts for display of a subcommand description in the online manual.

MANUAL or *M*

Online manual file whose index is searched.

AFM

The AFM online manual index is searched.

File

File name of the online manual whose index is searched.

If MANUAL is omitted, the default is AFM. The working catalog is searched for the file and then the $SYSTEM.MANUALS is searched.

Remarks

- If the SUBJECT parameter specifies a topic that is not in the manual index, a nonfatal error is returned notifying you that the topic could not be found.
- The default manual file, $SYSTEM.MANUALS.AFM, contains the online version of the NOS/VE Advanced File Management Usage manual, as provided with the NOS/VE system.
- If your terminal is defined for screen applications, online manuals are displayed in screen mode. Help is available for reading the online. To leave the online manual and return to the utility, use QUIT.

- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples The following session shows the default display returned by the HELP subcommand.

```
/administer_recovery_log
admrl/help
The following Administer_Recovery_Log subcommands are available:
BACKUP_LOG
CANCEL_LOG_CHANGES
CLEAR_PROBLEM_JOURNAL
CONFIGURE_LOG_BACKUP
CONFIGURE_LOG_RESIDENCE
DELETE_LOG
DISPLAY_LOG_CONFIGURATION
DISPLAY_PROBLEM_JOURNAL
HELP
QUIT
SET_LOG_BACKUP_ACCOUNT
SET_PERFORMANCE_OPTION
SET_VERIFICATION_LEVEL
```

For a description of a subcommand in the online manual, enter: HELP subject = '<subcommand>'

To return from an online manual, enter: QUIT

```
admrl/quit
/
```

QUIT
ADMRL Subcommand

Purpose Executes the accumulated log specifications and ends the session.

Format

QUIT or
QUI
APPLY_LOG_CHANGES=boolean
STATUS=status variable

Parameters *APPLY_LOG_CHANGES* or *ALC*

Indicates whether the log repositories are created or updated based upon the accumulated log specifications.

TRUE or YES or ON

The log is created or updated. Any logs specified on a DELETE_LOG subcommand during the session are deleted.

If a new log is being created, the log catalog is created if it does not exist. The log files are created and initialized. If the log catalog already exists, only the performance option and backup account information can be changed.

FALSE or NO or OFF

Log repositories are not created or updated; log specifications are discarded. Any logs specified on a DELETE_LOG subcommand during the session are kept.

The default value is TRUE.

Remarks

- To discard the accumulated log specifications or delete requests before ending the session, enter a CANCEL_LOG_CHANGES subcommand before entering the QUIT subcommand.
- The changes specified by the following subcommands do not take effect until the log changes are applied when the QUIT subcommand is entered:

  CONFIGURE_LOG_BACKUP
  CONFIGURE_LOG_RESIDENCE
  DELETE_LOG
  SET_LOG_BACKUP_ACCOUNT
  SET_PERFORMANCE_OPTION
  SET_VERIFICATION_LEVEL

- For more information, see the NOS/VE Advanced File Management Usage manual.

SET_LOG_BACKUP_ACCOUNT ADMRL Subcommand

Purpose Specifies the validation information used by backup jobs for the log.

NOTE

Each time the password is changed for the user name used as the backup account, the password must also be changed in the log configuration. Otherwise, all subsequent backup jobs fail to execute.

Format

SET_LOG_BACKUP_ACCOUNT or
SETLBA
USER=name
PASSWORD=name
FAMILY_NAME=name
USER_JOB_NAME=name
JOB_CLASS=name
ACCOUNT=name
PROJECT=name
OUTPUT_DISPOSITION=keyword or *file*
USER_INFORMATION=string
STATUS=status variable

Parameters

USER or **U**

User name under which backup jobs are run. This parameter is required.

PASSWORD or **PW**

Password for the user name specified by the USER parameter. This parameter is required.

FAMILY_NAME or *FN*

Optional family name under which backup jobs are run. If FAMILY_ NAME is omitted, backup jobs run under the family to which the specified user name belongs.

USER_JOB_NAME or *JOB_NAME* or *UJN* or *JN*

Optional name by which the backup jobs are identified in the system. If USER_JOB_NAME is omitted, the name assigned backup jobs is the user name.

JOB_CLASS or *JC*

Optional job class in which the backup jobs are run. If JOB_ CLASS is omitted, the jobs run in the default job class for the user name.

ACCOUNT or *A*

Account to which resource usage is charged for the backup jobs. If you omit this parameter for a user name that requires an account, the backup jobs will fail to execute. (See the Remarks.)

PROJECT or *P*

Project to which resource usage is charged for the backup jobs. If you omit this parameter for a user name that requires a project, the backup jobs will fail to execute. (See the Remarks.)

OUTPUT_DISPOSITION or *OD* or *ODI* or *STANDARD_OUTPUT* or *SO*

Specifies the default for how the backup job's standard output is to be disposed. If omitted, the attribute associated with this parameter does not change.

File name

The standard output is copied to the specified file name at job end.

DISCARD_ALL_OUTPUT or DAO

All output generated by the backup job is to be discarded at job end.

DISCARD_STANDARD_OUTPUT or DSO

Standard output is to be discarded at job end.

LOCAL or L

Any output generated by the backup job is printed at the destination system rather than being returned to the originating user's default output station.

PRINTER or P

Any output generated by the backup job is returned to the originating user's default output station.

WAIT_QUEUE or WQ

Any output generated by the backup job is returned to the originating user's $WAIT_QUEUE subcatalog on the originating system using the user's job name for the file name. If the $WAIT_QUEUE subcatalog does not exist at the time the output files are returned, it is created for the user.

The default value is PRINTER.

USER_INFORMATION or *UI*

Specifies a user information string of up to 256 characters. This string enables you to pass information (such as a file path) to a backup job. This string is also passed on to all output files generated by the backup job.

If omitted, the user information string associated with the backup job is assumed.

Remarks

- If backup files are included in the log configuration, each repository switch for the log starts a job to back up the log. Each backup job uses the validation information specified on this subcommand.

- To determine if the ACCOUNT and PROJECT parameters are required and the valid JOB_CLASS values, display the validation information for the user name.

  To display validation information for a user name, use the Administer_User utility with the DISPLAY_USER subcommand. If you are logged in as the family administrator, you can display information on any user in the family; otherwise, you can display information only for the user name you are using.

  For more information about family administration and user validation see the NOS/VE User Validation manual and the NOS/VE System Usage manual.

- For more information see the NOS/VE Advanced File Management Usage manual.

SET_PERFORMANCE_OPTION ADMRL Subcommand

Purpose Specifies the performance emphasis (speed or reliability) for the log.

Format

SET_PERFORMANCE_OPTION or
SETPO
EMPHASIS=keyword
LOG_ENTRY=keyword
STATUS=status variable

Parameters

EMPHASIS or E

Specifies whether speed or reliability is more important.

SPEED or S

Speed is more important than reliability.

RELIABILITY or R

Reliability is more important than than speed.

BALANCED or B

Both speed and reliability are important.

This parameter is required.

LOG_ENTRY or *LOG_ENTRIES* or *LE*

Indicates the types of log entries to which the specified emphasis applies.

RECORD or R

Record entries, but not parcel entries.

PARCEL or P

For future implementation.

ALL or A

For future implementation.

The default value is RECORD.

Remarks

- This subcommand determines how frequently log entries in memory are written to disk. (Its purpose is similar to that of the FORCED_WRITE attribute for keyed files.)
- If this subcommand is not specified, the default performance option is BALANCED.
- The EMPHASIS values have the following meanings:

  SPEED

  The system memory manager determines when log entries are written to disk.

RELIABILITY

Each log entry is written to disk before the next log entry begins.

BALANCED

The system must begin writing a log entry to disk before the next log entry can begin.

- Any value specified for parcels is recorded for future use, but is currently ignored.
- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples The following session changes the performance options for $USER.MY_LOG.

```
/administer_recovery_log
admrl/use_log, $user.my_log
admrl/set_performance_option, ..
admrl../emphasis=reliability
admrl/quit
/
```

SET_VERIFICATION_LEVEL ADMRL Subcommand

Purpose Indicates whether checksums should be performed for the header and trailer parts of log records.

Format

SET_VERIFICATION_LEVEL or
SETVL
VERIFY_LOG_ENTRIES=boolean
STATUS=status variable

Parameters

VERIFY_LOG_ENTRIES or **VLE**

Indicates whether checksums are performed for the log.

TRUE or YES or ON

Checksums are performed.

FALSE or NO or OFF

Checksums are not performed.

This parameter is required.

Remarks

- This subcommand can be specified only for a new log. The verification level cannot be changed for an existing log.
- This subcommand is optional. If it is omitted from a session that creates a new log, the default verification level is FALSE.
- For more information see the NOS/VE Advanced File Management Usage manual.

USE_LOG
ADMRL Subcommand

Purpose

Establishes the log to be created or changed by the session.

Format

USE_LOG or
USEL
CATALOG=file
STATUS=status variable

Parameters

CATALOG or **C**

Catalog path for the log created or changed by the session.

A session can create or change only one log; therefore, any subsequent USE_LOG subcommands are ignored.

If the catalog does not exist, the subcommand creates it. If the catalog exists, but does not contain a log, a log is created in it. If a log exists in the catalog, the session verifies that the log contains the proper characteristics.

This parameter is required.

Remarks

- You must establish a catalog before any of the other subcommands (except QUIT, DELETE_LOG, HELP, or CANCEL_LOG_CHANGES (after DELETE_LOG)) can be entered.
- Once established, the catalog can only be changed after using CANCEL_LOG_CHANGES.
- For more information see the NOS/VE Advanced File Management Usage manual.

Examples The following session establishes $USER.MY_LOG as the log to be used. The performance options for $USER.MY_LOG are changed, but then the changes are canceled and another log is specified.

```
/administer_recovery_log
admrl/use_log, $user.my_log
admrl/set_performance_option, emphasis=reliability
admrl/cancel_log_changes
admrl/use_log, $user.my_log_2
admrl/
```

ADMINISTER_VALIDATIONS 6

ADMINISTER_VALIDATIONS Command

Purpose Starts the ADMINISTER_VALIDATIONS utility to change and display validations.

Format

ADMINISTER_VALIDATIONS or
ADMINISTER_VALIDATION or
ADMV
STATUS=status variable

Remarks For more information, see the NOS/VE User Validation manual.

CHANGE_DEFAULT_ACCOUNT_PROJECT CREU and CHAU Subcommand

Purpose Changes the default account and project for the LOGIN and SUBMIT_JOB commands.

Format

CHANGE_DEFAULT_ACCOUNT_PROJECT or
CHADAP
ACCOUNT=keyword or *name*
PROJECT=keyword or *name*
STATUS=status variable

Parameters *ACCOUNT* or *A*

Specifies the account name. If the validation level is ACCOUNT or PROJECT and the account you specify does not exist, a warning message appears. You can specify a name or one of the following keywords:

DEFAULT

The account is set to the default value specified in the DEFAULT_ACCOUNT_PROJECT field description.

CURRENT

The account of the job executing this command is used.

NONE

There is no default account for the user name.

PROJECT or *P*

Specifies the project name. If the validation level is PROJECT and the project you specify does not exist, a warning message appears. You can specify a name or one of the following keywords:

DEFAULT

The project is set to the default value specified in the DEFAULT_ACCOUNT_PROJECT field description.

CURRENT

The project of the job executing this command is used.

NONE

There is no default project for the user name.

Remarks For more information, see the NOS/VE User Validation manual.

Examples To change the default login account and project, enter:

```
ADMV/change_user
CHAU/change_default_account_project ..
Changing user TERRY.
CHAU../account=a project=b
CHAU/quit
```

CHANGE_LINK_ATTRIBUTE_CHARGE CREU and CHAU Subcommand

Purpose Changes the charge number needed to gain access to NOS or NOS/BE permanent files or to submit a job to NOS or NOS/BE.

Format **CHANGE_LINK_ATTRIBUTE_CHARGE** or **CHALAC**

VALUE=keyword or *string*
STATUS=status variable

Parameters *VALUE* or *V*

Specifies a NOS or NOS/BE charge number. By default, the link attribute charge number is not changed. If you specify DEFAULT, the default is the value specified in the LINK_ATTRIBUTE_CHARGE field description.

Remarks

- You can override this value using the CHANGE_LINK_ATTRIBUTE command.
- For more information, see the NOS/VE User Validation manual.

CHANGE_LINK_ATTRIBUTE_FAMILY CREU and CHAU Subcommand

Purpose Changes the family name needed to gain access to NOS or NOS/BE permanent files or to submit a job to NOS or NOS/BE.

Format **CHANGE_LINK_ATTRIBUTE_FAMILY** or **CHALAF**

VALUE=keyword or *string*
STATUS=status variable

Parameters *VALUE* or *V*

Specifies a NOS or NOS/BE family name. By default, the link attribute family is not changed. If you specify DEFAULT, the default is the value specified in the LINK_ATTRIBUTE_FAMILY field description.

Remarks

- You can override this value using the CHANGE_LINK_ATTRIBUTE command.
- For more information, see the NOS/VE User Validation manual.

CHANGE_LINK_ATTRIBUTE_PASSWORD CREU and CHAU Subcommand

Purpose Changes the password needed to gain access to NOS or NOS/BE permanent files, or to submit a job to NOS or NOS/BE.

Format **CHANGE_LINK_ATTRIBUTE_PASSWORD** or **CHALAPW**
VALUE=keyword or *string*
STATUS=status variable

Parameters *VALUE* or *V*

Parameter Attributes: SECURE

Specifies a NOS or NOS/BE password. By default, the link attribute password is not changed. If you specify DEFAULT, the default is the value specified in the LINK_ATTRIBUTE_PASSWORD field description.

Remarks

- You can override this value using the CHANGE_LINK_ATTRIBUTE command.
- For more information, see the NOS/VE User Validation manual.

CHANGE_LINK_ATTRIBUTE_PROJECT CREU and CHAU Subcommand

Purpose Changes the project number needed to gain access to NOS or NOS/BE permanent files, or to submit a job to NOS or NOS/BE.

Format **CHANGE_LINK_ATTRIBUTE_PROJECT** or **CHALAP**
VALUE=keyword or *string*
STATUS=status variable

Parameters *VALUE* or *V*

Specifies a project number needed to gain access to NOS and NOS/BE permanent files or to submit a job to NOS or NOS/BE. By default, the link attribute project is not changed. If you specify DEFAULT, the default is the value specified in the LINK_ATTRIBUTE_PROJECT field description.

Remarks

- You can override this value using the CHANGE_LINK_ATTRIBUTE command.
- For more information, see the NOS/VE User Validation manual.

CHANGE_LINK_ATTRIBUTE_USER CREU and CHAU Subcommand

Purpose Changes the user name needed to gain access to NOS or NOS/BE permanent files, or to submit a job to NOS or NOS/BE.

Format

CHANGE_LINK_ATTRIBUTE_USER or
CHALAU
VALUE=keyword or *string*
STATUS=status variable

Parameters *VALUE* or *V*

Specifies a NOS or NOS/BE user name. By default, the link attribute user is not changed. If you specify DEFAULT, the default is the value specified in the LINK_ATTRIBUTE_USER field description.

Remarks

- You can override this value using the CHANGE_LINK_ATTRIBUTE command.
- For more information, see the NOS/VE User Validation manual.

CHANGE_LOGIN_PASSWORD CREU and CHAU Subcommand

Purpose Changes information about the user's login password.

Format

CHANGE_LOGIN_PASSWORD or
CHALPW
OLD_PASSWORD=name
NEW_PASSWORD=name
EXPIRATION_DATE=keyword or *date_time*
EXPIRATION_INTERVAL=keyword or *integer*
EXPIRATION_WARNING_INTERVAL=keyword or *integer*
MAXIMUM_EXPIRATION_INTERVAL=keyword or *integer*
ADD_ATTRIBUTES=keyword or *list of name*
DELETE_ATTRIBUTES=keyword or *list of name*
STATUS=status variable

Parameters *OLD_PASSWORD* or *OPW*

Parameter Attributes: SECURE

Specifies the current login password. To change a password, a user must specify the old password. Administrators need not specify the old password to change a password.

NEW_PASSWORD or *NPW*

Parameter Attributes: SECURE

Specifies a new login password for the user. By default, the password is not changed.

EXPIRATION_DATE or *ED*

Parameter Attributes: BY_NAME

Specifies the date and time the password expires. The number of days between the current date and the EXPIRATION_DATE cannot exceed the number of days specified by the MAXIMUM_EXPIRATION_INTERVAL parameter.

The format is YYYY-MM-DD.HH:MM:SS. The hours, minutes, and seconds portion is optional, and the time defaults to midnight 00:00:00.

The default expiration date for new passwords is the current date plus the value specified by the EXPIRATION_INTERVAL parameter. The default expiration date for an existing password is the current expiration date for that password.

NONE

The password does not have an expiration date.

DEFAULT

The expiration date is set to the default value specified in the LOGIN_PASSWORD field description.

EXPIRATION_INTERVAL or *EI*

Parameter Attributes: BY_NAME

Specifies the number of days (1 to 365) until the password expires. The number of days specified by the EXPIRATION_INTERVAL parameter must not exceed the MAXIMUM_EXPIRATION_INTERVAL parameter. By

default, the current EXPIRATION_INTERVAL parameter value is not changed. You can also specify one of the following keywords:

UNLIMITED

The password will not expire unless a specific date is specified by the EXPIRATION_DATE parameter.

DEFAULT

The expiration interval is set to the default value specified in the LOGIN_PASSWORD field description.

EXPIRATION_WARNING_INTERVAL or *EWI*

Parameter Attributes: BY_NAME

Specifies the number of days (0 to 365) before the password expiration date that warnings are sent to the user that the password will expire. If you specify zero, the user does not receive a warning. The default is that the current value is not changed. You can also specify one of the following keywords:

UNLIMITED

The user always receives a warning during each login.

DEFAULT

The expiration warning interval is set to the default value specified in the LOGIN_PASSWORD field description.

MAXIMUM_EXPIRATION_INTERVAL or *MAXEI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the maximum value for the EXPIRATION_INTERVAL parameter. Only users with user administration capability can specify a value for this parameter.

ADD_ATTRIBUTES or *AA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies a list of site-defined password attributes to be added. Only users with user administration capabilities can specify a value for this parameter.

DELETE_ATTRIBUTES or *DA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies a list of site-defined password attributes to be deleted. Only users with user administration capability can specify this parameter.

Remarks

- You can also change passwords using the CHANGE_LOGIN_PASSWORD command.
- You can change your expiration date only when you change your password.
- For more information, see the NOS/VE User Validation manual.

Examples To change the password and set the expiration date, enter:

```
ADMV/change_user
Changing user ABC.
CHAU/change_login_password
CHAU../old_password=example ..
CHAU../new_password=sample ..
CHAU../expiration_date=1989-12-10 ..
CHAU../expiration_interval=60 ..
CHAU/quit
ADMV/
```

This password expires in 60 days.

CHANGE_USER ADMV Subcommand

Purpose Starts the CHANGE_USER subutility to change validations for an existing user.

Format

CHANGE_USER or
CHAU
USER=name
STATUS=status variable

Parameters *USER* or *U*

Specifies the user name to be changed. The default is the user name specified during login.

Remarks

- A system or family administrator can change any user validations; account or project members with a user administration capability can change user validations only for users under their control; users can change only some of their own validations.
- For more information, see the NOS/VE User Validation manual.

Examples To change the default account and project for the LOGIN and SUBMIT_JOB commands, enter:

```
ADMV/change_user user=ABC
Changing user ABC.
CHAU/change_default_account_project account=a ..
CHAU../project=b
CHAU/quit
ADMV/
```

CHANGE_USER_EPILOG CREU and CHAU Subcommand

Purpose Changes the name of the user's epilog file.

Format

CHANGE_USER_EPILOG or
CHAUE
VALUE=keyword or *file* or *string*
STATUS=status variable

Parameters *VALUE* or *V*

Specifies the new file reference. If you specify a file path, the system resolves the reference immediately. If you specify a string, the system resolves the string reference during epilog execution.

DEFAULT

The name of the user epilog is set to the default value defined by the administrator.

NONE

The file reference $NULL is used.

Remarks For more information, see the NOS/VE User Validation manual.

Examples To change your epilog so that file ALL_DONE is used, enter:

```
ADMV/change_user
Changing user ABC.
CHAU/change_user_epilog value=$user.all_done
CHAU/quit
ADMV/
```

CHANGE_USER_PROLOG
CREU and CHAU Subcommand

Purpose Changes the name of the user's prolog file.

Format **CHANGE_USER_PROLOG** or
CHAUP
VALUE=keyword or *file* or *string*
STATUS=status variable

Parameters *VALUE* or *V*

Specifies the new file reference. If you specify a file path, the system resolves the file path immediately; if you specify a string, the system resolves the reference during prolog execution.

DEFAULT

The name of the user prolog is set to the default value defined by the administrator.

NONE

The file reference $NULL is used.

Examples To change your prolog so that file START_UP is used, enter:

```
ADMV/change_user
Changing user ABC.
CHAU/change_user_prolog value=$user.start_up
CHAU/quit
ADMV/
```

DISPLAY_USER
ADMV Subcommand

Purpose Displays your validations.

Format

DISPLAY_USER or
DISPLAY_USERS or
DISU
USER=keyword or *list of name*
OUTPUT=file
DISPLAY_OPTION=keyword or *list of name*
STATUS=status variable

Parameters

USER or *USERS* or *U*

Specifies the user names to be displayed. The default is the user name specified during login.

OUTPUT or *O*

Specifies the file to which information is written. The default is $OUTPUT.

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Parameter Attributes: BY_NAME

Specifies the names of the user validations to be displayed. You can specify a list of names or one of the following keywords; the default is ALL:

ALL

Displays the value of all user validations.

NONE

Displays only user names.

Remarks For more information, see the NOS/VE User Validation manual.

Examples

- To display all of the validations, enter:

```
ADMV/display_user
```

- To display the default login account and project, enter:

```
ADMV/display_user all ..
ADMV../display_option=default_account_project
```

QUIT
ADMV Subcommand

Purpose Ends an ADMINISTER_VALIDATIONS utility session.

Format
QUIT or
ENDAV or
END_ADMINISTER_VALIDATIONS or
QUI

Parameters None.

Remarks For more information, see the NOS/VE User Validation manual.

END_CHANGE_USER
CHAU Subcommand

Purpose Ends a CHANGE_USER subutility session.

Format
END_CHANGE_USER or
ENDCU or
QUIT or
QUI
WRITE_CHANGES=boolean

Parameters *WRITE_CHANGES* or *WC*

Specifies whether the changes made during the CHANGE_USER subutility session are written to the validation file. The default is TRUE.

TRUE

The changes are written to the validation file.

FALSE

The changes are not written to the validation file.

Remarks For more information, see the NOS/VE User Validation manual.

ANALYZE_OBJECT_LIBRARY 7

ANALYZE_OBJECT_LIBRARY Command

Purpose Begins an ANALYZE_OBJECT_LIBRARY utility session. The subcommands for this object code utility display the internal characteristics of object modules, including: object record counts, section sizes, section attributes, and performance data for modules on an object library or object file.

Format

ANALYZE_OBJECT_LIBRARY or
ANAOL
LIBRARY=file
STATUS=status variable

Parameters *LIBRARY* or *L*

Object library or object file to be analyzed.

If LIBRARY is omitted, you must use the USE_LIBRARY subcommand to specify the object library or object file.

Remarks

- After entering the ANALYZE_OBJECT_LIBRARY command, you can enter any of the ANAOL subcommands. The ANAOL session ends when you enter the QUIT subcommand.
- An object library or file must be specified on the ANALYZE_OBJECT_LIBRARY command or on the USE_LIBRARY subcommand before an ANAOL session can continue.
- For more information, see the NOS/VE Object Code Management manual.

Examples The following is a sequence that enters the ANALYZE_OBJECT_LIBRARY utility, specifies LGO as the file to be analyzed, and displays the characteristics of library LGO.

```
/analyze_object_library lgo
AOL/display_library_analysis
Library Analysis of LGO
Number of modules:  2
Record Analysis

    Identification records:    2
    Libraries:                 2 - items:    10
    Section definitions:       9
    Text records:              21 - items:   519
       .
       .
    Relocation records:        2 - items:     8
    Binding templates:         8
    Transfer symbols:          2
                               ---
      Total records:           84

AOL/quit
```

DISPLAY_LIBRARY_ANALYSIS ANAOL Subcommand

Purpose Displays the number of modules and/or the total number of each type of object record on the current object library or file. The current object library or file is specified by a previous USE_LIBRARY subcommand or ANALYZE_OBJECT_LIBRARY command.

Format **DISPLAY_LIBRARY_ANALYSIS** or **DISLA**

DISPLAY_OPTIONS=keyword or *list of keyword*
OUTPUT=file
STATUS=status variable

Parameters *DISPLAY_OPTIONS* or *DISPLAY_OPTION* or *DO*

List of one or more keywords indicating the analysis information to be displayed. Options are:

NUMBER_OF_MODULES or NOM

Number of modules on the object library or file.

RECORD_ANALYSIS or RA

Total number of each type of object record on the object library or file.

ALL

All of the previously listed options.

IF DISPLAY_OPTION is omitted, all analysis information is displayed.

OUTPUT or *O*

Output file. This file can be positioned.

If OUTPUT is omitted, file $OUTPUT is used.

Remarks

- In a library analysis (see example), the record analysis contains the number of each type of object record in the library or file. The total number of adaptable items is also listed with the object records that have adaptable fields.
- For more information, see the NOS/VE Object Code Management manual.

Examples The following ANAOL session lists the number of modules and the type and number of object records in the current library LGO.

```
/analyze_object_library lgo
AOL/display_library_analysis

Library Analysis of LGO

  Number of modules:  2

  Record Analysis

    Identification records:              2
    Libraries:                           2   items:      10
    Section definitions:                 9
    Text records:                       21   items:     519
    Address formulation records:        31   items:      31
    External linkage records:            5   items:       5
    Entry definitions:                   2
    Relocation records:                  2   items:       8
    Binding templates:                   8
    Transfer symbols:                    2
                                    --------
      Total records:                    84

AOL/
```

DISPLAY_MODULE_ANALYSIS ANAOL Subcommand

Purpose Displays analysis information about specified modules on the object library or file, such as:

- Total number of each type of object record in the module.
- Size, type, attributes initialized, addresses in, externals in, and addresses to each section in the module.

The current object library or file is specified by a previous USE_LIBRARY subcommand or ANALYZE_OBJECT_LIBRARY command.

Format **DISPLAY_MODULE_ANALYSIS** or **DISMA**

MODULES=keyword or *list of program_name* or *list of range of program_name*
DISPLAY_OPTIONS=keyword or *list of keyword*
OUTPUT=file
STATUS=status variable

Parameters

MODULES or *MODULE* or *M*

List of modules whose analysis information is to be displayed.

You use a string value for a module whose name is not an SCL name or a COBOL name.

If MODULE is omitted or the keyword ALL is used, analysis information for all modules in the object library or file is displayed.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

List of one or more keywords indicating the analysis information to be displayed. Options are:

RECORD_ANALYSIS or RA

Total number of each type of object record in the module.

SECTION_ANALYSIS or SA

Size, type, attributes, bytes initialized, addresses built in this section, and addresses built in other sections that the loader will build that point to this section.

ALL

All of the previously listed options.

If DISPLAY_OPTION is omitted, all analysis information is displayed.

OUTPUT or *O*

Output file. This file can be positioned. If OUTPUT is omitted, file $OUTPUT is used.

Remarks

- In a module analysis display, the record analysis contains the number of each type of object record in the module. The total number of adaptable items is also listed with the object records that have adaptable fields. The number of items contained in the next column lists the total size of the adaptable record types.

- The section analysis display includes the following:
  - Total number of bytes in the section.
  - Section type: code section, binding section, working storage section, common block, extensible working storage, and extensible common block.
  - Attributes of the section: R=read, W=write, X=execute, and B=binding.
  - Number of bytes initialized in the section by text and replication records or by allotted text.
  - Number of internal addresses (Addresses in) the loader will build in this section.
  - Number of addresses (Addresses to) in other sections the loader will build that point to this section.
- For more information, see the NOS/VE Object Code Management manual.

Examples The following subcommand lists the record analysis and section analysis of module TEST.

```
AOL/display_module_analysis module=test

Module Analysis of TEST

  Record Analysis

    Identification records:              1
    Libraries:                           1   items:          5
    Section definitions:                 4
    Text records:                        9   items:        233
    Address formulation records:        15   items:         15
    External linkage records:            2   items:          2
    Entry definitions:                   1
    Relocation records:                  1   items:          4
    Binding templates:                   4
    Transfer symbols:                    1
                                    --------
      Total records:                    39

  Section Analysis

    Section: TEST      60 bytes       CODE [ R X ]
      Bytes initialized: 60 Addresses to: 1
    Section:           56 bytes       BINDING [ B ]
      Externals in: 2 Addresses in: 3 Addresses to: 1
    Section:          207 bytes       WORKING STORAGE [ R ]
      Bytes initialized: 163 Addresses in: 6 Addresses to: 8
    Section:          104 bytes       WORKING STORAGE [ R W ]
      Bytes initialized: 10 Addresses in: 7 Addresses to: 5

AOL/
```

DISPLAY_PERFORMANCE_DATA ANAOL Subcommand

Purpose Displays possible load and execution time performance problems that may exist in specified modules on the object library or file. The current object library or file is specified by a previous USE_LIBRARY subcommand or ANALYZE_OBJECT_LIBRARY command.

Format **DISPLAY_PERFORMANCE_DATA** or **DISPD**

MODULES=keyword or *list of program_name* or *list of range of program_name*
PERFORMANCE_DATA=keyword or *list of keyword*
DISPLAY_OPTION=keyword or *list of keyword*
OUTPUT=file
STATUS=status variable

Parameters *MODULES* or *MODULE* or *M*

List of modules whose performance data is to be displayed.

You use a string value for a module whose name is not an SCL name or a COBOL name.

If MODULE is omitted or keyword ALL is specified, performance data for all modules is displayed.

PERFORMANCE_DATA or *PD*

List of one or more keywords indicating the performance data to be displayed. Options are:

BOUND_MODULES or BM

Bound modules that have not been prelinked.

LINE_TABLES or LT

Modules that have debug line address tables.

LOAD_MODULES or LM

Load modules that have not been bound.

MULTIPLE_ENTRY_POINTS or MEP

Bound or prelinked modules that have multiple entry points.

OBJECT_MODULES or OM

Object modules that are not on an object library.

OPT_DEBUG or OD

Modules that are compiled with the parameter OPTIMIZATION_LEVEL=DEBUG.

OPT_LOW or OL

Modules that are compiled with the parameter OPTIMIZATION_LEVEL=LOW.

PARAMETER_CHECKING or PC

Modules that have parameter checking records.

RUNTIME_CHECKING or RC

Modules that have run-time range checking for variables, subscripts, and substring character expressions.

RUNTIME_LIBRARIES or RL

Modules that have text-embedded run-time library directives.

RUNTIME_LIBRARY_CALLS or RLC

Modules that have calls to local run-time libraries.

SYMBOL_TABLES or ST

Modules that have debug symbol tables.

UNREFERENCED_SECTIONS or US

Modules that have uninitialized and unreferenced sections.

ALL

Both DESCRIPTION and MODULE_NAMES display options.

If PERFORMANCE_DATA is omitted, all performance data is displayed.

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

List of one or more keywords indicating the information to be displayed. The number of modules with the possible performance problem is always displayed. Options are:

NONE

No information other than the number of modules with the possible performance problem.

MODULE_NAMES or MN

Names of modules with the possible performance problem.

DESCRIPTION or D

Brief description of the possible performance problem and recommended changes to correct the problem.

ALL

Both DESCRIPTION and MODULE_NAMES options.

If DISPLAY_OPTION is omitted, the number of modules with the possible problem and the description of the problem (DESCRIPTION) are displayed.

OUTPUT or *O*

Output file. This file can be positioned.

If OUTPUT is omitted, file $OUTPUT is used.

Remarks

- The analysis performed is very general, and the recommendations may not be applicable to all programs. Each recommendation should be looked at to determine if any changes should be made to the program or its packaging.
- The quality of analysis performed depends on the amount of information placed in the object modules by the compilers. Some modules may have performance problems that are not detected.
- Some compilers put performance information in the comment string in the module header. If this string has been changed, the information will not be available to DISPLAY_PERFORMANCE_DATA.
- Since binding and prelinking may hide some of a product's performance problems, analysis should also be done on the unbound product.
- For more information, see the NOS/VE Object Code Management manual.

DISPLAY_SECTION_ANALYSIS ANAOL Subcommand

Purpose Displays section usage information for specified modules on the object library or file. Information displayed includes size, attributes, bytes initialized, addresses in the section, and addresses to the section. The current object library or file is specified by a previous USE_LIBRARY subcommand or ANALYZE_OBJECT_LIBRARY command.

Format

DISPLAY_SECTION_ANALYSIS or
DISSA
MODULES=keyword or *list of program_name* or *list of range of program_name*
SECTION_KINDS=keyword or *list of keyword*

SECTION_ACCESS_ATTRIBUTES=keyword or *list of keyword*
SECTION_NAME=name
OUTPUT=file
STATUS=status variable

Parameters

MODULES or *MODULE* or *M*

List of modules whose section usage information is to be displayed.

Use a string value for a module whose name is not an SCL name or a COBOL name.

If MODULE is omitted or the keyword ALL is used, section usage information for all modules in the object library or file is displayed.

SECTION_KINDS or *SK*

List of one or more keywords indicating the type of section to be displayed. Types are:

CODE or C

Code section.

BINDING or B

Binding section.

WORKING_STORAGE or WS

Working storage section.

EXTENSIBLE_WORKING_STORAGE or EWS

Extensible working storage section.

COMMON_BLOCK or CB

Common block section.

EXTENSIBLE_COMMON_BLOCK or ECB

Extensible common block section.

ALL

All of the previously listed section types:

If SECTION_KIND is omitted, all section types are displayed.

SECTION_ACCESS_ATTRIBUTES or *SAA*

List of one or more keywords indicating the access attributes of the section to be displayed. The access attributes are:

READ or R

Read attributes.

WRITE or W

Write attributes.

EXECUTE or E

Execute attributes.

BINDING or B

Binding attributes.

ALL

Any of the listed attributes.

If SECTION_ACCESS_ATTRIBUTE is omitted, sections with any attributes are displayed.

SECTION_NAME or *SN*

The name of the section to be displayed. If SECTION_NAME is omitted, sections with any names are displayed.

Use a string value for a section whose name is not an SCL name.

OUTPUT or *O*

Output file. This file can be positioned.

If OUTPUT is omitted, file $OUTPUT is used.

Remarks

The section analysis display (see example) includes the following:

- Name of section (if any).
- Total number of bytes in the section.
- Section type: code section, binding section, working storage section, common block, extensible working storage, and extensible common block.

- Attributes of the section: R=read, W=write, X=execute, B=binding.
- Number of bytes initialized in the section by text and replication records or by allotted text.
- Number of internal addresses (Addresses in) and external addresses (Externals in) the loader will build in this section.
- Number of addresses (Addresses to) in other sections which the loader will build that point to this section.
- For more information, see the NOS/VE Object Code Management manual.

Examples The following subcommand lists the section definitions for module SUB.

```
AOL/display_section_analysis module=sub

Section Usage of SUB
  Section: SUB       50 bytes            CODE [ R X ]
    Bytes initialized: 50
  Section:           24 bytes            BINDING [ B ]
    Externals in: 1 Addresses in: 1
  Section:           125 bytes            WORKING STORAGE [ R ]
    Bytes initialized: 101 Addresses in: 4 Addresses to: 2
  Section:           64 bytes            WORKING STORAGE [ R W ]
    Bytes initialized: 12 Addresses in: 2 Addresses to: 5

AOL/
```

QUIT
ANAOL Subcommand

Purpose Ends the ANALYZE_OBJECT_LIBRARY session.

Format **QUIT** or
QUI

Parameters None.

Remarks For more information, see the NOS/VE Object Code Management manual.

Examples The following sequence writes a library and a module analysis of LIBRARY_1 to file OUT1 and writes a library analysis of OBJECT_FILE_2 to file OUT2. The output files are then printed.

```
/analyze_object_library library_1
AOL/display_library_analysis output=out1
AOL/display_module_analysis display_option=..
AOL../section_analysis output=out1.$eoi
AOL/use_library object_file_2
AOL/display_library_analysis output=out2
AOL/quit
/print_file out1
/print_file out2
```

USE_LIBRARY
ANAOL Subcommand

Purpose Specifies the object library or object file to be analyzed.

Format

USE_LIBRARY or
USEL
LIBRARY=file
STATUS=status variable

Parameters

LIBRARY or **L**

Object library or object file to be analyzed. This parameter is required.

Remarks

- If an object library or object file was not specified on the ANALYZE_OBJECT_LIBRARY command, you must specify the library or file with the USE_LIBRARY subcommand before you can analyze the library, its modules, or its sections.
- You use this subcommand to specify a new object library or object file to analyze.
- For more information, see the NOS/VE Object Code Management manual.

Examples The following subcommand selects object file LGO as the next library to be analyzed.

```
AOL/use_library lgo
```

BACKUP_PERMANENT_FILES 8

BACKUP_PERMANENT_FILES Command

Purpose Initiates execution of the utility that backs up permanent files and catalogs. Further processing is directed by utility subcommands.

Format

BACKUP_PERMANENT_FILES or
BACKUP_PERMANENT_FILE or
BACPF
BACKUP_FILE=file
LIST=file
STATUS=status variable

Parameters

BACKUP_FILE or **BF**

Specifies the file to which backup information is copied. You can specify a file position of beginning-of-information or end-of-information if the file is a mass storage file or a labelled tape. If no file position is specified, or the file is an unlabelled tape, the file is initially positioned to beginning-of-information. This parameter is required.

LIST or *L*

Identifies the file to which a summary of the results of executing the backup utility is written and, optionally, specifies how the file is to be positioned prior to use. Omission causes $LIST to be used.

Remarks

- You can back up only the files for which you have read access.
- For more information, see the NOS/VE System Usage manual.

Examples The following command initiates a BACKUP_PERMANENT_FILE command utility session. The command specifies that the backed up files are to be written to file BACKED_UP_FILES with the report listing written to file BACKUP_LISTING.

```
/backup_permanent_files bf=backed_up_files ..
../l=backup_listing
```

Following the entry of this command, BACKUP_PERMANENT_FILE subcommands can be entered in response to the following prompt.

```
PUB/
```

BACKUP_CATALOG BACPF Subcommand

Purpose Creates a backup copy of each file cycle and catalog registered in a specified catalog.

Format

BACKUP_CATALOG or
BACC
CATALOG=file
STATUS=status variable

Parameters

CATALOG or **C**

Specifies the catalog to be backed up. This parameter is required.

Remarks

- Starting at the specified catalog, the complete catalog hierarchy is followed to obtain a backup copy of each file and its associated catalog information.
- You must have READ access to the files in the catalog to be backed up and not be required to share the files for APPEND, MODIFY or SHORTEN access.
- If you are not the owner of the catalog, back up copies for all file cycles (and their associated catalogs) to which you have read access and only for those files that have null passwords are made.
- BACKUP_CATALOG skips a file cycle if the file cycle is busy (that is, if it cannot access the file with an access mode of read and a share mode of read and execute).
- Previous EXCLUDE_CATALOG and EXCLUDE_FILE subcommands enable you to exclude catalogs and files from the backup operation.

- Previous INCLUDE_CYCLES, INCLUDE_VOLUME, INCLUDE_LARGE_CYCLES, and EXCLUDE_HIGHEST_CYCLE subcommands can limit the number of cycles actually backed up with the BACKUP_CATALOG subcommand.
- For more information, see the NOS/VE System Usage manual.

Examples The following command and subcommands back up all files in the master catalog:

```
/backup_permanent_files bf=back_up_files ..
PUB../list=backup_listing
PUB/backup_catalog c=$user
PUB/quit
```


BACKUP_FILE
BACPF Subcommand

Purpose Creates a backup copy of a specified permanent file.

Format

BACKUP_FILE or
BACF
FILE=file
PASSWORD=keyword or *name*
STATUS=status variable

Parameters

FILE or **F**

Specifies the permanent file or permanent file cycle for which a backup copy is to be made. This parameter is required.

PASSWORD or *PW*

Parameter Attributes: SECURE

Specifies the password of the file to be backed up. If you omit this parameter or specify the keyword NONE, no password is used.

Remarks

- If the FILE parameter specifies a cycle reference, only that cycle is backed up. If a cycle reference is omitted, all cycles of the file are backed up.
- You must have READ access to the files to be backed up and not be required to share the files for APPEND, MODIFY, or SHORTEN access.
- BACKUP_FILE skips a file cycle if the file cycle is busy (that is, if it cannot access the file with an access mode of read and a share mode of read and execute).
- A previous EXCLUDE_FILE subcommand can be used to exclude specific cycles from the backup operation.
- Previous INCLUDE_CYCLES, INCLUDE_VOLUME, INCLUDE_LARGE_CYCLES, and EXCLUDE_HIGHEST_CYCLE subcommands can limit the number of cycles actually backed up with the BACKUP_FILE subcommand.
- For more information, see the NOS/VE System Usage manual.

Examples

The following example backs up cycle number 87 of file DATA_FILE_0 in subcatalog CATALOG_1 of the master catalog:

```
/bacpf bf=copy_of_file
PUB/backup_file $user.catalog_1.data_file_0.87 ..
PUB../pw=new_data_0_pw
PUB/quit
```

DELETE_CATALOG_CONTENTS BACPF Subcommand

Purpose

Deletes all files and subcatalogs in a catalog.

Format

DELETE_CATALOG_CONTENTS or
DELETE_CATALOG_CONTENT or
DELCC
CATALOG=file
STATUS=status variable

Parameters

CATALOG or C

Specifies the catalog whose contents is to be deleted. This parameter is required.

Remarks

- Only the owner of a catalog can use this subcommand to delete a catalog and to delete files that do not have passwords.
- Alternate users can use this request to delete all files:
  - To which they have control and read access permission.
  - That they are not required to share for modify, shorten, and append access.
  - That have null passwords.
- If a file cycle is in use at the time this subcommand is entered, the actual delete is not done until the last user detaches the file.
- Previous EXCLUDE_CATALOG, EXCLUDE_FILE, EXCLUDE_HIGHEST_CYCLES, INCLUDE_CYCLES, INCLUDE_LARGE_CYCLES, INCLUDE_VOLUME, and INCLUDE_EMPTY_CATALOGS subcommands can be used to specify a subset of the permanent files to be deleted.
- DELETE_CATALOG_CONTENT skips a file cycle if the file cycle is busy (that is, if it cannot access the file with an access mode of read and a share mode of read and execute).
- You can obtain the same results by specifying the keyword CAC on the DELETE_OPTION parameter of the SCL command DELETE_CATALOG.
- For more information, see the NOS/VE System Usage manual.

Examples

The following example deletes the contents of catalog CATALOG_1 for the current user:

```
/backup_permanent_files bf=backup_of_files
PUB/delcc $user.catalog_1
```

DELETE_FILE_CONTENTS BACPF Subcommand

Purpose Deletes all cycles of a file.

Format

DELETE_FILE_CONTENTS or
DELETE_FILE_CONTENT or
DELFC
FILE=file
PASSWORD=keyword or *name*
STATUS=status variable

Parameters

FILE or **F**

Specifies the file to be deleted. The cycle number is ignored. This parameter is required.

PASSWORD or *PW*

Parameter Attributes: SECURE

Specifies the file password of the file to be deleted. This name must match the password registered with the file. Omission or specifying the keyword NONE causes no password to be used.

Remarks

- Only the owner of the file or a user with control and read access permission and a share mode permission that does not include modify, shorten, or append can delete a file.
- DELETE_FILE_CONTENT skips a file cycle if the file cycle is busy (that is, if it cannot access the file with an access mode of read and a share mode of read and execute).
- If a file cycle is in use at the time this subcommand is entered, the actual delete is not done until the last user detaches the file.
- Previous EXCLUDE_FILE, EXCLUDE_HIGHEST_CYCLES, INCLUDE_VOLUME, INCLUDE_LARGE_CYCLES, and INCLUDE_CYCLES subcommands can be used to specify a subset of the permanent file cycles to be deleted.
- For more information, see the NOS/VE System Usage manual.

Examples The following example deletes all cycles of permanent file DATA_FILE_1 for the current user:

```
/bacpf backup_of_files
PUB/delete_file_contents $user.data_file_1
```

EXCLUDE_CATALOG BACPF Subcommand

Purpose Excludes a catalog from subsequent backup and delete operations.

Format

EXCLUDE_CATALOG or
EXCC
CATALOG=file
STATUS=status variable

Parameters

CATALOG or C

Specifies the catalog that is to be excluded from subsequent backup and delete operations. This parameter is required.

Remarks

- This subcommand takes precedence over all INCLUDE subcommands.
- The catalog is excluded only if the subsequent backup operation is at a higher level in the catalog hierarchy; thus, you can override this subcommand by explicitly backing up a catalog that is at a lower level in the catalog hierarchy.
- For more information, see the NOS/VE System Usage manual.

EXCLUDE_FILE BACPF Subcommand

Purpose Excludes a file or cycle from subsequent backup and delete operations.

Format

EXCLUDE_FILE or
EXCF
FILE=file
STATUS=status variable

Parameters **FILE or F**

Specifies the file or cycle that is to be excluded from subsequent backup and delete operations. This parameter is required.

Remarks

- This subcommand takes precedence over all INCLUDE subcommands.
- The file or cycle is excluded only if the subsequent backup or delete operation is at a higher level in the catalog hierarchy; thus, you can override this subcommand by explicitly backing up the file or cycle.
- For more information, see the NOS/VE System Usage manual.

EXCLUDE_HIGHEST_CYCLES BACPF Subcommand

Purpose Causes the specified number of high (largest numbered) cycles of permanent files to be excluded from subsequent backup and delete operations.

Format

EXCLUDE_HIGHEST_CYCLES or
EXCLUDE_HIGHEST_CYCLE or
EXCHC
NUMBER_OF_CYCLES=keyword or *integer*
STATUS=status variable

Parameters *NUMBER_OF_CYCLES* or *NOC*

Specifies the number of high cycles to be excluded. The value must be an integer in the range from 0 through 999. Omission causes 3 to be used.

Remarks

- This subcommand takes precedence over all INCLUDE subcommands.
- For more information, see the NOS/VE System Usage manual.

Examples The following example excludes the highest cycle of each file in a user's catalog from a subsequent DELETE_CATALOG_CONTENTS command:

```
/bacpf bf=backup_of_files
PUB/exclude_highest_cycles noc=1
PUB/delete_catalog_contents $user
```

INCLUDE_CYCLES BACPF Subcommand

Purpose Includes cycles in subsequent backup and delete operations based on the creation date and time, last access date and time, last modification date and time, or expiration date of the cycle.

Format

INCLUDE_CYCLES or
INCLUDE_CYCLE or
INCC
SELECTION_CRITERIA=keyword
AFTER=date_time
BEFORE=date_time
STATUS=status variable

Parameters

SELECTION_CRITERIA or **SC**

Specifies the selection criteria to be used in determining which cycles will be backed up on subsequent backup and delete operations. Choose one of the following:

ACCESSED (A)

Selects files based on the date and time they were last accessed.

CREATED (C)

Selects files based on the date and time they were created.

EXPIRED (E)

Selects files based on their expiration dates and times.

MODIFIED (M)

Selects files based on the date and time they were last modified.

IGNORE_DATE_TIME (IDT)

Do not select files based on a date and time. This option turns off any criteria that may have been selected in previous INCLUDE_CYCLES commands.

This parameter is required.

AFTER or *A*

Specifies the date and time after which the SELECTION_CRITERIA operation must have occurred in order for a file to be included in subsequent backup and delete operations. If omitted, 1980-01-01.00:00:00.000 is used.

BEFORE or *B*

Specifies the date and time before which the SELECTION_CRITERIA operation must have occurred in order for a file to be included in subsequent backup and delete operations. If omitted, $NOW is used.

Remarks

- The values specified on this command take precedence over any previous calls to INCLUDE_CYCLES.
- For more information, see the NOS/VE System Usage manual.

INCLUDE_EMPTY_CATALOGS BACPF Subcommand

Purpose Specifies whether or not subsequent DELETE_CATALOG_CONTENTS subcommands should delete empty catalogs.

Format

INCLUDE_EMPTY_CATALOGS or
INCLUDE_EMPTY_CATALOG or
INCEC
DELETE_CATALOGS=boolean
STATUS=status variable

Parameters

DELETE_CATALOGS or *DELETE_CATALOG* or *DC*

Specifies whether or not empty catalogs encountered during a subsequent DELETE_ALL_FILES or DELETE_CATALOG_CONTENTS subcommand should be deleted. Omission causes TRUE to used.

Remarks

- This subcommand must be entered during a BACKUP_PERMANENT_FILES command utility session.
- If this subcommand is not issued prior to a DELETE_ALL_FILES or DELETE_CATALOG_CONTENTS subcommand, empty catalogs are not deleted when those subcommands are entered.
- For more information, see the NOS/VE System Usage manual.

Examples

- The following example deletes all catalogs in subcatalog CATALOG_1 of a user's master catalog:

```
PUB/include_empty_catalogs
PUB/delete_catalog_contents ..
PUB../$user.catalog_1
```

- The following example saves empty catalogs from being deleted for user DLH in family FAMILY1:

```
PUB/include_empty_catalogs dc=false
PUB/delete_catalog_contents :family1.dlh
```

INCLUDE_LARGE_CYCLES BACPF Subcommand

Purpose

Specifies that subsequent backup and delete operations should include only permanent file cycles whose size is greater than or equal to a specified number of bytes. An excluded cycle is not backed up or deleted, regardless of its size.

Format

INCLUDE_LARGE_CYCLES or
INCLUDE_LARGE_CYCLE or
INCLC
MINIMUM_SIZE=*integer*
STATUS=status variable

Parameters

MINIMUM_SIZE or **MS**

Specifies the minimum size in bytes of cycles included on subsequent backup and delete operations. This parameter is required.

Remarks For more information, see the NOS/VE System Usage manual.

Examples The following example backs up and deletes all cycles greater than or equal to 1,000,000 bytes in size:

```
PUB/include_large_cycles ms=1000000
PUB/backup_catalog c=$user
PUB/delete_all_files
```


INCLUDE_SMALL_CYCLES BACPF Subcommand

Purpose Specifies that subsequent backup and delete operations should include only permanent file cycles whose size is less than or equal to a specified number of bytes. An excluded cycle is not backed up or deleted, regardless of its size.

Format

INCLUDE_SMALL_CYCLES or
INCLUDE_SMALL_CYCLE or
INCSC
MAXIMUM_SIZE=keyword or **integer**
STATUS=status variable

Parameters

MAXIMUM_SIZE or **MS**

Specifies the maximum size in bytes of cycles included on subsequent backup and delete operations. The keyword MAXIMUM specifies that no limit is placed on the size of cycles included in subsequent backup commands. This parameter is required.

Remarks For more information, see the NOS/VE System Usage manual.

Examples The following example backs up and deletes all cycles less than or equal to 1,000,000 bytes in size:

```
PUB/include_small_cycles ms=1000000
PUB/backup_catalog c=$user
PUB/delete_all_files
```

INCLUDE_VOLUMES BACPF Subcommand

Purpose Specifies which permanent file cycles included in a specified volume are to be backed up or deleted by subsequent backup operations.

Format

INCLUDE_VOLUMES or
INCLUDE_VOLUME or
INCV
RECORDED_VSNS=list of: keyword or **name**
CYCLE_SELECTION=keyword
STATUS=status variable

Parameters

RECORDED_VSNS or **RECORDED_VSN** or **RVSN**

Specifies the volumes to include; must be a name of from 1 to 6 characters or the keyword ALL. The RECORDED_VSN specified when the volume was initialized must be supplied. This parameter is required.

CYCLE_SELECTION or *CS*

Specifies which cycles on a volume should be backed up. Options are:

INITIAL_VOLUME (IV)

Back up only the cycles whose beginning of information (BOI) is on the volume. Cycles whose BOI is on another volume are skipped.

MULTIPLE_VOLUMES (MV)

Back up all cycles which reside either partially or completely on the volume.

If CYCLE_SELECTION is omitted, MULTIPLE_VOLUMES is used.

Remarks

- The CYCLE_SELECTION parameter is ignored when the keyword ALL is specified on the RECORDED_VSN parameter.
- If you select the MULTIPLE_VOLUMES option and cycles reside on more than one volume and each volume is backed up by a different backup, then cycles will be redundantly backed up. If the system fails due to a permanent file device failure, you may reload the

lost cycles with the Permanent File Restore utility's RESTORE_EXCLUDED_ FILE_CYCLES subcommand on just the backup tapes containing the cycles of the failed device.

- If you select the INITIAL_VOLUME option, data will not be redundantly backed up. Hence, all volumes in a backup must be read when a restore operation is done after a device failure.
- For more information, see the NOS/VE System Usage manual.

Examples The following example backs up all files that reside on the disk volume VOL033 and then deletes and restores the files so that they are dispersed over all volumes in the permanent file system:

```
/backup_permanent_files bf=temp_backup
PUB/include_volume rvsn=VOL033 cs=mv
PUB/backup_catalog $user
PUB/delete_catalog_contents $user
PUB/quit
/restore_permanent_files
PUR/restore_existing_catalog ..
PUR../$user bf=temp_backup
PUR/quit
/
```

QUIT
BACPF Subcommand

Purpose Ends a BACKUP_PERMANENT_FILES utility session.

Format **QUIT** or
QUI

Parameters None.

Remarks For more information, see the NOS/VE System Usage manual.

SET_BACKUP_OPTIONS BACPF Subcommand

Purpose Specifies actions for the BACKUP_PERMANENT_FILE utility.

Format

SET_BACKUP_OPTIONS or
SET_BACKUP_OPTION or
SETBO
EXCLUDE_CATALOG_INFORMATION=boolean
NULL_BACKUP_FILE_OPTION=keyword
INCLUDE_ARCHIVE_INFORMATION=boolean
INCLUDE_DATA=list of keyword
STATUS=status variable

Parameters

EXCLUDE_CATALOG_INFORMATION or *ECI*

Reserved for the site administrator's use. For more information, see the NOS/VE System Performance and Maintenance manual, Volume 2.

NULL_BACKUP_FILE_OPTION or *NBFO*

Specifies whether to read file data backups to $NULL or to any file assigned to the NULL device class. This parameter has no effect unless $NULL is specified on the BACKUP_FILE parameter of the BACKUP_PERMANENT_FILES command. Specify one of the following values:

READ_DATA or RD

Reads all file data when backing up to $NULL.

UNSPECIFIED

Does not read all file data when backing up to $NULL, but generates a listing of the file base.

The default is the previously specified value for this parameter. If none exists, the default is UNSPECIFIED.

INCLUDE_ARCHIVE_INFORMATION or *IAI*

Reserved for site personnel.

INCLUDE_DATA or ID

Specifies the file cycle data to include in the backup based on archive status and storage location (disk or archive medium). Specify one or more of the following values:

UNRELEASABLE_DATA or UD

Includes data for file cycles never duplicated and for file cycles modified since they were last duplicated.

RELEASABLE_DATA or RD

Includes data for file cycles not modified since they were last duplicated.

OFFLINE_DATA or OD

Includes data for file cycles released from mass storage and residing on an archive medium.

ALL

Includes data for all file cycles making no distinctions based on archive status or storage location.

The default is UNRELEASABLE_DATA and RELEASABLE_DATA.

Remarks

- We recommend that you specify parameters by name rather than by position (we anticipate adding parameters at a future date).
- When you specify INCLUDE_DATA=OFFLINE_DATA, the system retrieves archived file cycle data as it is attached for the backup. The backup continues after the data is retrieved, and the retrieved data is released from mass storage immediately after it is backed up.
- For more information, see the NOS/VE System Usage manual.

SET_LIST_OPTIONS BACPF Subcommand

Purpose Specifies the information that is written to the list file by subsequent subcommands.

Format

SET_LIST_OPTIONS or
SET_LIST_OPTION or
SETLO
FILE_DISPLAY_OPTIONS=list of keyword
CYCLE_DISPLAY_OPTIONS=list of keyword
DISPLAY_EXCLUDED_ITEMS=boolean
STATUS=status variable

Parameters

FILE_DISPLAY_OPTIONS or *FILE_DISPLAY_OPTION* or *FDO*

Selects the data to be displayed with the file name. Options are:

ACCOUNT (A)

Displays the account name.

PROJECT (P)

Displays the project name.

NONE

Displays only the file name.

ALL

Displays the account and project name.

Omission causes NONE to be used.

CYCLE_DISPLAY_OPTIONS or
CYCLE_DISPLAY_OPTION or *CDO*

Selects the data to be displayed for each cycle backed up. The cycle number and whether the cycle was excluded is also displayed. Options are:

ALL

Selects all of the following.

ACCESS_COUNT (AC)

Displays the number of accesses to the cycle.

ACCESS_DATE_TIME (ADT)

Displays the date and time the cycle was last accessed.

ALTERNATE_FILE_MEDIA_DESCRIPTOR (AFMD)

Displays archive information.

CREATION_DATE_TIME (CDT)

Displays the date and time the cycle was created.

EXPIRATION_DATE (ED)

Displays the expiration date of the cycle.

GLOBAL_FILE_NAME (GFN)

Displays the internally generated global file name. This name is neither backed up nor restored.

MODIFICATION_DATE_TIME (MDT)

Displays the date and time the cycle was last modified.

NONE

Displays the cycle number.

RECORDED_VSN (RVSN)

Displays all mass storage volumes on which the cycle resides.

SIZE (S)

Displays the size of the cycle in bytes.

Omission causes (MODIFICATION_DATE_TIME, SIZE) to be used.

DISPLAY_EXCLUDED_ITEMS or
DISPLAY_EXCLUDED_ITEM or *DEI*

Specifies whether excluded catalogs, files, and cycles are displayed on the list file.

TRUE

The identification of all excluded catalogs, files, and cycles is displayed. This is the default.

FALSE

Excluded items are not displayed.

Remarks For more information, see the NOS/VE System Usage manual.

Build Software Utility 9

BUILD_SOFTWARE Command

Purpose Initiates the Build Utility.

Format

BUILD_SOFTWARE or
BUIS
INPUT=file
BUILD_TARGETS=keyword or *list of file*
DECKS=keyword or *list of name*
EXECUTE_TRANSFORMATIONS=boolean
DISPLAY_OPTIONS=list of keyword or *keyword*
OUTPUT=file
ERRORS=file
STATUS=status variable

Parameters

INPUT or I

Specifies a file that describes your file system or library to BU. This file may contain NOS/VE commands and BU subcommands.

BUILD_TARGETS or *BUILD_TARGET* or *BT*

Specifies which build targets from the Input file should be analyzed.

You can reference the build targets by name or use one of the following keywords:

FIRST

Specifies the first build target in the Input file.

ALL

Specifies all build targets in the Input file.

DECKS or *DECK* or *D*

Specifies the decks to build.

You can specify a deck name, a list of deck names, or the keyword ALL.

ALL

Specifies all decks.

By default, BU determines which decks are out of date by comparing the date/time stamp on the source deck with the date/time stamp on the build target.

There are only three occasions when you will want to specify this parameter.

- When you want a full build.
- When you know that a deck was changed in such a way that the object code will not be affected. For example, changing a comment in a program does not affect the execution of the object code.
- When you know exactly which decks need to be built.

EXECUTE_TRANSFORMATIONS or *ET*

Specifies whether to execute the transformations for an out-of-date build target.

This parameter accepts a boolean value. If you specify FALSE, the transformations are not made. Only the analysis phase of the build is executed. This allows you to determine which decks would be built without actually performing a build.

The default value is TRUE.

By specifying the DISPLAY_OPTIONS parameter, you can cause BU to display the out-of-date build targets and the reasons they were found to be out of date.

DISPLAY_OPTIONS or *DO*

Specifies the information to display about the build. BU writes this information to the file specified by the OUTPUT parameter.

Specify one of the following keys.

ANALYSIS_TRACE or AT

Writes the steps taken by BU during the build.

ANALYSIS_RESULTS or AR

Writes the results of the build.

NONE

Indicates that no information is written.

The default value is NONE.

OUTPUT or *O*

Specifies the name of the file to which the information generated by the DISPLAY_OPTIONS parameter is written.

The default value is $OUTPUT.

ERRORS or *E*

Specifies the name of the file to which error messages are written.

The default value is $ERRORS.

Remarks This is the only BU command that can be entered directly from the command prompt.

Examples The following example uses the BUILD_SOFTWARE command to start BU. IFILE is the name of the BU Input file.

```
/build_software  input=ifile  display_options=analysis_trace
```

or abbreviated,

```
/buis  ifile  do=at
```

$ALTERNATE_SOURCE_LIBRARIES BUIS Function

Purpose Returns a list of the internal and external source libraries excluding the first internal source library given.

Format **$ALTERNATE_SOURCE_LIBRARIES** or **$ASL**

Parameters None.

Remarks If no alternate source libraries are specified, an empty list is returned.

Examples The following example uses the $alternate_source_libraries function.

```
define_source_libraries..
  internal_source_libraries=(lib1,lib2)

display_value $alternate_source_libraries
```

The result is:

```
:V01.kevin.lib2
```

$BASE_SOURCE_LIBRARY BUIS Function

Purpose Returns the name of the first internal source library defined in the DEFINE_SOURCE_LIBRARY command.

Format **$BASE_SOURCE_LIBRARY** or **$BSL**

Parameters None.

Examples The following example uses the $base_source_library function.

```
define_source_libraries..
  internal_source_libraries=(lib1,lib2)

display_value $base_source_library
```

The result is:

```
:V01.kevin.lib1
```

$BUILD_CATALOG BUIS Function

Purpose Returns the name of the catalog used during the build.

Format **$BUILD_CATALOG** or **$BC**

Parameters None.

Remarks If the Input file does not specify the SET_BUILD_CATALOG command, this function returns the name of the working catalog established when BU was initiated.

Examples The following example uses the $build_catalog function.

```
set_working_catalog..
  working_catalog=$user.exmps

display_value $build_catalog
```

The result is:

```
:V01.kevin.exmps
```

$BUILD_TARGET
BUIS Function

Purpose Returns the name of the build target whose transformation is currently executing.

Format **$BUILD_TARGET** or
$BT

Parameters None.

Examples The following example uses the $build_target function.

```
define_build_target..
  build_target=target1..
  build_target_kind=none..
     .
     .

display_value $build_target
```

The result is:

```
:V01.kevin.target1
```

$BUILD_TARGET_KIND BUIS Function

Purpose Returns the type of build target whose transformation is currently executing.

Format **$BUILD_TARGET_KIND** or
$BTK

Parameters None.

Examples The following example uses the $build_target_kind function.

```
define_build_target..
  build_target=lib1..
  build_target_kind=object_library
    .
    .

display_value $build_target_kind
```

The result is:

```
object_library
```

$BUILD_TARGET_LAYERS BUIS Function

Purpose Returns the file reference for every layer of the specified build target.

Format **$BUILD_TARGET_LAYERS** or
$BTL
(*BUILD_TARGET: file*)

Parameters *BUILD_TARGET*

Specifies the build target to use.

If the specified file is not a build target, the function returns the file that was given. If no file is specified, the function returns the name of the build target whose transformation is currently executing.

$CHANGED_DECKS
BUIS Function

Purpose Returns the names of the expandable decks that compose the build target whose transformation is currently executing.

Format **$CHANGED_DECKS** or
$CD

Parameters None.

Remarks The value returned depends on the value of the DECKS parameter of the BUILD_SOFTWARE command. If ALL was specified, the function returns a list of all decks in the current build target. If no value for the DECKS parameter was specified, the function returns a list of decks from the current build target that are out of date. If a deck name or list of decks was specified in the DECKS parameter, the function returns a list of these decks.

Examples The following example uses the $changed_decks function.

```
build_software i=infile d=(deck1 deck2 deck3)

display_value $changed_decks
```

The result is:

```
deck1
deck2
deck3
```

$COMPOSITION
BUIS Function

Purpose Returns a list of all decks that comprise the build target whose transformation is currently executing.

Format **$COMPOSITION** or
$C

Parameters None.

Remarks If the COMPOSITION parameter on the DEFINE_BUILD_TARGET command was not specified, no value is returned.

Examples The following example uses the $composition function.

```
define_build_target..
  build_target=target1..
  build_target_kind=OL..
  composition='incd d=(deck1 deck2)'..
     .
     .

display_value $composition
```

The result is:

```
deck1
deck2
```


$COMPOSITION_MAP BUIS Function

Purpose Returns a list of the decks and their corresponding object library entries that comprise the build target whose transformation is currently executing.

Format **$COMPOSITION_MAP** or **$CM**

Parameters None.

Remarks If the COMPOSITION_MAP parameter on the DEFINE_BUILD_TARGET command was not specified, an empty list is returned.

Examples This example uses the following composition map file:

```
deck1 ent1
deck2 ent2
deck3 ent3

display_value $composition_map
```

The result is:

```
deck1
ent1
deck2
ent2
deck3
```

DEFINE _BUILD _TARGET
BUIS Subcommand

Purpose Defines a build target by specifying the files it depends on and the transformation to be performed when the target is found to be out of date.

Format

DEFINE _BUILD _TARGET or
DEFBT
BUILD _TARGET=file
BUILD _TARGET _KIND=keyword or *name*
DEPENDENCES=list of file
COMPOSITION=keyword or *file* or *string*
COMPOSITION _MAP=file
LAYERS=list of file
TRANSFORMATION=keyword or **file** or **string**
STATUS=status variable

Parameters

BUILD _TARGET or **BT**

Specifies the build target name. A file name or library must be specified.

BUILD _TARGET _KIND or *BTK*

Specifies the type of the build target.

Specify an appropriate name or one of the following keywords:

OBJECT_LIBRARY or OL

Indicates that the build target type is an object library.

NONE

No type is assigned to the build target.

DEPENDENCES or *D*

Specifies a list of files that the build target depends upon.

Files that are specified in this parameter can also be build targets.

If omitted, BU assumes that the build target is dependent on decks rather than files and uses the COMPOSITION parameter to determine the decks.

COMPOSITION or *C*

Specifies the expandable decks that compose the build target.

Specify a string or a file containing SCU selection criteria commands, or the following keyword:

MAPPED_DECKS_ONLY or MDO

Indicates that the build target is only dependent on the decks specified by the COMPOSITION_MAP parameter.

If omitted, BU assumes that the build target is dependent on files rather than decks and uses the DEPENDENCES parameter to determine the files.

COMPOSITION_MAP or *CM*

Specifies a file containing a list of source decks mapped to object library entries. Each mapping has the following format:

```
deck:name=$required   object_library_entry:name=$required
```

If omitted, the name in the object library matches the name of the deck.

LAYERS or *L*

Specifies a list of files that comprise the layers of a system. These layers are searched in order starting with the build target itself to find the first occurrence of a module. BU uses this module as the basis for its analysis.

TRANSFORMATION or T

Specifies the transformation to perform when the build target is out of date.

Specify a string or file which contains one or more SCL commands, or the following keyword:

DEFAULT or D

Specifies that transformation is determined by the BUILD_TARGET_KIND parameter. In order to use DEFAULT for the TRANSFORMATION parameter, the BUILD_TARGET_KIND must not be NONE.

Remarks

- This command can only be used in a BU Input file.
- A build target can be any file, including an object library.
- A build target can be dependent on other build targets.
- You must specify either the DEPENDENCES parameter or the COMPOSITION parameter on this command.

Examples

The following example defines a build target named target1 as an object library.

```
define_build_target..
  build_target=target1..
  build_target_kind=object_library..
  composition='incd d=(deck1 deck2 deck3)'..
  transformation=default
```

or abbreviated,

```
defbt..
  bt=target1..
  btk=ol..
  c='incd d=(deck1 deck2 deck3)'..
  t=default
```

DEFINE_PARAMETER_LIST BUIS Subcommand

Purpose

Defines the parameters to pass to the specified processor during a transformation.

Format

DEFINE_PARAMETER_LIST or
DEFPL
PARAMETER_LIST_NAME=name
PARAMETER_LIST=string
PROCESSOR=name
STATUS=status variable

Parameters

PARAMETER_LIST_NAME or *PLN*

Specifies a name to associate with the parameter list. The name must be unique to this parameter list.

The default value is the name DEFAULT.

PARAMETER_LIST or **PL**

Specifies a string containing the parameters to pass to a given processor.

PROCESSOR or **P**

Specifies a processor to associate with the parameter list. The processor specified in this parameter must be defined with a DEFINE_PROCESSOR command *prior* to being referenced by this parameter.

Remarks

- This command can only be used in a BU Input file.

Examples

The following example defines a parameter list to pass to the COBOL processor:

```
define_parameter_list..
  parameter_list_name=plist1..
  parameter_list='i=compile bo=object_file'..
  processor=cobol
```

or abbreviated,

```
defpl..
  pln=plist1..
  pl='i=compile bo=object_file'..
  p=cobol
```

DEFINE_PROCESSOR BUIS Subcommand

Purpose Defines a processor to use during the execution of a transformation.

Format

DEFINE_PROCESSOR or
DEFP
PROCESSOR=name
PREPROCESSOR=keyword or *name*
DEFAULT_PARAMETER_LIST=name
STATUS=status variable

Parameters

PROCESSOR or **P**

Specifies the name of the processor to define.

PREPROCESSOR or *PP*

Specifies a preprocessor to use prior to executing the processor. A preprocessor prepares the source text for the main processor. An example of a preprocessor is DMFPC, which converts all the embedded DM commands in source code to FORTRAN code.

The default is NONE.

DEFAULT_PARAMETER_LIST or *DPL*

Specifies the name of the default parameter list to use for the processor. BU uses this parameter list when the processor attribute of a deck header is not specified.

The parameter list must be defined using the DEFINE_PARAMETER_LIST command.

By default, BU uses the parameter list name DEFAULT.

Remarks

- This command can only be used in a BU Input file.
- This command must be specified before specifying the DEFINE_PARAMETER_LIST command.

Examples The following example defines a COBOL processor and uses the default parameter list PLIST1:

```
define_processor..
  processor=cobol..
  default_parameter_list=plist1
```

or abbreviated,

```
defp..
  p=cobol..
  dpl=plist
```

DEFINE_SOURCE_LIBRARIES BUIS Subcommand

Purpose Specifies the internal and external source libraries to use during the build.

Format

DEFINE_SOURCE_LIBRARIES or
DEFSL
INTERNAL_SOURCE_LIBRARIES=list of file
EXTERNAL_SOURCE_LIBRARIES=list of file
ANALYZE_EXTERNAL_SOURCE=boolean
DEFAULT_PROCESSOR=name
STATUS=status variable

Parameters

INTERNAL_SOURCE_LIBRARIES or **ISL**

Specifies one or more source libraries to use during the build. BU searched these libraries in the order they are specified.

EXTERNAL_SOURCE_LIBRARIES or *ESL*

Specifies one or more source libraries containing decks that are external to the system being built, but are referenced by internal decks. BU searches these libraries in the order they are specified.

ANALYZE_EXTERNAL_SOURCE or *AES*

Specifies whether to include the external decks in the dependency analysis.

The default is FALSE.

DEFAULT_PROCESSOR or *DP*

Specifies the processor to use during a build when the processor attribute in a deck header is undefined.

The processor must be defined using the DEFINE_PROCESSOR command *prior* to being referenced by this parameter.

If omitted, the processor is defined by each deck's PROCESSOR attribute.

Remarks

- This command can only be used in a BU Input file.
- This command is a required component of the Input file when any of the build targets are defined as object libraries.

Examples

The following example defines a source library called source_lib and specifies COBOL as the default processor:

```
define_source_libraries..
  internal_source_libraries=source_lib..
  default_processor=cobol
```

or abbreviated,

```
defsl..
  isl=source_lib..
  dp=cobol
```

$DEPENDENCES BUIS Function

Purpose

Returns the list of files that the build target whose transformation is currently executing depends upon.

Format

$DEPENDENCES or
$D
(*KIND: keyword*)

Parameters

KIND

Specifies the files to return.

Specify one of the following keywords:

YOUNGER_THAN_TARGET or YTT

Returns only files that are younger than the target (the changed files).

ALL

Returns all files referenced in the DEPENDENCES parameter.

The default is YOUNGER_THAN_TARGET.

Examples The following example uses the $dependences function.

```
define_build_target..
  build_target=target1..
  build_target_kind=none..
  dependences=(file1 file2)
      .
      .

display_value $dependences
```

The result is:

```
:V01.kevin.file1
:V01.kevin.file2
```

$DISPLAY_OPTIONS
BUIS Function

Purpose Returns the display option specified for the build.

Format **$DISPLAY_OPTIONS** or
$DO

Parameters None.

Remarks If no display options were specified, an empty list is returned.

$ERRORS_FILE
BUIS Function

Purpose Returns the name of the file containing the error messages from the build.

Format **$ERRORS_FILE** or
$EF

Parameters None.

Examples The following example assumes that no errors file was specified on the BUILD_SOFTWARE command.

```
display_value $errors_file
```

The result is:

```
:$local.$error.1
```

$EXTERNAL_SOURCE_LIBRARIES BUIS Function

Purpose Returns a list of external source libraries specified for the build.

Format **$EXTERNAL_SOURCE_LIBRARIES** or **$ESL**

Parameters None.

Remarks If no external source libraries are specified, an empty list is returned.

Examples The following example uses the $external_source_libraries function.

```
define_source_libraries..
  internal_source_libraries=slib..
  external_source_libraries=(lib3 lib4)..
     .
     .

display_value $external_source_libraries
```

The result is:

```
:V01.kevin.lib3
:V01.kevin.lib4
```

$INTERNAL_SOURCE_LIBRARIES BUIS Function

Purpose Returns a list of internal source libraries specified for the build.

Format **$INTERNAL_SOURCE_LIBRARIES** or **$ISL**

Parameters None.

Examples The following example uses the $internal_source_libraries function.

```
define_source_libraries..
  internal_source_libraries=slib..
  external_source_libraries=(lib3 lib4)..
     .
     .

display_value $internal_source_libraries
```

The result is:

```
:V01.kevin.slib
```

$LAYERS BUIS Function

Purpose Returns a list of files that comprise the layers of the build target whose transformation is currently executing.

Format **$LAYERS** or **$L**

Parameters None.

Remarks If no layers are specified, an empty list is returned.

Examples The following example uses the $layers function.

```
define_build_target..
  build_target=target1..
  build_target_kind=object_library..
  layers=(file1 file2)
```

```
display_value $layers
```

The result is:

```
:V01.kevin.file1
:V01.kevin.file2
```

$OUTPUT_FILE BUIS Function

Purpose Returns the name of the output file specified for the build.

Format **$OUTPUT_FILE** or
$OF

Parameters None.

Remarks If no output file is specified, $OUTPUT is returned.

Examples The following example assumes that no output file was specified on the BUILD_SOFTWARE command.

```
display_value $output_file
```

The result is:

```
:$local.$output.1
```

$PARAMETER_LIST_VALUE BUIS Function

Purpose Returns a string containing the list of parameters to pass to the processor.

Format **$PARAMETER_LIST_VALUE** or
$PLV
(*PROCESSOR: keyword* or *name*
PARAMETER_LIST_NAME: keyword or *name*)

Parameters *PROCESSOR*

Specifies the name of the processor to use. To use the default processor established for this build, specify the keyword DEFAULT_PROCESSOR.

PARAMETER_LIST_NAME

Specifies the name of the parameter list. To use the default parameter list for the specified processor, specify the keyword DEFAULT_PARAMETER_LIST.

Examples The following example uses the $parameter_list_value function:

```
display_value $parameter_list_value(expand_source default)
```

The result is:

```
d=$changed_decks b=$base_source_library ab=$alternate_source_libraries l=
$output_file e=$errors_file
```


$PROCESSOR_ATTRIBUTE BUIS Function

Purpose Returns the name of the preprocessor or the default parameter list for the specified processor.

Format

$PROCESSOR_ATTRIBUTE or
$PA
(PROCESSOR: name
ATTRIBUTE: keyword)

Parameters

PROCESSOR

Specifies the name of the processor.

ATTRIBUTE

Specifies the processor attribute.

Enter one of the following keywords:

PREPROCESSOR or PP

Returns the name of the preprocessor associated with the specified processor. If the processor does not have a preprocessor assigned to it, NONE is returned.

DEFAULT_PARAMETER_LIST or DPL

Returns the name of the default parameter list for the specified processor. If no default parameter list is specified, UNDEFINED is returned.

Examples The following example uses the $processor_attribute function.

```
display_value $processor_attribute(expand_source default_parameter_list)
```

The result is:

```
default
```

QUIT_SAVE
BUIS Subcommand

Purpose Ends the BU session.

Format

QUIT_SAVE or
QUI or
QUIT or
QUIS
STATUS=status variable

SET_BUILD_CATALOG
BUIS Subcommand

Purpose Specifies the catalog to use during the build.

Format

SET_BUILD_CATALOG or
SETBC
BUILD_CATALOG=file
STATUS=status variable

Parameters *BUILD_CATALOG* or *BC*

Specifies the full path name of the catalog to use during the build.

Remarks

- If you omit this command, the catalog that was active when BU was initiated is used.
- This command can only be used in a BU Input file.

$UNKNOWN_LIBRARY_ENTRIES BUIS Function

Purpose Returns a list of all object library modules for which no source deck is present in the build target.

Format **$UNKNOWN_LIBRARY_ENTRIES** or **$ULE**

Parameters None.

Remarks If no unknown library modules are found, an empty list is returned.

CHANGE_KEYED_FILE and CREATE_KEYED_FILE 10

CHANGE_KEYED_FILE and CREATE_KEYED_FILE 10

CHANGE_KEYED_FILE Command

Purpose Begins a CHANGE_KEYED_FILE utility session.

Format

CHANGE_KEYED_FILE or
CHANGE_KEYED_FILES or
CHAKF
INPUT=file
OUTPUT=file
STATUS=status variable

Parameters

INPUT or **I**

File path of an existing keyed file. If an output file is specified, the input file is opened and copied to the output file and then closed.

This parameter is required.

OUTPUT or *O*

File path of the keyed file to which the input keyed file is copied. The output file must be a duplicate of the input file. If the output file does not exist, the command creates it.

If an output file is specified, only the output file is changed. If OUTPUT is omitted, the input file is changed.

Remarks

- The command utility prompt is:

  ```
  chakf/
  ```

  In response to the `chakf/ prompt`, you can enter SCL commands and any of these subcommands:

  ADD_RECORDS
  REPLACE_RECORDS
  COMBINE_RECORDS
  EXTRACT_RECORDS
  DELETE_RECORDS
  CREATE_NESTED_FILE
  SELECT_NESTED_FILE
  DELETE_NESTED_FILE

DISPLAY_NESTED_FILE
CREATE_ALTERNATE_INDEXES
HELP
QUIT

- All subcommands in the session apply to the currently selected nested file. The initially selected nested file is $MAIN_FILE. The nested file selection can be changed by a CREATE_NESTED_FILE or SELECT_NESTED_FILE subcommand.

- If the existing keyed file or a new nested file to be created uses a user-defined collation table, hashing procedure, or compression procedure, the object library containing the compiled table or procedure must be in the program library list before the CHANGE_KEYED_FILE session begins.

  To add one or more object libraries to the program library list, use the ADD_LIBRARIES parameter on a SET_PROGRAM ATTRIBUTES command. For example:

```
set_program_attributes, add_library=$user.hash_library
```

- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples The following session copies an existing keyed file and then ends.

```
/change_keyed_file, input=$user.existing_keyed_file, ..
../output=$user.new_keyed_file
chakf/quit
/
```

CREATE_KEYED_FILE Command

Purpose Begins a CREATE_KEYED_FILE utility session.

Format

CREATE_KEYED_FILE or
CREATE_KEYED_FILES or
CREKF
OUTPUT=file
STATUS=status variable

Parameters

OUTPUT or O

File path of the keyed file to be created. The keyed-file attributes must already be specified by SET_FILE_ATTRIBUTES commands.

This parameter is required.

The minimum attributes that must be defined are KEY_LENGTH and MAXIMUM_RECORD_LENGTH. If the FILE_ORGANIZATION is omitted, CREATE_KEYED_FILE creates an indexed-sequential file.

Remarks

- The command utility prompt is:

  ```
  crekf/
  ```

  In response to the `crekf/` prompt, you can enter SCL commands and any of these subcommands:

  ADD_RECORDS
  REPLACE_RECORDS
  COMBINE_RECORDS
  EXTRACT_RECORDS
  DISPLAY_RECORDS
  DELETE_RECORDS
  CREATE_NESTED_FILE
  SELECT_NESTED_FILE
  DELETE_NESTED_FILE
  DISPLAY_NESTED_FILE
  CREATE_ALTERNATE_INDEXES
  HELP
  QUIT

- The new keyed file is created with one nested file, named $MAIN_FILE. It is the initially selected nested file and all subcommands apply to it until a CREATE_NESTED_FILE or SELECT_NESTED_FILE subcommand selects another nested file.

- If any nested file in the new keyed file uses a user-defined collation table, hashing procedure, or compression procedure, the object library containing the compiled table or procedure must be in the program library list before the CREATE_KEYED_FILE session begins.

To add one or more object libraries to the program library list, use the ADD_LIBRARIES parameter on a SET_PROGRAM_ ATTRIBUTES command. For example:

```
set_program_attributes, add_library=$user.hash_library
```

- If you specify DIRECT_ACCESS as the FILE_ORGANIZATION attribute on the SET_FILE_ATTRIBUTES command, but omit the INITIAL_HOME_BLOCK_COUNT attribute, CREATE_KEYED_FILE prompts you for calculation of the INITIAL_HOME_BLOCK_COUNT.

- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This CREATE_KEYED_FILE example defines the file $USER.INDEXED_SEQUENTIAL_FILE with the SET_FILE_ATTRIBUTES command and then creates it.

```
/set_file_attributes, file=$user.indexed_sequential_file ..
../file_organization=indexed_sequential ..
../maximum_record_length=32, minimum_record_length=14 ..
../key_length=14
/create_keyed_file, output=$user.indexed_sequential_file
crekf/
```

ADD_RECORDS
CHAKF and CREKF Subcommand

Purpose Adds records to the currently selected nested file.

Format

ADD_RECORDS or
ADD_RECORD or
ADDR
INPUT=list of record
SORT=boolean
ERROR_LIMIT=integer
STATUS=status variable

Parameters

INPUT or I

List of one or more files whose records are to be copied. You must have at least read access to the files. This parameter is required.

To specify a nested file in a keyed file, enclose all elements of the list in parentheses. An element can be specified in one of the following ways:

- Enclose the file reference followed by the nested-file name in parentheses, or
- Enclose a comma followed by the nested-file name in parentheses. In this case, the file reference is the keyed file specified on the command utility (CREKF or CHAKF). Or
- Enclose a single comma in parentheses. In this case, the file reference is the keyed file specified on the command utility (CREKF or CHAKF), and the nested-file name specifies $MAIN_FILE.

SORT or *S*

Indicates whether the records are sorted before they are added to the file. (Sorting is recommended for better file performance.)

TRUE, ON, or YES

The records from the input file list are copied to a temporary file and sorted. Records for an indexed-sequential file are sorted by their primary-key value; records for a direct-access file are sorted by their hash value.

FALSE, OFF, or NO

The records are copied to a temporary file, but are not sorted.

If SORT is omitted, the default is TRUE.

ERROR_LIMIT or *EL*

Number of nonfatal errors required to force termination of the add (0 through 65535). A 0 sets an unlimited error limit.

If ERROR_LIMIT is omitted, 0 is used.

Remarks For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This Create_Keyed_File example creates the file $USER.INDEXED_SEQUENTIAL_FILE, adds the records of file $USER.ADD_RECORDS to it, and then displays the file.

```
/set_file_attributes ..
../file=$user.indexed_sequential_file ..
../file_organization=indexed_sequential ..
../maximum_record_length=32 ..
../minimum_record_length=14 ..
../key_length=14

/create_keyed_file ..
../output=$user.indexed_sequential_file
crekf/add_records input=$user.add_records

crekf/display_records count=all
                Display_Nested_File                        1986-02-17
NOS/VE Keyed File Utilities  1.2 85357                       11:19:36
File = :NVE.USER99.INDEXED_SEQUENTIAL_FILE.1
Display of records in $MAIN_FILE

 Byte: 0            ASCII: Everest       Asia          8848
 Byte: 0            ASCII: K2            Asia          8611
 Byte: 0            ASCII: Kilimanjaro   Africa        5895
 Byte: 0            ASCII: Matterhorn    Europe        4478
 Byte: 0            ASCII: McKinley      North America 6194
crekf/
```

COMBINE_RECORDS
CHAKF and CREKF Subcommand

Purpose Combines additional records with the records in the currently selected nested file.

Format

COMBINE_RECORDS or
COMBINE_RECORD or
COMR
INPUT=list of record
SORT=boolean
ERROR_LIMIT=integer
STATUS=status variable

Parameters

INPUT or I

List of one or more files whose records are to be copied. You must have at least read access to the files. This parameter is required.

To specify a nested file in a keyed file, enclose all elements of the list in parentheses. An element can be specified in one of the following ways:

- Enclose the file reference followed by the nested-file name in parentheses, or

- Enclose a comma followed by the nested-file name in parentheses. In this case, the file reference is the keyed file specified on the command utility (CREKF or CHAKF). Or
- Enclose a single comma in parentheses. In this case, the file reference is the keyed file specified on the command utility (CREKF or CHAKF), and the nested-file name specifies $MAIN_FILE.

SORT or *S*

Indicates whether the input records are sorted before they are combined. (Sorting is recommended for better file performance.)

TRUE

The records from the input file list are copied to a temporary file and sorted. Records for an indexed-sequential file are sorted by their primary-key value; records for a direct-access file are sorted by their hash value.

FALSE

The records are copied to a temporary file, but are not sorted.

If SORT is omitted, the default is TRUE.

ERROR_LIMIT or *EL*

Number of nonfatal errors required to force termination of the combine (0 through 65535). A 0 sets an unlimited error limit.

If ERROR_LIMIT is omitted, 0 is used.

Remarks For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This Create_Keyed_File example adds records that have a new primary key and replaces records that have an existing primary-key value.

```
/copy_keyed_file_add_file
Everest       Africa        8800
K2            Asia          8611
Kilimanjaro   Africa        5895

/copy_keyed_file combine_file
Everest       Asia          8848
Matterhorn    Europe        4478
McKinley      North America 6194

/create_keyed_file ...
../output=$user.indexed_sequential_file
crekf/add_records input-$user.add_file
crekf/combine_records input=$user.combine_file
crekf/display_records count=all
     Display_Nested_File                    1986-02-17
NOS/VE Keyed File Utilities  1.2 85357        12:01:46
File =:NVE.USER99.INDEXED_SEQUENTIAL_FILE.1
Display of records in $MAIN_FILE

 Byte: 0          ASCII: Everest       Asia          8848
 Byte: 0          ASCII: K2            Asia          8611
 Byte: 0          ASCII: Kilimanjaro   Africa        5895
 Byte: 0          ASCII: Matterhorn    Europe        4478
 Byte: 0          ASCII: McKinley      North America 6194
crekf/
```

CREATE_ALTERNATE_INDEXES CHAKF and CREKF Subcommand

10

Purpose Initiates execution of the CREATE_ALTERNATE_INDEXES command utility.

Format

CREATE_ALTERNATE_INDEXES or
CHANGE_ALTERNATE_INDEX or
CHANGE_ALTERNATE_INDEXES or
CHANGE_ALTERNATE_INDICES or
CREAI or
CREATE_ALTERNATE_INDEX or
CREATE_ALTERNATE_INDICES or
CHAAI
STATUS=status variable

Remarks

- The subutility prompt is:

  ```
  creai/
  ```

  In response to the creai/ prompt, you can enter NOS/VE commands and any of these subcommands:

  CREATE_KEY_DEFINITIONS

DISPLAY_KEY_DEFINITIONS
DELETE_KEY_DEFINITIONS
CANCEL_KEY_DEFINITIONS
APPLY_KEY_DEFINITIONS
HELP
QUIT

- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples The following subutility session creates an alternate-key definition and then displays it.

```
  crekf/creat_alternate_indexes
  creai/create_key_definitions ..
  creai../key_name=alternate_key_1 ..
  creai../key_position=28 key_length=4
  creai/display_key_definitions display_options=all
  Display_Nested_File                                    1986-02-17
NOS/VE Keyed File Utilities  1.2 86034                   12:20:26
File = :NVE.INDEXED_SEQUENTIAL_FILE
Nested_File_Name

KEY_NAME                      POSITION LENGTH   TYPE        STATE
--------------------------- -------- -------- ---------- ----------------
ALTERNATE_KEY_1                 28       4   uncollated  creation pending
 Duplicate_Key_Value        : not_allowed
 Null_Suppression           : no
================================================================================

RECORD 1 ...(in ascii) :E v e r e s t                A s i a
           ( in hex ) :4576657265737420202020202020417369612020202020

ALTERNATE_KEY_1            :

                (in ascii) :        8 8 4 8
                ( in hex ) :2020202038383438
                           >        U_U_U_U_
creai/
```

10

CREATE_NESTED_FILE
CHAKF and CREKF Subcommand

Purpose Creates and selects a new nested file.

Format

CREATE_NESTED_FILE or
CRENF
NAME=**name**
KEY_LENGTH=**integer**
KEY_POSITION=*integer*
KEY_TYPE=*keyword*
MAXIMUM_RECORD_LENGTH=**integer**
COLLATE_TABLE_NAME=*name*
COMPRESSION_PROCEDURE_NAME=*keyword* or

entry_point_reference
DATA_PADDING=integer
DYNAMIC_HOME_BLOCK_SPACE=boolean
EMBEDDED_KEY=boolean
FILE_ORGANIZATION=keyword
HASHING_PROCEDURE_NAME=keyword or
entry_point_reference
INDEX_PADDING=integer
INITIAL_HOME_BLOCK_COUNT=integer
LOADING_FACTOR=integer
MINIMUM_RECORD_LENGTH=integer
RECORDS_PER_BLOCK=integer
RECORD_TYPE=keyword
STATUS=status variable

Parameters

NAME or N

Name of the new nested file. It must be unique in the keyed file.

This parameter is required.

KEY_LENGTH or KL

Primary-key length in bytes (for integer keys, 1 through 8; for character keys from 1 through 255).

This parameter is required.

KEY_POSITION or KP

Position of the leftmost byte of the primary key (specified only if the key is embedded). The byte positions in a record are numbered from the left, from 0 through 65535, beginning with 0.

If KEY_POSITION is omitted, the default is 0.

KEY_TYPE or KT

Primary key type.

UNCOLLATED or UC

Key values ordered byte-by-byte according to the ASCII collating sequence.

INTEGER or I

Key values ordered numerically as integer values.

COLLATED or C

Key values ordered byte-by-byte according to the collating sequence specified by the COLLATE_TABLE_NAME parameter (invalid if FILE_ORGANIZATION = DIRECT_ACCESS).

If KEY_TYPE is omitted, the default is UNCOLLATED.

MAXIMUM_RECORD_LENGTH or **MAXRL**

Maximum number of bytes of data in a record (1 through 65497).

This parameter is required.

COLLATE_TABLE_NAME or *CTN*

Name of the collating sequence used to sort the primary key (indexed-sequential files only).

This parameter is required if the KEY_TYPE is COLLATED.

COMPRESSION_PROCEDURE_NAME or *CPN*

Name

Data compression or encryption procedure used with the nested file. The name can be either the name of the system-defined compression procedure (AMP$RECORD_ COMPRESSION or the name of an entry point in the current program library list.

NONE

No compression procedure is used with the nested file.

If COMPRESSION_PROCEDURE_NAME is omitted, the nested file does not use a compression procedure.

DATA_PADDING or *DP*

Percentage of data block space left empty when the indexed-sequential file is created (integer from 0 through 99).

The percentage must allow for storage of at least one maximum-length record per block.

If DATA_PADDING is omitted, the default is 0.

DYNAMIC_HOME_BLOCK_SPACE or *DHBS*

This parameter is reserved for future use. Its default value is FALSE.

EMBEDDED_KEY or *EK*

Indicates whether the primary-key value is embedded in the record data.

TRUE, ON, or YES

Primary-key value is embedded in the record data.

FALSE, OFF, or NO

Primary-key value is not part of the record data.

If EMBEDDED_KEY is omitted, the default is TRUE.

FILE_ORGANIZATION or *FO*

Keyed-file structure used.

INDEXED_SEQUENTIAL or IS

Data records accessed by searching for the primary-key value in a hierarchical index.

DIRECT_ACCESS or DA

Data record block accessed directly by hashed primary-key value.

If FILE_ORGANIZATION is omitted, the default is INDEXED_SEQUENTIAL.

HASHING_PROCEDURE_NAME or *HPN*

Name

Hashing procedure to be executed for the direct-access file.

NONE

No hashing procedure is executed with this direct-access file.

If HASHING_PROCEDURE_NAME is omitted, the default is the system-provided hashing procedure (named AMP$SYSTEM_HASHING_PROCEDURE).

INDEX_PADDING or *IP*

Percentage of index block space left empty when the indexed-sequential file is created (integer from 0 through 99).

The percentage must allow for storage of at least one index record per block. (The length of an index record is the key length plus 4.)

If INDEX_PADDING is omitted, the default is 0.

INITIAL_HOME_BLOCK_COUNT or *IHBC*

Number of home blocks to be created in the direct-access file (1 through 2**31-1).

This parameter is required when FILE_ORGANIZATION=DIRECT_ FILE_ORGANIZATION ACCESS.

LOADING_FACTOR or *LF*

Percentage of file space used when the direct-access file is created (no more than 90%).

If an initial home block count is specified, the loading factor is ignored. Otherwise, if LOADING_FACTOR is omitted, the default is 75%.

MINIMUM_RECORD_LENGTH or *MINRL*

Minimum number of bytes of data in a record (0 through 65497).

The minimum record length for a fixed-length record is the same as its maximum record length. The default minimum record length for variable-length records with an embedded key is the sum of the key_position and the key_length. Otherwise, the default minimum record length is 0.

RECORDS_PER_BLOCK or *RPB*

Reserved.

RECORD_TYPE or *RT*

Record type.

FIXED or F

Fixed-length records.

VARIABLE or V

Variable-length records.

UNDEFINED or U

Variable-length records.

If RECORD_TYPE is omitted, the default is UNDEFINED.

Remarks For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This Create_Keyed_File example creates a new nested file NESTED_FILE_1 and then displays the newly created file.

```
crekf/create_nested_file name=nested_file_1 ..
crekf../maximum_record_length=32, key_length=14 ..
crekf../file_organization=indexed_sequential
crekf/display_nested_file
   Display_Nested_File                    1986-02-17
NOS/VE Keyed File Utilities 1.2 85357       12:42:49
File = :NVE.INDEXED_SEQUENTIAL_FILE

List of Nested Files for file INDEXED_SEQUENTIAL_FILE
    NESTED_FILE_1              (currently selected nested file)
    $MAIN_FILE
```

10

DELETE_NESTED_FILE CHAKF and CREKF Subcommand

Purpose Deletes one or more nested files.

Format **DELETE_NESTED_FILE** or **DELNF**

NAMES=list of name

STATUS=status variable

Parameters **NAMES or NAME or N**

List of one or more nested files to be deleted.

This parameter is required.

Remarks

- You cannot delete the currently selected nested file or $MAIN_FILE.
- To delete the currently selected nested file, select another nested file first using the SELECT_NESTED_FILE subcommand and then issue the DELETE_NESTED_FILE subcommand.

- To display the names of the nested files, enter a DISPLAY_NESTED_FILE subcommand.
- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This Create_Keyed_File example displays the list of nested files and then deletes the nested file NESTED_FILE_2.

```
crekf/display_nested_file
Display_Nested_File                                    1986-02-17
NOS/VE Keyed File Utilities  1.2 85357                   12:50:12
File = :NVE.INDEXED_SEQUENTIAL_FILE

List of Nested Files for file INDEXED_SEQUENTIAL_FILE
    NESTED_FILE_1              (currently selected nested file)
    NESTED_FILE_2
    $MAIN_FILE

crekf/delete_nested_file name=nested_file_2
crekf/display_nested_file
                Display_Nested_File                    1986-02-17
NOS/VE Keyed File Utilities  1.2 85357                   12:52:02
File =:NVE.INDEXED_SEQUENTIAL_FILE

List of Nested Files for file INDEXED_SEQUENTIAL_FILE
    $MAIN_FILE                 (currently selected nested file)
    NESTED_FILE_1
```

DELETE_RECORDS
CHAKF and CREKF Subcommand

Purpose Deletes records from the currently selected nested file.

Format

DELETE_RECORDS or
DELETE_RECORD or
DELR
KEYS=range of: integer or *keyword range of: string* or *keyword*
COUNT=keyword or *integer*
VETO=boolean
STATUS=status variable

Parameters *KEYS* or *KEY* or *K*

Optional range of primary-key values to be deleted. The range may be specified as either:

1. Two primary-key values separated by two periods (..). (such as 'KEY1'..'KEY2'). The first key value must be less than the second. (Valid only for indexed-sequential files.)

2. One primary-key value specifying the beginning of the range. The number of records in the range is specified by the COUNT parameter.

The keywords $FIRST_KEY and $LAST_KEY can specify the lowest and highest key values, respectively, in an indexed-sequential file.

If KEYS is omitted, the range of records to be deleted begins with the first record in the nested file.

COUNT or *C*

Number of records to be deleted (0 through 4,398,046,511,103 or, to delete all records, the keyword ALL or A).

If a range is specified by the KEYS parameter, the COUNT value limits the number of records deleted.

If COUNT is omitted, but KEYS is specified, the default count the number of records in the specified range. Otherwise, the default is 1.

VETO or *V*

Indicates whether the interactive user must confirm each deletion.

TRUE

Each record to be deleted is displayed with the prompt Okay to delete?= = >.

FALSE

All specified records are deleted.

If VETO is omitted, the default is FALSE.

The possible responses to the veto prompt are:

YES or Y

Delete the record.

NO or N

Do not delete the record.

ALL or A

Delete the rest of the records without prompts.

QUIT or Q

Stop without deleting any more records.

HEX or H

Redisplays the record in hexadecimal and reissues the prompt.

Remarks For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This Create_Keyed_File example deletes a record in the currently selected nested file.

```
crekf/delete_records, keys='Matterhorn'..'McKinley' ..
crekf../count=2, veto=true
Byte: 0              ASCII: Matterhorn     Europe        4478
Okay to delete:  ==>Yes
Byte: 0              ASCII: McKinley       North America 6194
Okay to delete:  ==>No
--INFORMATIVE AA 501285-- As requested by the user, this record was not
deleted.
--INFORMATIVE AA 501285-- The Delete_Records subcommand of
CREATE_KEYED_FILE deleted 1 record from nested file $MAIN_FILE in file
:NVE.INDEXED_SEQUENTIAL_FILE.
crekf/
```


DISPLAY_NESTED_FILE CHAKF and CREKF Subcommand

Purpose Displays the nested file definitions and the alternate-key names and number of records in each nested file.

Format

DISPLAY_NESTED_FILE or
DISNF
NAMES=keyword or *list of name*
OUTPUT=file
DISPLAY_OPTIONS=keyword or *list of keyword*
STATUS=status variable

Parameters

NAMES or *NAME* or *N*

List of one or more names of nested files to be displayed or the keyword ALL to display all nested files in the file.

If NAMES is omitted, the default is ALL.

OUTPUT or *O*

File to which the display is written. The file must be a sequential file.

If OUTPUT is omitted, the default file is $OUTPUT.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

List of one or more keywords indicating the type of information to be displayed.

DEFINITIONS or DEFINITION or D

Nested-file definitions.

KEY_NAMES or KEY_NAME or KN

Names of the alternate keys in each nested file.

NAMES or NAME or N

Nested-file names.

RECORD_COUNTS or RECORD_COUNT or RC

Number of records in each nested file.

ALL or A

All of the above.

If DISPLAY_OPTION is omitted, the default is NAMES.

Remarks

- The currently selected nested file is marked as such in the list of nested files.
- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This Create_Keyed_File example displays the default nested file ($MAIN_FILE) with the DISPLAY_OPTIONS parameter set to ALL. No alternate keys have been defined.

```
crekf/display_nested_file Display_options=all
Display_Nested_File                                    1986-02-17
NOS/VE Keyed File Utilities  1.2 86034                   12:59:58
File = :NVE.INDEXED_SEQUENTIAL_FILE

$MAIN_FILE                      (currently selected nested file)
Record_Count                    : 3

Nested_File_Definitions
  Compression_Procedure_Name    : none
  Embedded_Key                  : yes
  Key-Position                  : 0
  Key-Length                    : 14
  Maximum_Record_Length         : 32
  Minimum_Record_Length         : 32
  Record_Type                   : undefined
  File_Organization             : indexed_sequential
  Key_Type                      : uncollated
  Collate_Table_Name            :
  Data_Padding                  : 0
  Index Padding                 : 0
```

DISPLAY_RECORDS
CHAKF and CREKF Subcommand

Purpose Displays records in the currently selected nested file.

Format

DISPLAY_RECORDS or
DISPLAY_RECORD or
DISR
OUTPUT=file
KEYS=range of: integer or *keyword range of: string* or *keyword*
COUNT=keyword or *integer*
DISPLAY_OPTION=keyword
STATUS=status variable

Parameters *OUTPUT* or *O*

File to which the display is written. The file must be a sequential file for which you have append access.

If OUTPUT is omitted, $OUTPUT is the default.

KEYS or *KEY* or *K*

Optional range of primary-key values to be displayed. The range may be specified as either:

1. Two primary-key values separated by two periods (..). (such as 'KEY1'..'KEY2'). The first key value must be less than the second. (Valid only for indexed-sequential files.)

2. One primary-key value specifying the beginning of the range. The number of records in the range is specified by the COUNT parameter.

The keywords $FIRST_KEY and $LAST_KEY can specify the lowest and highest key values, respectively, in an indexed-sequential file.

If KEYS is omitted, the range of records to be displayed begins with the first record in the nested file.

COUNT or *C*

Number of records to be displayed (0 through 4,398,046,511,103 or, to display all records, the keyword ALL or A).

If a range is specified by the KEYS parameter, the COUNT value limits the number of records displayed.

If COUNT is omitted, but KEYS is specified, the default count is the number of records in the specified range. Otherwise, the default is 1.

DISPLAY_OPTION or *DO*

List of one or more keywords indicating the representation used to display records.

ASCII

ASCII characters.

HEX or H

Hexadecimal digits.

BOTH

Both ASCII characters and hexadecimal digits.

ALTERNATE_KEY_DEFINITION or AKD or ALL

Both ASCII and hexadecimal representation with alternate-key values marked.

If DISPLAY_OPTION is omitted, the default is ASCII.

Remarks

- The ALTERNATE_KEY_DEFINITION display shows the record contents in ASCII characters and hexadecimal digits with the alternate-key values underscored.
- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples

The following session displays a range of records showing both ASCII and hexadecimal representations.

```
crekf/display_records display_option=both ..
crekf../keys='Everest'..'Kilimanjaro'
Display_Nested_File                                          1986-04-23
NOS/VE Keyed File Utilities 1.2 86099                         15:08:18
File = :NVE.USER99.INDEXED_SEQUENTIAL_FILE.1
Display of records in $MAIN_FILE for:

COUNT: all
FIRST_KEY: Everest
LAST_KEY: Kilimanjaro
Byte: 0          ASCII: E v e r e s t                 A s i a
Byte: 0(16)        HEX: 45766572657374202020202020204173696120202020202020
Byte: 25         ASCII:         8 8 4 8
Byte: 19(16)       HEX: 20202038303438
Byte: 0          ASCII: K 2                           A s i a
Byte: 0(16)        HEX: 4B32202020202020202020202020204173696120202020202020
Byte: 25         ASCII:         8 6 1 1
Byte: 19(16)       HEX: 20202038363131
Byte: 0          ASCII: K i l i m a n j a r o       A f r i c a
Byte: (16)         HEX: 4B696C696D616E6A61726F20202041667269636120202020202020
Byte: 25         ASCII:         5 8 9 5
Byte: 19(16)       HEX: 20202035383935
crekf/
```

EXTRACT_RECORDS CHAKF and CREKF Subcommand

Purpose Copies records from the currently selected nested file.

Format

EXTRACT_RECORDS or
EXTRACT_RECORD or
EXTR
OUTPUT=record
KEYS=range of: integer or *keyword range of: string* or *keyword*
COUNT=keyword or *integer*
ERROR_LIMIT=integer
STATUS=status variable

Parameters

OUTPUT or *O*

File to which records are copied. You must have at least append access to the file. If OUTPUT is omitted, the default is $OUTPUT.

To specify a nested file in a keyed file, you can:

- Enclose the file reference followed by the nested-file name in parentheses, or
- Enclose a comma followed by the nested-file name in parentheses. In this case, the file reference is the keyed file specified on the command utility (CREKF or CHAKF). Or
- Enclose a single comma in parentheses. In this case, the file reference is the keyed file specified on the command utility (CREKF or CHAKF), and the nested-file name specifies $MAIN_FILE.

KEYS or *KEY* or *K*

Optional range of primary-key values of the records to be copied. The range may be specified as either:

1. Two primary-key values separated by two periods (..) (such as 'KEY1'..'KEY2'). The first key value must be less than the second. (Valid only for indexed-sequential files.)

2. One primary-key value specifying the beginning of the range. The number of records in the range is specified by the COUNT parameter.

The keywords $FIRST_KEY and $LAST_KEY can specify the lowest and highest key values, respectively, in an indexed-sequential file.

If KEYS is omitted, the range of records to be copied begins with the first record in the nested file.

COUNT or *C*

Number of records to be copied (0 through 4,398,046,511,103 or, to copy all records, the keyword ALL or A).

If a range is specified by the KEYS parameter, the COUNT value limits the number of records copied.

If COUNT is omitted, but KEYS is specified, the default count is the number of records in the specified range. Otherwise, the default is 1.

ERROR_LIMIT or *EL*

Number of nonfatal (trivial) errors allows for the EXTRACT_RECORDS operation (integer from 0 through 65535). A 0 value indicates no limit; 0 is the default value.

Remarks

- Records are extracted only from the currently selected nested file.
- For more information, see the NOS/VE Advanced File Management Usage manual.

HELP CHAKF and CREKF Subcommand

Purpose Displays information about utility subcommands.

Format

HELP or
HEL
SUBJECT=string
MANUAL=keyword or *file*
STATUS=status variable

Parameters *SUBJECT* or *S*

Index topic to be located in the online manual.

If SUBJECT is omitted, the HELP subcommand lists the names of the utility subcommands.

MANUAL or *M*

Online manual whose index is searched.

AFM

The AFM online manual index is searched.

File

File name of the online manual whose index is searched.

If MANUAL is omitted, the default is AFM. The working catalog is searched for the file and then the $SYSTEM.MANUALS is searched.

Remarks

- If you enter a topic that is not in the manual index, a message appears telling you that the topic could not be found.
- The default manual, $SYSTEM.MANUALS.AFM, contains the online version of the NOS/VE Advanced File Management Usage manual, as provided with the NOS/VE system.
- If your terminal is defined for screen applications, the online manual is displayed in screen mode.

  To leave the online manual, use QUIT. To get help on reading the online manual, use HELP.
- For more information, see the NOS/VE Advanced File Management Usage manual.

QUIT
CHAKF and CREKF Subcommand

Purpose Ends the utility session and closes the output file.

Format

QUIT or
QUI
STATUS=status variable

REPLACE_RECORDS
CHAKF and CREKF Subcommand

Purpose Replaces existing records in the currently selected nested file.

Format

REPLACE_RECORDS or
REPLACE_RECORD or
REPR
INPUT=list of record
SORT=boolean
ERROR_LIMIT=integer
STATUS=status variable

Parameters

INPUT or **I**

List of one or more files whose records are to replace the corresponding records already in the keyed file. You must have at least read access to the input files. This parameter is required.

To specify a nested file in a keyed file, enclose all elements of the list in parentheses. An element can be specified in one of the following ways:

- Enclose the file reference followed by the nested-file name in parentheses, or
- Enclose a comma followed by the nested-file name in parentheses. In this case, the file reference is the keyed file specified on the command utility (CREKF or CHAKF). Or
- Enclose a single comma in parentheses. In this case, the file reference is the keyed file specified on the command utility (CREKF or CHAKF), and the nested-file name specifies $MAIN_FILE.

SORT or *S*

Indicates whether the records are sorted before they are copied to the file. (Sorting is recommended for better file performance.)

TRUE, ON, or YES

The records from the input file list are copied to a temporary file and sorted. Records for an indexed-sequential file are sorted by their primary-key value; records for a direct-access file are sorted by their hash value.

FALSE, OFF, or NO

The records are copied to a temporary file, but are not sorted.

If SORT is omitted, the default is TRUE.

ERROR_LIMIT or *EL*

Number of nonfatal errors required to force termination of the replace (0 through 65535). A 0 sets an unlimited error limit.

If ERROR_LIMIT is omitted, 0 is used.

Remarks For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This Create_Keyed_File example replaces records in file $USER.INDEXED_SEQUENTIAL_FILE that have the same primary key.

```
/copy_keyed_file $user.add_file
Everest       Africa        8800
K2            Asia          8611
Kilimanjaro   Africa        5895

/copy_keyed_file $user.replace_file
Everest       Asia          8848

/create_keyed_file ..
../output=$user.indexed_sequential_file
crekf/add_records input=$user.add_file
crekf/replace_records input=$user.replace_file
crekf/display_records count=all
Display_Nested_file                    1986-02-17
NOS/VE Keyed File Utilities  1.2 85357                    13:19:24
File = :NVE.USER99.INDEXED_SEQUENTIAL_FILE.1
Display of records in $MAIN_FILE

 Byte: 0            ASCII: Everest       Asia          8848
 Byte: 0            ASCII: K2            Asia          8611
 Byte: 0            ASCII: Kilimanjaro   Africa        5895
crekf/
```

SELECT_NESTED_FILE
CHAKF and CREKF Subcommand

Purpose Selects the nested file to which subsequent subcommands are to apply.

Format

SELECT_NESTED_FILE or
SELNF
NAME=name
STATUS=status variable

Parameters

NAME or **N**

Name of an existing nested file. To select the default nested file, specify $MAIN_FILE.

This parameter is required.

Remarks For more information, see the NOS/VE Advanced File Management Usage manual.

CREATE _ALTERNATE _INDEXES 11

CREATE_ALTERNATE_INDEXES Command

Purpose Initiates execution of the CREATE_ALTERNATE_INDEXES command utility. The utility can create, delete, and display alternate-key definitions in a keyed file.

Format

CREATE_ALTERNATE_INDEXES or
CREATE_ALTERNATE_INDEX or
CREATE_ALTERNATE_INDICES or
CREAI
INPUT=record
STATUS=status variable

Parameters

INPUT or I

Keyed file to be processed by the utility. The file permissions required depend on the subcommands entered during the utility as described in the Remarks. This parameter is required.

To specify a nested file, first specify the file reference and then the nested-file name, enclosed in parentheses.

Remarks

- The command utility prompt is:

  ```
  creai/
  ```

- In response to the creai/ prompt, you can enter NOS/VE commands and any of these subcommands:

  QUIT
  DISPLAY_KEY_DEFINITIONS
  CREATE_KEY_DEFINITION
  DELETE_KEY_DEFINITION
  CANCEL_KEY_DEFINITIONS
  APPLY_KEY_DEFINITIONS

- The CREATE_ALTERNATE_INDEXES utility creates the specified keyed file if:

  - The file does not exist and,

  - A SET_FILE_ATTRIBUTES command has specified the KEY_LENGTH and MAXIMUM_RECORD_LENGTH attributes for the file.

  If the SET_FILE_ATTRIBUTES command defining the new file omits an attribute, the default attribute value is used. However, if it omits the FILE_ORGANIZATION attribute, indexed-sequential organization is used.

- The CREATE_ALTERNATE_INDEXES command does not check your file permissions. The subcommands you enter in the utility session check that you have the required permissions to do the operation.

  To display key definitions, you must have at least read permission. To create, delete, cancel, or apply key definitions, you must have at least three permissions: append, modify, and shorten.

- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This command begins a utility session that displays the alternate key definitions of keyed file $USER.IS_FILE.

```
/create_alternate_indexes input=$user.is_file
creai/display_key_definitions key_names=all display_options=brief
       Display_Key_Definitions       NOS/VE Keyed File Utilities 1.1
File = :NVE.USER99.IS_FILE

 KEY NAME                 POSITION    LENGTH   TYPE         STATE
 ----------------------- --------    ------   ----------   --------------
 ALTERNATE_KEY_1                0        10   uncollated   Exists in file
creai/quit    "The APPLY_KEY_DEFINITIONS parameter is not required here
              "because no creation or deletion requests are pending.
```

ADD_PIECE
CREKD Subcommand

Purpose Defines a piece of a concatenated key within a CREATE_KEY_DEFINITION utility session.

Format

ADD_PIECE or
ADDP
POSITION=integer
LENGTH=integer
TYPE=keyword
STATUS=status variable

Parameters

POSITION or **KEY_POSITION** or **P** or **KP**

Byte position in the record at which the piece begins. The byte positions are numbered from the left, beginning with 0. The maximum byte position is 65496. This parameter is required.

LENGTH or **KEY_LENGTH** or **L** or **KL**

Number of bytes in the piece. The maximum length is 255 bytes. The piece must be within the minimum record length unless sparse-key control is used. This parameter is required.

TYPE or *KEY_TYPE* or *T* or *KT*

Type of the piece.

INTEGER (I)

Integer key ordered numerically.

UNCOLLATED (UC or U)

Character key ordered byte-by-byte according to the ASCII collating sequence.

COLLATED (C)

Character key ordered byte-by-byte according to the collation table specified by the COLLATE_TABLE_NAME parameter on the CREATE_KEY_DEFINITION command.

The default key type is UNCOLLATED.

Remarks

- The utility is initiated in response to a CREATE_KEY_DEFINITION subcommand that specifies the CONCATENATED_PIECES=TRUE parameter.
- To end concatenated-key specification, enter the QUIT subcommand for the CREATE_KEY_DEFINITION utility.
- You must enter an ADD_PIECE subcommand for each piece to be concatenated to the first piece to define a concatenated key. The first piece is defined by the KEY_LENGTH, KEY_POSITION, and KEY_TYPE parameters on the CREATE_KEY_DEFINITION command.
- A concatenated key can comprise from 2 through 64 pieces. The pieces are concatenated in the order that you enter the ADD_PIECE subcommands that define the pieces.
- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This CREATE_ALTERNATE_INDEXES session defines an alternate key that concatenates the first, third and fifth bytes of the record in reverse order. It displays the definition and then cancels the request.

```
/create_alternate_index input=$user.is_file
creai/create_key_definition key_name=alternate_key_2 ..
creai../key_position=4 key_length=1 concatenated_pieces=yes
crekd/add_piece key_position=2 key_length=1
crekd/addp kp=0 kl=1
crekd/quit
creai/display_key_definitions
                Display_Key_Definitions      NOS/VE Keyed File Utilities 1.1
 File = .NVE.USER99.IS_FILE
 KEY NAME                        POSITION LENGTH TYPE       STATE
 ------------------------------- -------- ------ ---------- ----------------
 ALTERNATE_KEY_2                        4      1 uncollated Creation pending
                         piece b        2      1 uncollated
                         piece c        0      1 uncollated
   Duplicate_Key_Values        : not_allowed
   Null_Suppression            : no
================================================================================
=====
RECORD 1 .......(in ascii) : T h i s   i s   t h e   f i r s t   r e c o r-
 d
                 ( in hex ) : 546869732069732074686520666972737420726 5636F72
ALTERNATE_KEY_2             : c_  b_  a_
                 (in ascii) : d .
                 ( in hex ) : 642E
                              >

creai/quit cancel
/
```

APPLY_KEY_DEFINITIONS CREAI Subcommand

Purpose Applies the pending alternate-key definition and deletion requests within a CREATE_ALTERNATE_INDEXES utility session.

Format

APPLY_KEY_DEFINITIONS or
APPLY_KEY_DEFINITION or
APPKD
ERROR_LIMIT=integer
STATUS=status variable

Parameters *ERROR_LIMIT* or *EL*

Number of trivial (nonfatal) errors allowed for the apply operation (integer from 0 through 65535).

A 0 value indicates no limit; 0 is the default value.

See Remarks for a description of apply error processing.

Remarks

- This CREATE_ALTERNATE_INDEXES subcommand applies all pending alternate-key creation and deletion requests to the file. It applies deletion requests first and then the creation requests.

- The ERROR_LIMIT file attribute value has no effect on keyed-file utility processing. This is done so that nonfatal errors (such as typing errors during interactive use) do not terminate the utility session.

  However, you can specify an error limit that applies to the apply operation only by specifying the ERROR_LIMIT parameter.

- The two nonfatal (trivial) errors that an apply operation can detect result from improper record data, as follows:

  Duplicate_Key_Value

  The duplicate_key_value attribute of the alternate index being built is NOT_ALLOWED, but the apply operation finds an alternate-key value matching an alternate-key value already in the alternate index.

  Sparse_Key_Beyond_EOR

  The apply operation is building an alternate index that uses sparse-key control and it finds a record for which an alternate-key value should be included in the index except that the record is too short to provide a complete alternate-key value.

- APPLY_KEY_DEFINITIONS keeps a count of the number of times it detects a nonfatal (trivial) error. Each time it increments the count, it checks whether the count has reached the value specified on the ERROR_LIMIT parameter.

  - If the error limit is not yet reached, APPLY_KEY_DEFINITIONS performs the correction processing. for the condition as described later.

  - If the error limit is reached, APPLY_KEY_DEFINITIONS terminates with a fatal error. The fatal error returned depends on the last nonfatal error detected:

For a Duplicate_Key_Value error, it returns AAE$DUPLICATE_KEY_LIMIT.

For a Sparse_Key_Beyond_EOR error, it returns AAE$ERROR_LIMIT_EXCEEDED.

- Before terminating, APPLY_KEY_DEFINITIONS discards all alternate indexes it has built. (Deleted alternate indexes are not restored.)
- If APPLY_KEY_DEFINITIONS finds one or more nonfatal errors, but completes its processing before reaching the error limit, it returns a warning message.
- As correction processing for a sparse_key_beyond_EOR error, APPLY_KEY_DEFINITIONS does not enter an alternate-key value for the record in the alternate index it is building, even though the sparse-key character indicates that a value should be entered for the record.
- As correction processing for a Duplicate_Key_Value error, APPLY_KEY_DEFINITIONS changes the duplicate_key_values attribute of the alternate-key definition from NOT_ALLOWED to ORDERED_BY_PRIMARY_KEY. It then discards the partially-built index and begins building the index again, ordering duplicate alternate-key values by their primary-key value.
- Entry of a pause-break character is ignored during application of alternate-key definitions.
- Entry of a terminate_break_character during application of alternate-key definitions returns a prompt to the terminal user, asking for confirmation.
- As described in the prompt, the terminal user should then enter a carriage return or any entry other than RUIN FILE (uppercase or lowercase) to continue the application of alternate-key definitions. Applied alternate-key definitions can be removed without harm to the file after the apply operation executes.

- A request to ruin the file is not recommended. No file operation can be performed on a ruined file; therefore, no data can be retrieved from the file.

- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This CREATE_ALTERNATE_INDEXES session attempts to create and apply an alternate key. The attempt fails when it finds a duplicate alternate-key value because the alternate-key definition does not allow duplicate values and the error limit for the apply is 1.

```
/create_alternate_indexes input=$user.is_file
creai/create_key_definition key_name=alternate_key_6 ..
creai../key_position=5 key_length=10
creai/apply_key_definition error_limit=1
-- File :NVE.USER99.IS_FILE : begin creating labels for
alternate key definitions.
-- File :NVE.USER99.IS_FILE : finished creating labels for
alternate key definitions.
-- File :NVE.USER99.IS_FILE : begin collecting the alternate
key values from the file.
-- File :NVE.USER99.IS_FILE : AMP$APPLY_KEY_DEFINITIONS has
reached a file boundary: EOI.
-- File :NVE.USER99.IS_FILE : collecting of the alternate key
values completed.
-- File :NVE.USER99.IS_FILE : begin sorting the alternate key
values.
-- File :NVE.USER99.IS_FILE : sorting of the alternate key
values completed.
-- File :NVE.USER99.IS_FILE : begin building alternate key
indexes into the file.
-- File :NVE.USER99.IS_FILE : the ALTERNATE_KEY_6 index
is being built.
-- File :NVE.USER99.IS_FILE : Alternate key ALTERNATE_KEY_6
has been deleted.
--ERROR-- File :NVE.USER99.IS_FILE :
AMP$APPLY_KEY_DEFINITIONS encountered a duplicate key
and found that error limit had been reached.  Because
ERROR_LIMIT was involved, any new indexes were removed
(though deleted indexes are gone).  Had ERROR_LIMIT not
been reached, the key definition would have been
modified to allow duplicates.  The duplicate key values
relate to alternate key name = ALTERNATE_KEY_6, primary
key = 96070, alternate_key_value = John Smith.
-- FATAL-- File  :NVE.USER99.IS_FILE :
AMP$APPLY_KEY_DEFINITIONS :  the user-declared maximum
number of trivial errors has been recorded since the
last OPEN.
creai/quit
/
```

CANCEL_KEY_DEFINITIONS CREAI Subcommand

Purpose Removes a pending request to create or delete an alternate key within a CREATE_ALTERNATE_INDEXES utility session.

Format

CANCEL_KEY_DEFINITIONS or
CANCEL_KEY_DEFINITION or
CANKD
NAMES=keyword or **list of name**
STATUS=status variable

Parameters **NAMES** or **KEY_NAME** or **KEY_NAMES** or **N** or **NAME** or **KN**

Pending requests to be canceled.

list of names

Cancel the requests for the listed alternate-key names.

ALL

Cancel all requests.

This parameter is required.

Remarks

- The CANCEL_KEY_DEFINITIONS subcommand can cancel creation and deletion requests only while they are pending.
- After a creation or deletion request is applied, the CANCEL_KEY_DEFINITIONS subcommand has no effect. To reverse the action of an APPLY_KEY_DEFINITIONS subcommand, you must issue new requests to delete the created alternate key or recreate the deleted alternate key.
- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This CREATE_ALTERNATE_INDEXES session requests creation of an alternate key and deletion of another alternate key, cancels the creation request, and finally applies the deletion request.

```
/create_alternate_indexes input=$user.is_file
creai/create_key_definition key_name=alternate_key_4 ..
creai../key_position=5 key_length=2
creai/delete_key_definition key_name=alternate_key_1
creai/cancel_key_definition alternate_key_4
creai/quit apply
-- File :NVE.USER99.IS_FILE : begin deleting alternate key
definitions.
-- File :NVE.USER99.IS_FILE : Alternate key ALTERNATE_KEY_1
has been deleted.
-- File :NVE.USER99.IS_FILE : end deleting alternate key
definitions.
/
```

CREATE_KEY_DEFINITION CREAI Subcommand

Purpose Creates a pending alternate-key definition within a CREATE_ALTERNATE_INDEXES utility session.

Format

CREATE_KEY_DEFINITION or
CREKD
NAME=name
POSITION=integer
LENGTH=integer
TYPE=keyword
COLLATE_TABLE_NAME=name
DUPLICATE_KEY_VALUES=keyword or *boolean*
NULL_SUPPRESSION=boolean
SPARSE_KEY_CONTROL_POSITION=integer
SPARSE_KEY_CONTROL_CHARACTERS=string
SPARSE_KEY_CONTROL_EFFECT=keyword
REPEATING_GROUP_LENGTH=integer
REPEATING_GROUP_COUNT=integer or *keyword*
GROUP_NAME=name
CONCATENATED_PIECES=boolean
VARIABLE_LENGTH_KEY=string
STATUS=status variable

Parameters

NAME or KEY_NAME or N or KN

Name of the new alternate key. The name must follow the SCL naming rules. This parameter is required.

POSITION or KEY_POSITION or P or KP

Byte position within the record at which the alternate-key field begins. The byte positions are numbered from the left, beginning with 0. The maximum byte position is 65496. This parameter is required.

LENGTH or KEY_LENGTH or L or KL

Number of bytes in the alternate-key field. The maximum length is 255 bytes. The key field must be within the minimum record length (unless sparse key control is used). This parameter is required.

TYPE or KEY_TYPE or T or KT

Type of the alternate key.

INTEGER (I)

Integer key ordered numerically.

UNCOLLATED (UC or U)

Character key ordered byte-by-byte according to the ASCII collating sequence.

COLLATED (C)

Character key ordered byte-by-byte according to the collation table specified by the COLLATE_TABLE_NAME parameter.

If the KEY_TYPE parameter is omitted, the key type is UNCOLLATED.

COLLATE_TABLE_NAME or CTN

Name of the collation table used to order the alternate key if its key type is collated. The collation table can be for NOS/VE predefined collating sequence or a user-defined collating sequence.

If the file is an indexed-sequential file with a collated primary key, the collation table for the primary key is used as the default collation table for an alternate key.

Otherwise, you must specify a collation table for a collated alternate key.

DUPLICATE_KEY_VALUES or *DKV*

Keyword value indicating whether duplicate alternate-key values are allowed and, if so, how the duplicate values are ordered.

NOT_ALLOWED (NA)

No duplicate values are allowed for the alternate key.

ORDERED_BY_PRIMARY_KEY (OBPK)

Duplicate values are allowed. Duplicates are accessed in order by their primary key.

FIRST_IN_FIRST_OUT (FIFO)

Duplicate values are allowed. Duplicates are accessed in the order of their primary-key value.

TRUE (ON or YES)

Duplicate values are allowed.

FALSE (OFF or NO)

No duplicates are allowed for the alternate key

If the DUPLICATE_KEY_VALUES parameter is omitted, no duplicate values are allowed.

NULL_SUPPRESSION or *S*

Reserved.

SPARSE_KEY_CONTROL_POSITION or *SKCP*

Byte position of the sparse-key control character. The position must be within the minimum record length. The byte positions are numbered from the left, beginning with 0. The maximum byte position is 65496.

NOTE

The two parameters, SPARSE_KEY_CONTROL_POSITION and SPARSE_KEY_CONTROL_CHARACTERS, work together; they must either both be specified or both be omitted. If they are omitted, sparse-key control is not used for the alternate key.

SPARSE_KEY_CONTROL_CHARACTERS or *SKCC*

String containing the set of characters with which the sparse-key control character in each record is compared.

SPARSE_KEY_CONTROL_EFFECT or *SKCE*

Indicates whether a sparse-key control character match causes the alternate-key value to be included in or excluded from the alternate index.

INCLUDE_KEY_VALUE (IKV)

The alternate-key value is included in the alternate index.

EXCLUDE_KEY_VALUE (EKV)

The alternate-key value is excluded from the alternate index.

You can specify the SPARSE_KEY_EFFECT parameter only if you specify the SPARSE_KEY_POSITION and SPARSE_KEY_CHARACTERS parameters.

If the SPARSE_KEY_CONTROL_EFFECT parameter is omitted, INCLUDE_KEY_VALUE is used.

REPEATING_GROUP_LENGTH or *RGL*

Length, in bytes of the repeating group of fields. It is the distance from the beginning of an alternate-key value to the beginning of the next value for the same alternate key in the same record.

The group length range is from 1 through 65497.

If the REPEATING_GROUP_LENGTH parameter is omitted, the alternate key has no more than one value per record.

REPEATING_GROUP_COUNT or *RGC*

Indicates how many alternate-key values are in a record. (The alternate-key value is in a repeating group of fields.)

integer (1 through 65497)

Number of times the alternate key occurs in a record. The specified number of alternate-key values must occur within the minimum record length.

REPEAT_TO_END_OF_RECORD (RTEOR)

The alternate key repeats until the record ends. (An incomplete key at the end of the record is not used.)

You can specify the REPEATING_GROUP_COUNT parameter only if you specify the REPEATING_GROUP_LENGTH parameter.

If the REPEATING_GROUP_COUNT parameter is omitted, the alternate key repeats until the end of the record.

GROUP_NAME or *GN* or *KEY_GROUP_NAME* or *KGN*

Name of the key group for this key. The key-grouping feature is not currently implemented. The default value for the key-group name is the key name.

CONCATENATED_PIECES or *CONCATENATED_PIECE* or *CP*

Indicates whether the alternate key is a concatenated key.

TRUE (ON or YES)

The key is a concatenated key.

FALSE (OFF or NO)

The key is not a concatenated key.

If you specify CONCATENATED_PIECES=TRUE, the CREATE_KEY_DEFINITION command initiates the CREATE_KEY_DEFINITION subcommand utility. The utility prompt is crekd/ and it processes ADD_PIECE, HELP, and QUIT subcommands.

If the CONCATENATED_PIECES parameter is omitted, the key is not a concatenated key.

VARIABLE_LENGTH_KEY or *VLK*

Indicates that the key is a variable_length key by specifying its set of delimiter characters. The set is specified as a string (0 through 256 characters, enclosed in apostrophes).

If the REPEATING_GROUP_LENGTH parameter is omitted, no more than one value for the key is taken from a record. The end of the value is marked by a delimiter character, by the end of the key field (KEY_LENGTH length), or by the end of the record, whichever occurs first after the KEY_POSITION.

If the REPEATING_GROUP_LENGTH parameter is specified, the record can contain more than one value for the key. Multiple key values are separated by one or more delimiter characters. The REPEATING_GROUP_COUNT parameter indicates whether the sequence of values continues to the end of the record or is limited to a fixed number of characters.

If VARIABLE_LENGTH_KEY is omitted, the alternate key has fixed-length values.

Remarks

- The CREATE_KEY_DEFINITION subcommand defines an alternate key but does not apply the definition to the file. The definition remains pending until it is either applied or canceled.
- A definition is applied by either an APPLY_KEY_DEFINITIONS subcommand or an APPLY_KEY_DEFINITIONS=YES parameter on the QUIT subcommand. It is canceled by a CANCEL_KEY_DEFINITIONS subcommand or an APPLY_KEY_DEFINITIONS=NO parameter on the QUIT subcommand.
- The REPEATING_GROUP_LENGTH and the VARIABLE_LENGTH_KEY parameters cannot be specified with either the CONCATENATED_PIECES parameter or the DUPLICATE_KEY_VALUES=FIRST_IN_FIRST_OUT parameter.
- If the alternate-key definition defines a collated key, CREATE_KEY_DEFINITIONS searches for the collation-table name as an entry point in the object libraries in the program-library list.
- You must set the program-library list before you enter the utility. You cannot change the object libraries searched from within the utility session.
- The following command adds an object library to the program-library list:

```
/set_program_attributes add_library=file_reference
```

- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This CREATE_ALTERNATE_INDEXES utility session creates and applies an alternate-key definition to file $USER.IS_FILE.

```
/create_alternate_index  input=$user.is_file
creai/create_key_definition  key_name=alternate_key_1 ..
creai../key_position=0  key_length=10
creai/quit  apply
-- File :NVE.USER99.IS_FILE : begin creating labels for
alternate key definitions.
-- File :NVE.USER99.IS_FILE : finished creating labels for
alternate key definitions.
-- File :NVE.USER99.IS_FILE : begin collecting the alternate
key values from the file.
-- File :NVE.USER99.IS_FILE : AMP$APPLY_KEY_DEFINITIONS has
reached a file boundary: EOI .
-- File :NVE.USER99.IS_FILE : collecting of the alternate key
values completed.
-- File :NVE.USER99.IS_FILE : begin sorting the alternate key
values.
-- File :NVE.USER99.IS_FILE : sorting of the alternate key
values completed.
-- File :NVE.USER99.IS_FILE : begin building alternate key
indexes into the file.
-- File :NVE.USER99.IS_FILE : the ALTERNATE_KEY_1 index is
being built.
-- File :NVE.USER99.IS_FILE : AMP$APPLY_KEY_DEFINITIONS
completed building the alternate indexes into the file.
/
```

DELETE_KEY_DEFINITION CREAI Subcommand

Purpose Requests the deletion of an existing alternate key within a CREATE_ALTERNATE_INDEXES utility session.

Format

DELETE_KEY_DEFINITION or
DELKD
NAME=name
STATUS=status variable

Parameters

NAME or KEY_NAME or N or KN

Name of the alternate key to be deleted. This parameter is required.

Remarks

- The DELETE_KEY_DEFINITION subcommand requests deletion of an alternate key but does not actually delete the key from the file. The deletion remains pending until it is applied by either an APPLY_KEY_DEFINITIONS or QUIT subcommand or, it is canceled by a CANCEL_KEY_DEFINITIONS subcommand.

- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This CREATE_ALTERNATE_INDEXES session deletes an alternate key named ALTERNATE_KEY_1.

```
/create_alternate_indexes input=$user.is_file
creai/delete_key_definition key_name=alternate_key_1
creai/quit apply_key_definitions=yes
-- File :NVE.USER99.IS_FILE : begin deleting alternate key
definitions.
-- File :NVE.USER99.IS_FILE : Alternate key ALTERNATE_KEY_1
has been deleted.
-- File :NVE.USER99.IS_FILE : end deleting alternate key
definitions.
/
```

DISPLAY_KEY_DEFINITIONS CREAI Subcommand

Purpose Displays alternate-key definitions within a CREATE_ALTERNATE_INDEXES utility session.

Format

DISPLAY_KEY_DEFINITIONS or
DISPLAY_KEY_DEFINITION or
DISKD
NAMES=keyword or *list of name*
DISPLAY_OPTION=keyword
SAMPLE_RECORD_COUNT=integer or *keyword*
OUTPUT=file
STATUS=status variable

Parameters *NAMES* or *KEY_NAME* or *KEY_NAMES* or *N* or *NAME* or *KN*

Indicates the alternate key definitions displayed.

list of names

Displays the specified alternate-key definitions.

PENDING

Displays only the pending alternate-key creations and deletions.

ALL

Displays both pending and existing alternate-key definitions.

If the KEY_NAMES parameter is omitted, only the pending alternate-key creations and deletions are displayed.

DISPLAY_OPTION or *DO*

Indicates the contents of the display.

BRIEF (B)

Displays the key name, position, length, type, and state.

FULL (F)

Displays all information in the alternate-key definition.

SAMPLE_RECORDS (SR)

Displays only sample records with the alternate keys marked.

BRIEF_SAMPLE_RECORDS (BSR)

Displays the brief definition and the sample records.

FULL_SAMPLE_RECORDS (FSR)

Displays the full definition and the sample records.

ALL (A)

If the DISPLAY_OPTIONS parameter is omitted, the full definition and the sample records are displayed.

SAMPLE_RECORD_COUNT or *SRC*

Indicates the number of records displayed if the DISPLAY_OPTIONS parameter requests a sample record display.

integer

Displays the specified number of records. Values can be 0 through 4398046511103.

ALL

Displays all records in the file.

The default is a one-record display.

OUTPUT or *O*

File to which the display is written. If the OUTPUT parameter is omitted, the display is written to file $OUTPUT.

Remarks

- A sample-record display shows the record contents in ASCII characters and hexadecimal digits with the alternate-key fields underscored. Each alternate key is shown separately by underscores as follows:

  If the concatenated-key or repeating-groups attributes are not defined for the key, the underscore characters indicate the alternate-key type (C for collated, I for integer, or U for uncollated).

  If the key is a concatenated key, the underscores for each key field include one or two letters indicating the order the fields are concatenated (a_, b_, and so forth up to z_ and then, aa, ba, ca, and so forth).

  If the alternate-key definition specifies repeating groups, the underscores for each alternate-key value in the record include a number (1, 2, and so forth).

- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples

This CREATE_ALTERNATE_INDEXES session writes a display to file LIST. The listing includes all records in the file, marked with the proposed alternate-key ALTERNATE_KEY_2.

```
/create_alternate_indexes  input=$user.is_file
creai/crekd  key_name=alternate_key_2 ..
creai../key_position=0  key_length=2 ..
creai../repeating_group_length=20
creai/display_key_definitions  ..
creai../display_option=sample_records ..
creai../sample_record_count=all  output=list
creai/quit  apply_key_definitions=no
/
```

The following CREATE_ALTERNATE_INDEXES session contains a DISPLAY_KEY_DEFINITIONS subcommand for a default display, that is, a full definition of all pending alternate-key creations and deletions and a single sample record.

```
/create_alternate_indexes input=$user.is_file
creai/create_key_definition key_name=alternate_key_1 ..
creai../key_position=0 key_length=2 ..
creai/display_key_definitions
                Display_Key_Definitions     NOS/VE Keyed File Utilities 1.1
File = .NVE.USER99.IS_FILE

 KEY NAME                        POSITION LENGTH TYPE       STATE
 ------------------------------- -------- ------ ---------- ----------------
 ALTERNATE_KEY_1                        0      2 uncollated Creation pending
   Duplicate_Key_Values        : not_allowed
   Null_Suppression            : no
   Repeating_Groups_Specified
     Repeating_Group_Length    : 4
       Repeating_Group_Count   : repeat_to_end_of_record
================================================================================
RECORD 1 .....(in ascii) : This is the first record
                ( in hex ) : 546869732069732074686520666972737420726563 6F72
ALTERNATE_KEY_1         .. : 1_1_   2_2_   3_3_   4_4_   5_5_   6_6_
                (in ascii) : d .
                ( in hex ) : 642E
                           >
creai/quit apply_key_definitions=no
/
```

HELP CREAI Subcommand

Purpose Displays information about utility subcommands.

Format

HELP or
HEL
SUBJECT=string
MANUAL=keyword or *file*
STATUS=status variable

Parameters

SUBJECT or *S*

Index topic to be located in the online manual.

If SUBJECT is omitted, the HELP subcommand lists the names of the utility subcommands.

MANUAL or *M*

Online manual file whose index is searched.

AFM

The AFM online manual index is searched.

FILE

File name of the online manual whose index is searched.

If MANUAL is omitted, the default is AFM. The working catalog is searched for the file and then the $SYSTEM.MANUALS is searched.

Remarks

- If you enter a topic that is not in the manual index, a message appears telling you that the topic could not be found.
- The default manual, $SYSTEM.MANUALS.AFM, contains the online version of the NOS/VE Advanced File Management Usage manual, as provided with the NOS/VE system.
- If your terminal is defined for screen applications, the online manual is displayed in screen mode.

  To leave the online manual, use QUIT. To get help on reading the online manual, use HELP.
- For more information, see the NOS/VE Advanced File Management Usage manual.

HELP CREKD Subcommand

Purpose Displays information about utility subcommands.

Format

HELP or
HEL
SUBJECT=string
MANUAL=file
STATUS=status variable

Parameters

SUBJECT or *S*

Index topic to be located in the online manual.

If SUBJECT is omitted, the HELP subcommand lists the names of the utility subcommands.

MANUAL or *M*

Online manual whose index is searched.

AFM

The AFM online manual index is searched.

File

File name of the online manual whose index is searched.

If MANUAL is omitted, the default is AFM. The working catalog is searched for the file and then the $SYSTEM.MANUALS is searched.

Remarks

- If you enter a topic that is not in the manual index, a message appears telling you that the topic could not be found.
- The default manual, $SYSTEM.MANUALS.AFM, contains the online version of the NOS/VE Advanced File Management Usage manual, as provided with the NOS/VE system.
- If your terminal is defined for screen applications, the online manual is displayed in screen mode.

  To leave the online manual, use QUIT. To get help on reading the online manual, use HELP.
- For more information, see the NOS/VE Advanced File Management Usage manual.

QUIT
CREAI Subcommand

Purpose Ends the CREATE_ALTERNATE_INDEXES utility session.

Format

QUIT or
QUI
APPLY_KEY_DEFINITIONS=boolean or *keyword*
ERROR_LIMIT=integer
STATUS=status variable

Parameters

APPLY_KEY_DEFINITIONS or *APPLY_KEY_DEFINITION* or *AKD*

Indicates how pending alternate-key creation and deletion requests are processed.

APPLY (A), TRUE (ON or YES)

Apply all pending creation and deletion requests.

CANCEL (C), FALSE (OFF or NO)

Cancel all pending creation and deletion requests.

This parameter is required if creation or deletion requests are pending.

ERROR_LIMIT or *EL*

Number of trivial (nonfatal) errors allowed for the apply operation (integer from 0 through 65535).

0 is the default value and indicates no limit.

See the APPLY_KEY_DEFINITIONS command description for a description of apply error processing.

Remarks

- The APPLY_KEY_DEFINITIONS parameter is required only if alternate-key creation or deletion requests are pending. In this case, you must specify whether to apply or cancel the pending requests.

  If you request application of the pending creations and deletions, the QUIT subcommand performs the same processing as the APPLY_KEY_DEFINITIONS subcommand before exiting the utility.

If you request cancellation of the requests, the QUIT subcommand performs the same processing as the CANCEL_KEY_DEFINITIONS subcommand before exiting the utility.

- For more information, see the APPLY_KEY_DEFINITIONS and CANCEL_KEY_DEFINITIONS subcommand descriptions.
- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This CREATE_ALTERNATE_INDEXES session requests an alternate-key deletion and an alternate-key creation, but then cancels the requests.

```
/create_alternate_indexes file=$user.isfile
creai/delete_key_definition alternate_key_1
creai/create_key_definition alternate_key_1 ..
creai../key_position=0 key_length=5 key_type=integer
creai/quit apply_key_definitions=no
/
```

QUIT CREKD Subcommand

Purpose Exits the CREATE_KEY_DEFINITION utility, ending concatenated-key specification.

Format

QUIT or
QUI
STATUS=status variable

Remarks

- Entry of the QUIT subcommand returns you to the CREATE_ALTERNATE_INDEXES utility session. This is indicated by the prompt creai/.
- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples This CREATE_ALTERNATE_INDEXES session defines a concatenated alternate key having two pieces. The first piece is the ten bytes beginning at byte 5. (Remember, bytes are numbered from the left beginning with zero.) The second piece is the five-byte integer at the beginning of the record.

```
/create_alternate_indexes input=$user.is_file
creai/create_key_definition alternate_key_3 ..
creai../key_position=5 key_length=10 ..
creai../concatenated_pieces=yes
crekd/add_piece key_position=0 key_length=5 ..
crekd../key_type=integer
crekd/quit          "Exits CREATE_KEY_DEFINITIONS.
creai/quit no       "Exits CREATE_ALTERNATE_INDEXES without
/                   "applying the alternate-key definition.
```

SEPARATE_KEY_GROUPS CREAI Subcommand

Remarks Reserved for site personnel, Control Data, or future use.

CREATE_INTERSTATE_CONNECTION 12

CREATE_INTERSTATE_CONNECTION Command

Purpose Establishes a NOS batch control point on a dual state system.

Format

CREATE_INTERSTATE_CONNECTION or
CREIC
PARTNER_JOB_CARD=string
STATUS=status variable

Parameters

PARTNER_JOB_CARD or *PJC*

Specifies the job statement parameters to be used for the NOS batch job. The parameter syntax must conform to NOS job statement rules.

Omission causes the NOS default job statement parameters to be used (an infinite time limit and no other parameters specified).

Remarks

- After you enter a CREATE_INTERSTATE_CONNECTION command, prompts are issued until you enter QUIT (QUI) or DELETE_INTERSTATE_CONNECTION (DELIC).
- While the interstate connection is open, you can enter any NOS/VE command (except another CREIC command). You can enter NOS commands to be executed on the NOS side of the dual state system through the EXECUTE_INTERSTATE_COMMAND command. The CREIC command is generally used in conjunction with the File Management Utility to migrate files between NOS and NOS/VE.
- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples The following commands create an interstate connection, execute NOS commands (ATTACH, DEFINE, and COPY), and close the connection. FA is the CREATE_INTERSTATE_CONNECTION prompt for user input.

```
/create_interstate_connection partner_job_card=..
../'myjob,,64.'
FA/execute_interstate_command command='attach,oldfl.'
FA/execute_interstate_command command='define,newfl.'
FA/execute_interstate_command command=..
FA../'copy,oldfl,newfl.'
FA/delete_interstate_connection
/
```

DELETE_INTERSTATE_CONNECTION CREIC Subcommand

Purpose Ends a CREATE_INTERSTATE_CONNECTION session.

Format **DELETE_INTERSTATE_CONNECTION** or
QUI or
QUIT or
DELIC

Parameters None.

Remarks For more information, see the NOS/VE Advanced File Management manual.

EXECUTE_INTERSTATE_COMMAND CREIC Subcommand

Purpose Precedes all NOS commands when the interstate connection established by CREATE_INTERSTATE_CONNECTION (CREIC) is in effect.

Format **EXECUTE_INTERSTATE_COMMAND** or
EXEIC
COMMANDS=list of string
STATUS=status variable

Parameters **COMMANDS** or **COMMAND** or **C**

A NOS command followed by a period. The command string can include up to 80 characters and must be enclosed in apostrophes. This command is required.

Remarks For more information, see the Migration From NOS to NOS/VE manual.

CREATE_OBJECT_LIBRARY 13

CREATE_OBJECT_LIBRARY Command

Purpose Begins a CREATE_OBJECT_LIBRARY utility session. The utility produces an object library or an object file and allows post-compilation manipulation of object or load modules. It can also produce a text version of certain kinds of modules on an object library.

Format

CREATE_OBJECT_LIBRARY or
CREOL or
OCU
STATUS=status variable

Remarks

- The following files can be created by the GENERATE_LIBRARY subcommand of this utility. The utility issues a warning and does not process input files whose attributes do not conform to the attributes listed in the right-hand column of the table below. The utility sets the accompanying file attributes listed for output files it creates. You can override attributes with the SCL SET_FILE_ATTRIBUTE command.

File Created	Attributes Given to the File
Form source	FILE_CONTENT=LEGIBLE_SCL_INCLUDE
Form variable	FILE_CONTENT=LEGIBLE
Object library	FILE_CONTENT=OBJECT_LIBRARY
Object file	FILE_CONTENT=OBJECT_DATA
SCL procedure file	FILE_CONTENT=LEGIBLE_SCL_PROCEDURE
Message module file	FILE_CONTENT=LEGIBLE_SCL_INCLUDE

- The CREOL session ends when you enter the QUIT subcommand.
- For more information, see the NOS/VE Object Code Management manual.

Examples Following is a sequence that removes an object library from the command list, creates a new version of the object library from the modules on file $LOCAL.LGO, and then adds the object library to the command list.

```
/delete_command_list_entry entry=$local.my_commands
/create_object_library
COL/add_module $local.lgo
COL/generate_library $local.my_commands
COL/quit
/create_command_list_entry entry=$local.my_commands
/
```

ADD_BOX
CREFM Subcommand

Purpose ADD_BOX adds a graphic box object to a form.

Format

ADD_BOX or
ADDB
COLUMN=integer
LINE=integer
WIDTH=integer
HEIGHT=integer
DISPLAY=list of keyword
NAME=name or *cobol_name*
OCCURRENCE=integer
STATUS=status variable

Parameters

COLUMN or **C**

The column position for the upper left corner of the graphic box object. Column 1 is the upper left corner of the form. The valid values are from 1 through 256. This parameter is required.

LINE or L

The line position for the upper left corner of the graphic box object. Line 1 is the upper left corner of the form. The valid values are from 1 through 256. This parameter is required.

WIDTH or W

The number of columns the graphic box object occupies. The valid values are from 1 through 256. This parameter is required.

HEIGHT or H

The number of lines the graphic box object occupies. The valid values are from 1 through 256. This parameter is required.

DISPLAY* or *DISPLAYS* or *D

A list of display attributes for the graphic box object. The following values are valid:

INVERSE
LOW_INTENSITY
HIGH_INTENSITY
BLINK
BLACK_BACKGROUND
BLUE_BACKGROUND
GREEN_BACKGROUND
MAGENTA_BACKGROUND
RED_BACKGROUND
CYAN_BACKGROUND
YELLOW_BACKGROUND
WHITE_BACKGROUND
BLACK_FOREGROUND
BLUE_FOREGROUND
GREEN_FOREGROUND
MAGENTA_FOREGROUND
RED_FOREGROUND
CYAN_FOREGROUND
YELLOW_FOREGROUND
WHITE_FOREGROUND
FINE_LINE
MEDIUM_LINE
BOLD_LINE

The defaults are the foreground and background colors of the form.

13

NAME or *N*

The name of the graphic box object. The default is spaces.

OCCURRENCE or *O*

The occurrence of the name. The valid values are from 1 through 1000. The default is 1.

Remarks For more information, see the NOS/VE Screen Formatting manual.

ADD_CONSTANT_TEXT CREFM Subcommand

Purpose ADD_CONSTANT_TEXT adds a constant text object to the form. A constant text object occupies a single line. In contrast, a constant text box object occupies more than one line (see ADD_CONSTANT_TEXT_BOX).

Format

ADD_CONSTANT_TEXT or
ADDCT
COLUMN=integer
LINE=integer
TEXT=string
DISPLAY=list of keyword
NAME=name or *cobol_name*
OCCURRENCE=integer
WIDTH=integer
STATUS=status variable

Parameters

COLUMN or **C**

The position for the first column of the constant text object. Column 1 is the upper left corner of the form. The valid values are from 1 through 256. This parameter is required.

LINE or **L**

The line position for the constant text object. Line 1 is the upper left corner of the form. The valid values are from 1 through 256. This parameter is required.

TEXT or T

The constant text. Neither the application program nor the application user can change this text. You can enter a string of from 1 to 65,535 characters. This parameter is required.

DISPLAY or *DISPLAYS* or *D*

A list of display attributes for the constant text object. The program can change these attributes. The following values are valid:

INVERSE
LOW_INTENSITY
HIGH_INTENSITY
BLINK
HIDDEN
UNDERLINE
BLACK_BACKGROUND
BLUE_BACKGROUND
GREEN_BACKGROUND
MAGENTA_BACKGROUND
RED_BACKGROUND
CYAN_BACKGROUND
YELLOW_BACKGROUND
WHITE_BACKGROUND
BLACK_FOREGROUND
BLUE_FOREGROUND
GREEN_FOREGROUND
MAGENTA_FOREGROUND
RED_FOREGROUND
CYAN_FOREGROUND
YELLOW_FOREGROUND
WHITE_FOREGROUND
ITALIC
TITLE
INPUT
ERROR
MESSAGE
DISPLAY_LEFT_TO_RIGHT
DISPLAY_RIGHT_TO_LEFT

The defaults are the foreground and background colors of the form and DISPLAY_LEFT_TO_RIGHT.

NAME or *N*

The name of the constant text object. The default is spaces.

OCCURRENCE or *O*

The occurrence of the name. The valid values are from 1 through 1000. The default is 1.

WIDTH or *W*

The number of columns the constant text object occupies. The valid values are from 1 through 256. The default is the number of characters in the text.

Use this parameter to specify a display attribute that occupies more space than the text.

Remarks For more information, see the NOS/VE Screen Formatting manual.

ADD_CONSTANT_TEXT_BOX CREFM Subcommand

Purpose ADD_CONSTANT_TEXT_BOX adds a constant text box object to a form. A constant text box object occupies more than one line. In contrast, a constant text object occupies only one line (see ADD_CONSTANT_TEXT).

Format

ADD_CONSTANT_TEXT_BOX or
ADDCTB
COLUMN=integer
LINE=integer
TEXT=string
WIDTH=integer
HEIGHT=integer
DISPLAY=list of keyword
NAME=name or *cobol_name*
OCCURRENCE=integer
TEXT_FORMAT=keyword
STATUS=status variable

Parameters

COLUMN or **C**

The column position of the upper left corner of the constant text box object. Column 1 is the upper left corner of the form. The valid values are from 1 through 256. This parameter is required.

LINE or L

The line position for the upper left corner of the constant text box object. Line 1 is the upper left corner of the form. The valid values are from 1 through 256. This parameter is required.

TEXT or T

The constant text (neither the application program nor the application user can change this text). You can enter a string of from 1 through 65,535 characters. This parameter is required.

WIDTH or W

The number of columns the constant text box object occupies. The valid values are from 1 through 256. This parameter is required.

HEIGHT or H

The number of lines the constant text box object occupies. The valid values are from 1 through 256. This parameter is required.

DISPLAY* or *DISPLAYS* or *D

A list of display attributes for the constant text box object. (The program can change these attributes.) The following values are valid:

INVERSE
LOW_INTENSITY
HIGH_INTENSITY
BLINK
HIDDEN
UNDERLINE
BLACK_BACKGROUND
BLUE_BACKGROUND
GREEN_BACKGROUND
MAGENTA_BACKGROUND
RED_BACKGROUND
CYAN_BACKGROUND
YELLOW_BACKGROUND
WHITE_BACKGROUND
BLACK_FOREGROUND
BLUE_FOREGROUND
GREEN_FOREGROUND
MAGENTA_FOREGROUND

RED_FOREGROUND
CYAN_FOREGROUND
YELLOW_FOREGROUND
WHITE_FOREGROUND
ITALIC
TITLE
INPUT
ERROR
MESSAGE
DISPLAY_LEFT_TO_RIGHT
DISPLAY_RIGHT_TO_LEFT

The defaults are the foreground and background colors of the form and DISPLAY_LEFT_TO_RIGHT.

NAME or *N*

The name of the constant text box object. The default is spaces.

OCCURRENCE or *O*

The occurrence of the name. The valid values are from 1 through 1000. The default is 1.

TEXT_FORMAT or *TF*

The format for breaking text between lines. The following values are valid:

Value	Meaning
WRAP_WORDS	Breaks text between words.
WRAP_CHARACTERS	Breaks text at any character.

The default is WRAP_WORDS.

Remarks For more information, see the NOS/VE Screen Formatting manual.

ADD_DISPLAY CREFM Subcommand

Purpose ADD_DISPLAY specifies a program name for a terminal display attribute. When the program interacts with the form to change a display attribute of an object, it uses the name specified by this subcommand.

Format

ADD_DISPLAY or
ADDD
NAME=name or **cobol_name**
DISPLAY=list of keyword
STATUS=status variable

Parameters

NAME or **N**

The name the program uses to change display attributes for objects on the form. This parameter is required.

DISPLAY or **DISPLAYS** or **D**

A list of display attributes. These attributes correspond to the attributes specified in the terminal definition input statement. (For more information, see the NOS/VE Terminal Definitions manual.) This parameter is required.

The following values are valid:

INVERSE
LOW_INTENSITY
HIGH_INTENSITY
BLINK
UNDERLINE
PROTECT
HIDDEN
BLACK_BACKGROUND
BLUE_BACKGROUND
GREEN_BACKGROUND
MAGENTA_BACKGROUND
RED_BACKGROUND
CYAN_BACKGROUND
YELLOW_BACKGROUND
WHITE_BACKGROUND
BLACK_FOREGROUND
BLUE_FOREGROUND
GREEN_FOREGROUND
MAGENTA_FOREGROUND
RED_FOREGROUND
CYAN_FOREGROUND
YELLOW_FOREGROUND
WHITE_FOREGROUND
FINE_LINE
MEDIUM_LINE
BOLD_LINE
ITALIC
TITLE

INPUT
ERROR
MESSAGE
DISPLAY_LEFT_TO_RIGHT
DISPLAY_RIGHT_TO_LEFT

Remarks For more information, see the NOS/VE Screen Formatting manual.

ADD_EVENT CREFM Subcommand

Purpose ADD_EVENT adds an event definition to the form.

Format

ADD_EVENT or
ADDE
PROGRAM_EVENT=name or **cobol_name**
TERMINAL_EVENT=keyword
ACTION=keyword
LABEL=string
STATUS=status variable

Parameters

PROGRAM_EVENT or PE

The name the program uses for an event. This parameter is required.

TERMINAL_EVENT or TE

The key or keys that execute the event. The values correspond to the function keys specified in the terminal definition input statements (for information, see the NOS/VE Terminal Definitions manual). For information on how Screen Formatting uses the value you specify to assign a key, see the NOS/VE Screen Formatting manual. This parameter is required.

The following values are valid:

NEXT	SHIFT_NEXT
HELP	SHIFT_HELP
STOP	SHIFT_STOP
BACK	SHIFT_BACK
UP	SHIFT_UP
DOWN	SHIFT_DOWN
FORWARD	SHIFT_FORWARD
BACKWARD	SHIFT_BACKWARD
UNDO	

REDO	
QUIT	
EXIT	
FIRST	
LAST	
EDIT	SHIFT_EDIT
DATA	SHIFT_DATA
F1	SHIFT_F1
F2	SHIFT_F2
F3	SHIFT_F3
F4	SHIFT_F4
F5	SHIFT_F5
F6	SHIFT_F6
F7	SHIFT_F7
F8	SHIFT_F8
F9	SHIFT_F9
F10	SHIFT_F10
F11	SHIFT_F11
F12	SHIFT_F12
F13	SHIFT_F13
F14	SHIFT_F14
F15	SHIFT_F15
F16	SHIFT_F16
PICK	
INSERT_LINE	
DELETE_LINE	
HOME	

The following terminal events are equivalent:

QUIT	STOP
EXIT	SHIFT_STOP
FIRST	SHIFT_BACKWARD
LAST	SHIFT_FORWARD

When Screen Formatting returns these events, it uses the second form in the list.

ACTION or A

Screen Formatting's action when the terminal event occurs. For descriptions of the values, see the NOS/VE Screen Formatting manual. This parameter is required.

The following values are valid:

RETURN_NORMAL
RETURN ABNORMAL

PAGE_TABLE_FORWARD
PAGE_TABLE_BACKWARD
SCROLL_TABLE_FORWARD
SCROLL_TABLE_BACKWARD
DISPLAY_HELP
ERASE_HELP
IGNORE
TAB_NEXT
TAB_PREVIOUS
SCROLL_VARIABLE_FORWARD
SCROLL_VARIABLE_BACKWARD
PAGE_VARIABLE_FORWARD
PAGE_VARIABLE_BACKWARD
PAGE_VARIABLE_FIRST
PAGE_VARIABLE_LAST
PAGE_TABLE_FIRST
PAGE_TABLE_LAST

LABEL or *L*

The label for the event (this label appears on the screen). The label is a 0- to 6-character string. The default is no label.

Remarks For more information, see the NOS/VE Screen Formatting manual.

ADD_LINE
CREFM Subcommand

Purpose ADD_LINE adds a graphic line object to form.

Format

ADD_LINE or
ADDL
START_COLUMN=integer
START_LINE=integer
END_COLUMN=integer
END_LINE=integer
DISPLAY=list of keyword
NAME=name or *cobol_name*
OCCURRENCE=integer
STATUS=status variable

Parameters

START_COLUMN or SC

The column position for the start of the graphic line object. Column 1 is the upper left corner of the form. The valid values are from 1 through 256. This parameter is required.

START_LINE or SL

The line position for the start of the graphic line object. Line 1 is the upper left corner of the form. The valid values are from 1 through 256. This parameter is required.

END_COLUMN or EC

The column position for the end of the graphic line object. The valid values are from 1 through 256. This parameter is required.

END_LINE or EL

The line position for the end of the graphic line object. The valid values are from 1 through 256. This parameter is required.

DISPLAY or *DISPLAYS* or *D*

A list of display attributes for the graphic line object. The following values are valid:

INVERSE
LOW_INTENSITY
HIGH_INTENSITY
BLINK
BLACK_BACKGROUND
BLUE_BACKGROUND
GREEN_BACKGROUND
MAGENTA_BACKGROUND
RED_BACKGROUND
CYAN_BACKGROUND
YELLOW_BACKGROUND
WHITE_BACKGROUND
BLACK_FOREGROUND
BLUE_FOREGROUND
GREEN_FOREGROUND
MAGENTA_FOREGROUND
RED_FOREGROUND
CYAN_FOREGROUND
YELLOW_FOREGROUND

WHITE_FOREGROUND
FINE_LINE
MEDIUM_LINE
BOLD_LINE

The defaults are the foreground and background colors of the form.

NAME or *N*

The name of the graphic line object. The default is spaces.

OCCURRENCE or *O*

The occurrence of the name. The valid values are from 1 through 1000. The default is 1.

Remarks For more information, see the NOS/VE Screen Formatting manual.

ADD_MODULE CREOL Subcommand

Purpose Adds one or more modules to the module list.

Format

ADD_MODULE or
ADD_MODULES or
ADDM
LIBRARY=list of file
MODULE=list of program_name or *list of range of program_name*
PLACEMENT=keyword
DESTINATION=program_name
STATUS=status variable

Parameters **LIBRARY** or **LIBRARIES** or **L**

Object files, SCL procedure files, or object library files containing the modules to be added. This parameter is required.

MODULE or *MODULES* or *M*

Modules to be added.

You use a string value for a module whose name is not an SCL name or a COBOL name. An example of such a module name is a C function, where lowercase is significant.

If MODULE is omitted, all modules on the files specified on the LIBRARY parameter are added.

PLACEMENT or *P*

Indicates whether the added modules are placed before or after the module specified on the DESTINATION parameter. Options are:

BEFORE (B)

Modules added before the destination module.

AFTER (A)

Modules added after the destination module.

If PLACEMENT is omitted, AFTER is used.

DESTINATION or *D*

Module before or after which the added modules are placed.

If DESTINATION is omitted, the location depends on the PLACEMENT parameter value. If PLACEMENT=BEFORE, the modules are placed at the beginning of the module list; if PLACEMENT=AFTER, the modules are placed at the end of the module list.

Remarks

- The ADD_MODULE subcommand can specify object files, SCL procedure files, or object libraries. The CREOL utility adds modules from files in the order you specify the files on the LIBRARY parameter. If you do not want to use all modules in the files, specify which modules to be added on the MODULE parameter.

- The ADD_MODULE subcommand adds each module to the end of the module list unless otherwise specified by the PLACEMENT and DESTINATION PARAMETERS. If the module list already contains a module of the same name, a warning status message is returned and the module is not added.

- The ADD_MODULE subcommand does not replace modules in the module list. To replace modules, enter a REPLACE_MODULES subcommand. To add and replace modules, enter a COMBINE_MODULES subcommand.

- If you specify an SCL procedure whose header references a non-standard type, you must make the type definition available. For instance, if you want to add the following procedure:

```
PROCEDURE show(
   p1: address_list = $required
   status)
  :
PROCEND show
```

  then the type definition for ADDRESS_LIST must be created outside the procedure. This is accomplished by using the TYPE control statement, as in:

```
TYPE
   address_list : list 1..3 of string
TYPEND
```

- For more information, see the NOS/VE Object Code Management manual.

Examples The following subcommand adds all modules on files BINARY1 and BINARY2 to the beginning of the module list.

```
COL/add_module (binary1,binary2) placement=before
```

ADD_STORED_TEXT CREFM Subcommand

Purpose ADD_STORED_TEXT adds an initial value for a table variable occurrence that does not initially appear on the form.

Format

ADD_STORED_TEXT or
ADDST
VARIABLE_NAME=name or **cobol_name**
OCCURRENCE=integer
TEXT=string
DISPLAY=list of keyword
STATUS=status variable

Parameters

VARIABLE_NAME or VN

The name of the stored object. This parameter is required.

OCCURRENCE or O

The occurrence of the name. The value 1 is the first or only occurrence. The valid values are from 1 through 1000. This parameter is required.

TEXT or *T*

The initial text for the stored object. You can enter a string of from 1 through 65,535 characters. The default is spaces.

DISPLAY or *DISPLAYS* or *D*

A list of display attributes for the variable text object. The following values are valid:

INVERSE
LOW_INTENSITY
HIGH_INTENSITY
BLINK
UNDERLINE
PROTECT
HIDDEN
BLACK_BACKGROUND
BLUE_BACKGROUND
GREEN_BACKGROUND
MAGENTA_BACKGROUND
RED_BACKGROUND
CYAN_BACKGROUND
YELLOW_BACKGROUND
WHITE_BACKGROUND
BLACK_FOREGROUND
BLUE_FOREGROUND
GREEN_FOREGROUND
MAGENTA_FOREGROUND
RED_FOREGROUND
CYAN_FOREGROUND
YELLOW_FOREGROUND
WHITE_FOREGROUND
ITALIC
TITLE
INPUT
ERROR
MESSAGE

DISPLAY_LEFT_TO_RIGHT
DISPLAY_RIGHT_TO_LEFT

The defaults are the foreground and background colors of the form and DISPLAY_LEFT_TO_RIGHT.

Remarks For more information, see the NOS/VE Screen Formatting manual.

ADD_TABLE CREFM Subcommand

Purpose ADD_TABLE adds a table (a group of one or more variables). The table may have one or more occurrences.

Format

ADD_TABLE or
ADDT
TABLE_NAME=name or **cobol_name**
VARIABLE_NAME=list of: name or **cobol_name**
STORED_OCCURRENCE=integer
VISIBLE_OCCURRENCE=integer
STATUS=status variable

Parameters

TABLE_NAME or **TN**

The name of the table. This parameter is required.

VARIABLE_NAME or **VARIABLE_NAMES** or **VN**

A list of the names of variables that belong to the table. This parameter is required.

STORED_OCCURRENCE or **STORED_OCCURRENCES** or **SO**

The maximum number of stored occurrences allowed in the table. The value must be greater than or equal to the value for the VISIBLE_OCCURRENCE parameter. The valid values are from 1 through 1000. This parameter is required.

VISIBLE_OCCURRENCE or *VISIBLE_OCCURRENCES* or *VO*

The number of occurrences visible to the user. You must add a variable text object for each visible occurrence. The valid values are from 1 through 1000. The default is the value specified for STORED_OCCURRENCE.

Remarks For more information, see the NOS/VE Screen Formatting manual.

ADD_VARIABLE CREFM Subcommand

Purpose ADD_VARIABLE specifies general attributes for a variable.

Format

ADD_VARIABLE or
ADDV
VARIABLE_NAME=name or **cobol_name**
IO_MODE=keyword
DATA_TYPE=keyword
ERROR_PROCESSING=keyword or *name* or *string*
ERROR_DISPLAY=list of keyword
HELP_PROCESSING=keyword or *name* or *string*
LENGTH=integer
USER_ENTRY=list of keyword
COMMENT=list of string
STATUS=status variable

Parameters

VARIABLE_NAME or **VN**

The name of the variable. This parameter is required.

IO_MODE or *IM*

The input and output performed on the variable. The following values are valid:

Value	Meaning
INPUT	The user inputs data, which is blanked as soon as possible.
INPUT_OUTPUT	The user inputs data, which remains visible. The program outputs data to this variable.

OUTPUT	The program outputs data to the terminal (the user cannot enter data). Any user modification to the variable is corrected as soon as possible.
PROGRAM	Programs save data from one terminal interaction to another. The terminal user does not see the variable.

The default is INPUT_OUTPUT.

DATA_TYPE or *DT*

The data type of the variable. The following values are valid:

Value	Meaning
CHARACTER	Passes user-entered characters to the program without converting them.
COBOL	Converts user-entered characters to a COBOL format.
INTEGER	Converts user-entered characters to an integer.
REAL	Converts user-entered characters to a real number.
UPPERCASE	Converts both user-entered and program-supplied characters to uppercase.

The default is CHARACTER.

ERROR_PROCESSING or *EP*

Error processing for the variable. Error processing occurs when:

- The user enters data that cannot be converted to the specified program data type.
- The data is not one of the valid values for the variable.

- The user did not enter required data.

The following values are valid:

Value	Meaning
name	When an error occurs, Screen Formatting displays a form with the specified name.
string	When an error occurs, Screen Formatting displays the specified string. The string can be from 0 through 256 characters.
NONE	Screen Formatting displays nothing on the screen, but returns the error in the VARIABLE_STATUS parameter of the application program call.
SYSTEM	When an error occurs, Screen Formatting displays a default message.

The default is NONE.

ERROR_DISPLAY or *ERROR_DISPLAYS* or *ED*

A list of attributes for displaying an error when a variable does not pass validation. The following values are valid:

INVERSE
LOW_INTENSITY
HIGH_INTENSITY
BLINK
UNDERLINE
PROTECT
HIDDEN
BLACK_BACKGROUND
BLUE_BACKGROUND
GREEN_BACKGROUND
MAGENTA_BACKGROUND
RED_BACKGROUND
CYAN_BACKGROUND
YELLOW_BACKGROUND
WHITE_BACKGROUND
BLACK_FOREGROUND
BLUE_FOREGROUND
GREEN_FOREGROUND
MAGENTA_FOREGROUND
RED_FOREGROUND

CYAN_FOREGROUND
YELLOW_FOREGROUND
WHITE_FOREGROUND
ITALIC
TITLE
INPUT
ERROR
MESSAGE

The defaults are INVERSE and UNDERLINE.

HELP_PROCESSING or *HP*

Help processing for the variable. Help processing occurs when the user executes a help event. The following values are valid:

Value	Meaning
name	Displays a form with the specified name.
string	Displays the specified string. The string can be from 0 through 256 characters.
NONE	Displays no help information.
SYSTEM	Displays a default message.

The default is NONE.

LENGTH or *L*

The length of the variable in characters. This attribute applies only to variables with the data type CHARACTER or UPPERCASE. The user can execute scrolling commands to see all the data in the program variable. The valid values are from 1 through 65535. The default is the size of the text object for the variable.

USER_ENTRY or *USER_ENTRIES* or *UE*

The user entry actions for the variable. The following values are valid:

Value	Meaning
OPTIONAL	The user does not need to enter data for the variable.
MUST_ENTER	The user must enter data for the variable. The initial value of the variable need not be in the list of valid values. For example, you can use this value to require the user to enter a password, although you do not want to let the user see a valid password. The list of valid values lists valid passwords. You specify spaces as the initial value.

The default is OPTIONAL.

COMMENT or *COMMENTS* or *C*

A list of strings to be saved as comments with the variable definition. By default, no comments are saved.

Remarks

- When you use this subcommand, you must also use either the ADD_VARIABLE_TEXT or ADD_VARIABLE_TEXT_BOX subcommand.
- When you specify COBOL for the DATA_TYPE parameter, you can also use the following subcommands to specify the program format and the display format for a COBOL variable:

  SET_COBOL_DATA
  SET_COBOL_OUTPUT

  If you do not use these subcommands, Screen Formatting uses the following string for each subcommand:

  PIC X(n)

  (n represents the size of the variable text object.)

- If you want Screen Formatting to validate a COBOL variable, use the following subcommands:

  SET_INTEGER_INPUT
  SET_REAL_INPUT
  SET_CHARACTER_INPUT

  For more information on validating COBOL variables, see the NOS/VE Screen Formatting manual.

- When you specify CHARACTER, INTEGER, REAL, or UPPERCASE for the DATA_TYPE parameter, you can also use the following subcommands to specify the user entry format, the valid values, and the display format for the variable:

  SET_CHARACTER_INPUT
  SET_EXPONENT_OUTPUT
  SET_FLOAT_OUTPUT
  SET_INTEGER_INPUT
  SET_INTEGER_OUTPUT
  SET_MONEY_INPUT
  SET_MONEY_OUTPUT
  SET_REAL_INPUT

  If you do not use the preceding subcommands, Screen Formatting uses the defaults appropriate to the data type of the variable.

- For more information, see the NOS/VE Screen Formatting manual.

ADD_VARIABLE_TEXT CREFM Subcommand

Purpose ADD_VARIABLE_TEXT adds a variable text object to the form. A variable text object occupies a single line. In contrast, a variable text box object occupies more than one line (see ADD_VARIABLE_TEXT_BOX).

Format

ADD_VARIABLE_TEXT or
ADDVT
COLUMN=integer
LINE=integer
TEXT=string

VARIABLE_NAME=name or **cobol_name**
OCCURRENCE=integer
DISPLAY=list of keyword
WIDTH=integer
STATUS=status variable

Parameters

COLUMN or **C**

The position for the first column of the variable text object. Column 1 is the upper left corner of the form. The valid values are from 1 through 256. This parameter is required.

LINE or **L**

The line position for the variable text object. Line 1 is the upper left corner of the form. The valid values are from 1 through 256. This parameter is required.

TEXT or **T**

The initial text for the variable text object. You can enter a string of from 0 through 65,535 characters. If you want the initial display to be blank space, specify a null string and the WIDTH parameter. This parameter is required.

VARIABLE_NAME or **VN**

The name of the variable text object. This parameter is required.

OCCURRENCE or *O*

The occurrence of the name. The valid values are from 1 through 1000. The default is 1.

DISPLAY or *DISPLAYS* or *D*

A list of display attributes for the variable text object. The following values are valid:

INVERSE
LOW_INTENSITY
HIGH_INTENSITY
BLINK
UNDERLINE
PROTECT
HIDDEN
BLACK_BACKGROUND
BLUE_BACKGROUND
GREEN_BACKGROUND

MAGENTA_BACKGROUND
RED_BACKGROUND
CYAN_BACKGROUND
YELLOW_BACKGROUND
WHITE_BACKGROUND
BLACK_FOREGROUND
BLUE_FOREGROUND
GREEN_FOREGROUND
MAGENTA_FOREGROUND
RED_FOREGROUND
CYAN_FOREGROUND
YELLOW_FOREGROUND
WHITE_FOREGROUND
ITALIC
TITLE
INPUT
ERROR
MESSAGE
DISPLAY_LEFT_TO_RIGHT
DISPLAY_RIGHT_TO_LEFT

The defaults are the foreground and background colors of the form and DISPLAY_LEFT_TO_RIGHT.

WIDTH or *W*

The number of columns the variable text object occupies. The valid values are from 1 through 256. The default is the number of characters in the text.

Use this parameter when you want a display attribute to occupy more space than the text.

Remarks

- When you use this subcommand, you must also use the ADD_VARIABLE subcommand.
- For more information, see the NOS/VE Screen Formatting manual.

ADD_VARIABLE_TEXT_BOX CREFM Subcommand

Purpose ADD_VARIABLE_TEXT_BOX adds a variable text box object to a form. A variable text box object occupies more than one line. In contrast, variable text occupies only one line (see ADD_VARIABLE_TEXT).

Format

ADD_VARIABLE_TEXT_BOX or
ADDVTB
COLUMN=integer
LINE=integer
TEXT=string
WIDTH=integer
HEIGHT=integer
VARIABLE_NAME=name or **cobol_name**
OCCURRENCE=integer
DISPLAY=list of keyword
TEXT_FORMAT=keyword
STATUS=status variable

Parameters

COLUMN or **C**

The column position of the upper left corner of the variable text box object. Column 1 is the upper left corner of the form. The valid values are from 1 through 256. This parameter is required.

LINE or **L**

The line position of the upper left corner of the variable text box object. Line 1 is the upper left corner of the form. The valid values are from 1 through 256. This parameter is required.

TEXT or **T**

The initial text for the variable text box object. You can enter a string of from 0 through 65,535 characters. If you want the initial display to be blank space, specify a null string. This parameter is required.

WIDTH or **W**

The number of columns the variable text box object occupies. The valid values are from 1 through 256. This parameter is required.

HEIGHT or **H**

The number of lines the variable text box object occupies. The valid values are from 1 through 256. The parameter is required.

VARIABLE_NAME or **VN**

The name of the variable text box object. This parameter is required.

OCCURRENCE or *O*

The occurrence of the name. The valid values are from 1 through 1000. The default is 1.

DISPLAY or *DISPLAYS* or *D*

A list of display attributes for the variable text box object. The following values are valid:

INVERSE
LOW_INTENSITY
HIGH_INTENSITY
BLINK
UNDERLINE
PROTECT
HIDDEN
BLACK_BACKGROUND
BLUE_BACKGROUND
GREEN_BACKGROUND
MAGENTA_BACKGROUND
RED_BACKGROUND
CYAN_BACKGROUND
YELLOW_BACKGROUND
WHITE_BACKGROUND
BLACK_FOREGROUND
BLUE_FOREGROUND
GREEN_FOREGROUND
MAGENTA_FOREGROUND
RED_FOREGROUND
CYAN_FOREGROUND
YELLOW_FOREGROUND
WHITE_FOREGROUND
ITALIC
TITLE
INPUT
ERROR
MESSAGE
DISPLAY_LEFT_TO_RIGHT
DISPLAY_RIGHT_TO_LEFT

The defaults are the foreground and background colors of the form.

TEXT_FORMAT or *TF*

The format for breaking text between lines. The following values are valid:

Value	Meaning
WRAP_WORDS	Breaks text between words.
WRAP_CHARACTERS	Breaks text at any character.

The default is WRAP_WORDS.

Remarks

- When you use this subcommand, you must also use the ADD_VARIABLE subcommand.
- For more information, see the NOS/VE Screen Formatting manual.

BIND_MODULE CREOL Subcommand

Purpose

Subcommand used in a restructuring procedure to bind component modules into a single load module. This subcommand is not recommended for your use. The subcommand description is provided only to help you interpret the commands in a restructuring procedure. To create a new module by binding component modules, you should use the subcommand CREATE_MODULE.

Format

BIND_MODULE or
BINM
MODE=keyword
NAME=program_name
FILE=file
STARTING_PROCEDURE=program_name
SECTION_ORDER=list of record
PRESET_VALUE=keyword
INCLUDE_BINARY_SECTION_MAPS=boolean
OUTPUT=file
STATUS=status variable

Parameters

MODE or **M**

Indicates whether additional BIND_MODULE subcommands for the module follow this subcommand. Options are:

CONTINUE

More BIND_MODULE subcommands follow.

QUIT

This is the last BIND_MODULE subcommand for the module.

This parameter is required.

NAME or *N*

Name of the new module. This parameter is required only on the first BIND_MODULE subcommand for the module.

Use a string value for a module name which is not an SCL name or a COBOL name.

FILE or *F*

File containing the modules to be bound. This parameter is required only on the first BIND_MODULE subcommand for the module.

STARTING_PROCEDURE or *SP*

Name of the transfer symbol for the new module.

You use a string value for a transfer symbol whose name is not an SCL name.

If STARTING_PROCEDURE is omitted, the last transfer symbol encountered is used.

SECTION_ORDER or *SO*

Parameter Attributes: BY_NAME

Code section ordering for the component modules in the new module. Each record in the list contains a module name and its section ordinal.

PRESET_VALUE or *PV*

Parameter Attributes: BY_NAME

Specifies text record reduction as follows.

ZERO (Z)

Reduces the number of individual text records in an object module. Reducing the number of records reduces the amount of time it takes to load the module.

If PRESET_VALUE is omitted, the number of text records is not reduced.

INCLUDE_BINARY_SECTION_MAPS or *IBSM*

Parameter Attributes: BY_NAME

Indicates whether the binary section map is included in the information element for the bound module.

OUTPUT or *O*

Parameter Attributes: BY_NAME

File to which the section map for the new module is written. This file can be positioned. If OUTPUT is omitted, no section map is written.

Remarks

- The new module is not generated until you enter a GENERATE_LIBRARY subcommand. Therefore, the section map for the module is not written on the file specified on the OUTPUT parameter until the module is generated.

- A restructuring procedure uses a sequence of BIND_MODULE subcommands to direct the generation of the load module. The first subcommand in the sequence must specify the module name and the file containing the modules to be bound. Each subcommand except the last in the sequence for the module must specify MODE=CONTINUE. The last subcommand in the sequence must specify MODE=QUIT. Refer to the Application Efficiency chapter of the Object Code Management manual for more information on restructuring.

- For more information, see the NOS/VE Object Code Management manual.

Examples The following is a restructuring procedure generated for two object modules named EXAMP and NAND on file BIN3.

```
PROCEDURE MY_PROC(
  target_text,tt:file=:$LOCAL.BIN3
  restructured_module,rm:file=$LOCAL.MY_FILE
  restructured_module_name,rmn:program_name='MY_FILE'
  status)
    create_object_library
      bind_module name=$string(restructured_module_name) ..
          file=target_text mode=continue
      bind_module "EXAMP       " section_order=(('EXAMP'   1)) ..
          mode=continue
      bind_module "NAND        " section_order=(('NAND'    1)) ..
          mode=quit
      generate_library library=restructured_module
    quit
PROCEND
```

CHANGE_COMMAND_DESCRIPTION CREOL Subcommand

Purpose Changes the command description for a command processor.

Format **CHANGE_COMMAND_DESCRIPTION** or **CHACD**

NAME=list of name
STARTING_PROCEDURE=program_name
LIBRARY=keyword or *file* or *string*
SYSTEM_COMMAND_NAME=name
AVAILABILITY=keyword
SCOPE=keyword
LOG_OPTION=keyword
APPLICATION_IDENTIFIER=keyword or *name*
STATUS=status variable

Parameters **NAME or NAMES or N**

The names of the command descriptions being changed. This parameter is required.

STARTING_PROCEDURE or *SP*

Parameter Attributes: BY_NAME

Name of the command processor's starting procedure (entry point in the module). Specify a name which conforms to the type PROGRAM_NAME. For names that are not SCL or COBOL names, use a string value. An example is a C function name, in which lowercase is significant.

You may specify either this parameter or the SYSTEM_COMMAND_NAME parameter, but not both.

LIBRARY or *L*

Parameter Attributes: BY_NAME

Designates the library containing the command processor's starting procedure. Enter the name of the library, or enter the file path to the library as a string value. File paths containing $FAMILY, $USER, or $SYSTEM elements or file variable names should be entered as strings. The string is then evaluated at the time the command description is used.

The keyword OSF$CURRENT_LIBRARY specifies the library containing the command description.

You may specify this parameter only if a STARTING_PROCEDURE parameter is specified for the command description.

SYSTEM_COMMAND_NAME or *SCN*

Parameter Attributes: BY_NAME

The name of a command in the $SYSTEM command list entry. This name need not be the same as the name specified on the NAME parameter.

When this parameter is specified, the command is called by means of the library containing the command description and not actually via $SYSTEM. To the user, however, this distinction is transparent.

You may specify either this parameter or the STARTING_PROCEDURE parameter, but not both.

AVAILABILITY or *A*

Parameter Attributes: BY_NAME

Specifies whether the command is included in a display of the command list. Keyword options are:

NORMAL_USAGE (NU)

The command is included in displays of the command list as output on the DISPLAY_COMMAND_LIST_ENTRY command and other similar situations.

ADVANCED_USAGE (AU)

The command is included in displays of the user's command list but only if the user specifies the ADVANCED_USAGE display option for the DISPLAY_COMMAND_LIST_ENTRY command.

HIDDEN (H)

The command is not included in displays of the user's command list.

The default is NORMAL_USAGE.

SCOPE or *S*

Parameter Attributes: BY_NAME

The manner in which the command processor may be called. The keyword options are:

XDCL (X)

The command is externally declared and may be called from outside the object library on which it resides.

GATE (G)

The command processor can be invoked from ring brackets that are less privileged than the command processor's execution ring brackets. The GATE attribute implies the XDCL attribute.

LOCAL (L)

Reserved.

The default is XDCL.

LOG_OPTION or *LO*

Parameter Attributes: BY_NAME

Determines the manner in which calls to the command are logged. The keyword options are:

AUTOMATIC (A)

The logging is performed by the SCL Interpreter.

MANUAL (M)

Logging is performed by the command processor. Use this option to suppress logging of secure information that should not be written to a log.

The default is AUTOMATIC.

APPLICATION_IDENTIFIER or *AI*

Parameter Attributes: BY_NAME, ADVANCED

Name of the application associated with the command. When the command is executed, accounting statistics for the application are gathered.

Only a user with APPLICATION_ADMINISTRATION capability can specify an application identifier.

If this parameter is not specified, no application is associated with the command.

Remarks

- This command changes values for an already existing command description. For more information about command descriptions, see the CREATE_COMMAND_DESCRIPTION subcommand.
- For more information, see the NOS/VE Object Code Management manual.

Examples

You use this subcommand to change an existing command description. The following example changes the command description for the command processor TAPE_FILE_DUMP. The example shows the changing of the LIBRARY and STARTING_PROCEDURE for TAPE_FILE_DUMP.

```
COL/change_command_description ..
COL../name=tape_file_dump ..
COL../starting_procedure=new_tape_prog ..
COL../library=:nve.smith.program_library
```

When the TAPE_FILE_DUMP command is called, the module containing starting procedure NEW_TAPE_PROG in library :NVE.SMITH.PROGRAM_LIBRARY will be loaded.

13

CHANGE_FUNCTION_DESCRIPTION CREOL Subcommand

Purpose Changes the function description parameters for a function processor.

Format

CHANGE_FUNCTION_DESCRIPTION or
CHAFD
NAME=data_name or **list of data_name**
STARTING_PROCEDURE=program_name
LIBRARY=keyword or *file* or *string*
AVAILABILITY=keyword
SCOPE=keyword
STATUS=status variable

Parameters

NAME or **NAMES** or **N**

The names of the function descriptions being changed.

For more information about the DATA_NAME type, see the NOS/VE System Usage manual.

This parameter is required.

STARTING_PROCEDURE or *SP*

Parameter Attributes: BY_NAME

Name of the function processor's starting procedure (entry point in the module). Specify a name which conforms to the type PROGRAM_NAME. For names other than SCL or COBOL names, use a string value. An example is a C function name, in which lowercase is significant.

LIBRARY or *L*

Parameter Attributes: BY_NAME

Designates the library containing the function processor's starting procedure. Enter the name of the library, or enter the file path to the library as a string value. File paths containing $FAMILY, $USER, or $SYSTEM elements or file variable names should be entered as strings. The string is then evaluated at the time the function description is added to the task from which the function is called.

The keyword OSF$CURRENT_LIBRARY specifies the library containing the function description.

AVAILABILITY or *A*

Parameter Attributes: BY_NAME

Specifies whether the function is included in a display of the command list. Keyword options are:

NORMAL_USAGE (NU)

The command is included in displays of the command list as output on the DISPLAY_COMMAND_LIST_ENTRY command and other similar situations.

ADVANCED_USAGE (AU)

The command is included in displays of the user's command list but only if the user specifies the ADVANCED_USAGE display option for the DISPLAY_COMMAND_LIST_ENTRY command.

HIDDEN (H)

The command is not included in displays of the user's command list.

The default is NORMAL_USAGE.

SCOPE or *S*

Parameter Attributes: BY_NAME

The manner in which the function processor may be called. The keyword options are:

XDCL (X)

The function is externally declared and may be called from outside the object library on which it resides.

GATE (G)

The function processor can be invoked from rings less privileged than the function processor's execution ring brackets. The GATE attribute implies the XDCL attribute.

LOCAL (L)

Reserved.

The default is XDCL.

Remarks

- This command changes values for an already existing function description. For more information about function descriptions, see the CREATE_ FUNCTION_ DESCRIPTION subcommand.
- For more information, see the NOS/VE Object Code Management manual.

Examples

You use this subcommand to change an existing function description. The following example changes the function description for the function processor $WHO.

```
COL /change_function_description name=$who ..
COL../availability=advanced_usage
```

CHANGE_MODULE_ATTRIBUTE CREOL Subcommand

Purpose

Changes one or more attributes of a module in the module list.

Format

CHANGE_MODULE_ATTRIBUTE or
CHANGE_MODULE_ATTRIBUTES or
CHAMA
MODULE=keyword or **list of program_name** or **list of range of program_name**
NEW_NAME=program_name
SUBSTITUTE=list of record
OMIT=list of program_name
GATE=keyword or *list of program_name*
NOT_GATE=keyword or *list of program_name*
STARTING_PROCEDURE=program_name
OMIT_LIBRARY=list of name
ADD_LIBRARY=list of name
RETAIN=keyword or *list of program_name*
NOT_RETAIN=keyword or *list of program_name*
OMIT_NON_RETAINED_ENTRY_POINTS=boolean
OMIT_DEBUG_TABLES=keyword or *list of keyword*
COMMENT=string
APPLICATION_IDENTIFIER=keyword or *name*
CYBIL_PARAMETER_CHECKING=keyword
STATUS=status variable

Parameters

MODULE or **MODULES** or **M**

Modules whose attributes are changed.

You use a string value for an entry point whose name is not an SCL name or a COBOL name. An example of such a module name is a C function, where lowercase is significant.

ALL may be specified to change the attributes of all modules.

This parameter is required.

NEW_NAME or *NN*

New module name.

You use a string value for an entry point whose name is not an SCL name or a COBOL name. Other kinds of module names must be specified as a string.

If the keyword ALL is specified on the MODULE parameter, or more than one name is specified on the MODULE parameter, NEW_NAME should not be used.

If NEW_NAME is omitted, the module name is not changed.

SUBSTITUTE or *SUBSTITUTES* or *S*

Parameter Attributes: BY_NAME

List of name substitutions. Each record in the list specifies two names: the name to be replaced and the name to replace it.

You use a string value for an entry point whose name is not an SCL name or a COBOL name. An example of such a name is in the C language where lowercase is significant.

The name to be replaced can be an entry point name or the name of a CYBIL variable with the XDCL attribute. If SUBSTITUTE is omitted, no names are changed.

OMIT or *O*

Parameter Attributes: BY_NAME

List of names whose definitions are removed from the module. The name to be removed can be an entry point name or the name of a CYBIL variable with the XDCL attribute.

You use a string value for an entry point whose name is not an SCL name. If OMIT is omitted, no name definitions are removed.

GATE or *GATES* or *G*

Parameter Attributes: BY_NAME

List of entry points to which the gate attribute is added.

You use a string value for an entry point whose name is not an SCL name or a COBOL name. An example of such an entry point is a name in the C language where lowercase is significant.

If ALL is specified, the gate attribute is added to all entry points in the module.

If GATE is omitted, the gate attribute is not added to any entry point name.

NOT_GATE or *NOT_GATES* or *NG*

Parameter Attributes: BY_NAME

List of entry points from which the gate attribute is removed.

You use a string value for an entry point whose name is not an SCL name or a COBOL name. An example of such an entry point is in the C a language where lowercase is significant.

If ALL is specified, the gate attribute is removed from all entry points in the module.

If NOT_GATE is omitted, the gate attribute is not removed from any entry point.

STARTING_PROCEDURE or *SP*

Parameter Attributes: BY_NAME

Name of the entry point where execution begins.

You use a string value for an entry point whose name is not an SCL name or a COBOL name. An example of such an entry point name is in the C language where lowercase is significant.

If STARTING_PROCEDURE is omitted, the starting procedure is not changed.

OMIT_LIBRARY or *OMIT_LIBRARIES* or *OL*

Parameter Attributes: BY_NAME

List of local file names to be removed from the object text (text-embedded libraries). The local file names specify object libraries to be added to the program library list when the module is loaded. All specifications for these files are removed from the object text when the load module is written on the new object library.

If OMIT_LIBRARY is omitted, no library specifications are removed.

ADD_LIBRARY or *ADD_LIBRARIES* or *AL*

Parameter Attributes: BY_NAME

List of local file names to be added to the object text (text-embedded libraries). The local file names specify object libraries to be added to the program library list when the module is loaded. The CREOL utility adds the file specifications to each module when it writes the load module on the new object library.

If ADD_LIBRARY is omitted, no library specifications are added.

RETAIN or *R*

Parameter Attributes: BY_NAME

List of additional entry points to be given the retain attribute. An entry point with the retain attribute is kept in a new module created by combining this module with other modules.

You use a string value for an entry point whose name is not an SCL name or a COBOL name. An example of such an entry point is in the C language, where lowercase is significant.

If ALL is specified, the retain attribute is given to all entry points.

If RETAIN is omitted, no additional entry points are given the retain attribute.

NOT_RETAIN or *NR*

Parameter Attributes: BY_NAME

List of entry points from which the retain attribute is removed. Without the retain attribute, the entry point is removed from any new module created by combining this

module with other modules that reference the entry point. You use a string value for an entry point whose name is not an SCL name or a COBOL name. An example of such an entry point is in the C language where lowercase is significant.

If ALL is specified, the retain attribute is removed from all entry points.

If NOT_RETAIN is omitted, the retain attribute is not removed from any entry point.

OMIT_NON_RETAINED_ENTRY_POINTS or *ONREP*

Parameter Attributes: BY_NAME

Specifies that all entry points are removed from the module unless they are explicitly retained. If OMIT_NON_RETAINED_ENTRY_POINTS is omitted, all entry points are retained.

OMIT_DEBUG_TABLES or *OMIT_DEBUG_TABLE* or *ODT*

Parameter Attributes: BY_NAME

List of one or more keywords indicating the debug tables to be omitted when the module is loaded. Options are:

LINE_TABLE (LT)

Omits the debug table containing line numbers that correspond to the module.

SYMBOL_TABLE (ST)

Omits the debug table containing the names and addresses of the program variables in the module.

SUPPLEMENTAL_DEBUG_TABLE (SDT)

Omits the debug table containing information used to debug the module in screen mode.

PARAMETER_CHECKING (PC)

Omits parameter checking records in the module.

ALL

Omits all debug tables.

Using the OMIT_DEBUG_TABLE parameter causes the module to load faster. If it is omitted, any debug tables in the module are included when the module is loaded. (Debug tables are generated during compilation, if requested by the compiler command.)

COMMENT or *C*

Parameter Attributes: BY_NAME

Commentary stored in the module header (1 to 40 characters). If COMMENT is omitted, the commentary is not changed.

APPLICATION_IDENTIFIER or *AI*

Parameter Attributes: BY_NAME, ADVANCED

Name of the application associated with the module. When the module is executed, accounting statistics are gathered for the application. The application identifier is stored in the module header. You can associate application identifiers only with program description modules, command description modules, command procedure modules, and load modules.

Only a user with APPLICATION_ADMINISTRATION capability can specify an application identifier.

If the keyword $UNSPECIFIED is used, the application identifier is removed.

If an application identifier is already assigned and this parameter is omitted, the application identifier is not changed.

CYBIL_PARAMETER_CHECKING or *CPC*

Parameter Attributes: BY_NAME

Specifies the kind of parameter checking to be performed for each entry point in the module when the module is loaded, bound, or prelinked. This parameter affects only entry points in CYBIL modules. Options are:

SOURCE (S)

Perform parameter checking based on the source text of the entry point definition. This is a stronger type-checking algorithm.

OBJECT (O)

Perform parameter checking based on the object text of the entry point definition.

When any change occurs in a type definition referenced by an entry point definition, source text parameter checking detects parameter verification errors in CYBIL programs that reference the entry point. To override the parameter verification errors, change the TERMINATION_ERROR_LEVEL to FATAL. To eliminate the errors, recompile the program with the correct type declarations.

Parameter checking based on object text requires recompilation of programs when the structure of the interface changes. This type of parameter checking error always indicates an interface incompatibility. Recompilation is required.

Parameter checking based on object text detects the following kinds of changes:

- The number of parameters.
- The order of parameters.
- The type (integer, record, pointer, etc.) of a parameter.
- The size of any field in the fixed part of a record.
- The number of fields in the fixed part of a record.
- The size of the largest variant of a variant record.
- A change from a fixed type to an adaptable type.
- A change from an adaptable type to a fixed type.
- Array bounds.
- Component type of an array.

For example, assume a change is made to the upper or lower bound of a subrange of a type that does not affect the number of bytes in the type. Object parameter checking would not detect the change, but source parameter checking would detect the change.

If CYBIL_PARAMETER_CHECKING is omitted, the kind of parameter checking is not changed.

Remarks

- The MODULE parameter specifies the module whose attributes are changed. The module must be in the current module list.

- You specify an attribute parameter value for each attribute to be changed. If you omit an attribute parameter, the attribute value is not changed.

- The CHANGE_MODULE_ATTRIBUTES subcommand only changes the attributes of the module written by a subsequent GENERATE_LIBRARY subcommand. It does not change the attributes of the original module.
- For more information, see the NOS/VE Object Code Management manual.

Examples The following subcommand changes the name of entry point EXAMPLE in module MY_MODULE to EXAMPLE_1.

```
COL/change_module_attributes my_module ..
COL../substitute=((example,example_1))
```

CHANGE_PROGRAM_DESCRIPTION CREOL Subcommand

Purpose Changes the components of a program description.

Format **CHANGE_PROGRAM_DESCRIPTION** or **CHAPD**

NAME=list of program_name
FILE=keyword or *list of: file* or *string*
LIBRARY=keyword or *list of: keyword* or *file* or *string*
MODULE=keyword or *list of program_name*
STARTING_PROCEDURE=keyword or *program_name*
LOAD_MAP=keyword or *file* or *string*
LOAD_MAP_OPTION=keyword or *list of keyword*
TERMINATION_ERROR_LEVEL=keyword
PRESET_VALUE=keyword
STACK_SIZE=keyword or *integer*
ABORT_FILE=keyword or *file* or *string*
DEBUG_INPUT=keyword or *file* or *string*
DEBUG_OUTPUT=keyword or *file* or *string*
DEBUG_MODE=keyword or *boolean*
AVAILABILITY=keyword
SCOPE=keyword
LOG_OPTION=keyword
APPLICATION_IDENTIFIER=keyword or *name*
ARITHMETIC_OVERFLOW=keyword or *boolean*
ARITHMETIC_LOSS_OF_SIGNIFICANCE=keyword or *boolean*
DIVIDE_FAULT=keyword or *boolean*
EXPONENT_OVERFLOW=keyword or *boolean*

EXPONENT_UNDERFLOW=keyword or *boolean*
FP_INDEFINITE=keyword or *boolean*
FP_LOSS_OF_SIGNIFICANCE=keyword or *boolean*
INVALID_BDP_DATA=keyword or *boolean*
STATUS=status variable

Parameters

NAME or **NAMES** or **N**

Specifies the names of the program descriptions being changed. This parameter is required.

FILE or *FILES* or *F*

List of object files or object libraries to be unconditionally loaded when the program is executed.

Path values containing $FAMILY, $USER, or $SYSTEM elements can be supplied as strings to be evaluated when the program description is used.

If the FILE parameter is omitted, the FILE parameter of the program description is not changed. If $UNSPECIFIED is used, the FILE parameter is removed from the program description.

LIBRARY or *LIBRARIES* or *L*

List of library files to be added to the program library list when the program is executed. A file value is evaluated when the object library is generated. Path values containing $FAMILY, $USER, or $SYSTEM elements can be supplied as strings to be evaluated when the program description is used.

If $UNSPECIFIED is used, the LIBRARY parameter is removed from the program description.

The keyword OSF$TASK_SERVICES_LIBRARY specifies the system table, and keyword OSF$CURRENT_LIBRARY represents the library that contains the program description being changed.

If the LIBRARY parameter is omitted, the LIBRARY parameter of the program description is not changed.

MODULE or *MODULES* or *M*

List of modules to be loaded from the program library list when the program is executed. The modules are loaded in the order in which they are specified.

For module names which are not SCL or COBOL names, use a string value. An example is a C function name, in which lowercase is significant.

If the MODULE parameter is omitted, the MODULE parameter of the program description is not changed. If $UNSPECIFIED is used, the MODULE parameter is removed from the program description.

STARTING_PROCEDURE or *SP*

Name of the entry point at which program execution begins.

You use a string value for an entry point whose name is not an SCL or COBOL name. An example is the name of a C function in which lowercase is significant.

If the STARTING_PROCEDURE parameter is omitted, the STARTING_PROCEDURE parameter of the program description is not changed. If $UNSPECIFIED is used, the STARTING_PROCEDURE parameter is removed from the program description.

LOAD_MAP or *LM*

Parameter Attributes: BY_NAME

File on which the load map is written. A file value is evaluated when the object library is generated.

Path values containing $FAMILY, $USER, or $SYSTEM elements can be supplied as strings to be evaluated when the program description is used.

If the LOAD_MAP parameter is omitted, the LOAD_MAP parameter of the program description is not changed. If $UNSPECIFIED is used, the LOAD_MAP parameter is removed from the program description.

LOAD_MAP_OPTION or *LOAD_MAP_OPTIONS* or *LMO*

Parameter Attributes: BY_NAME

List one or more keywords indicating the information to include in the load map. Options are:

NONE

No load map is written.

SEGMENT (S)

Segment map.

BLOCK (B)

Block map.

ENTRY_POINT (EP)

Entry point map.

CROSS_REFERENCE (CR)

Entry point cross-reference.

ALL

Selects SEGMENT, BLOCK, ENTRY_POINT, and CROSS_REFERENCE.

$UNSPECIFIED

The LOAD_MAP_OPTION parameter is removed from the program description.

If the LOAD_MAP_OPTION parameter is omitted, the LOAD_MAP_OPTION parameter of the program description is not changed.

TERMINATION_ERROR_LEVEL or *TEL*

Parameter Attributes: BY_NAME

Specifies the severity level of error that terminates program loading. Options are:

WARNING (W)

Warning, error, or fatal severity level errors.

ERROR (E)

Error or fatal severity level errors.

FATAL (F)

Fatal severity level errors.

$UNSPECIFIED

The TERMINATION_ERROR_LEVEL parameter is removed from the program description.

If the TERMINATION_ERROR_LEVEL parameter is omitted, the TERMINATION_ERROR_LEVEL parameter of the program description is not changed.

PRESET_VALUE or *PV*

Parameter Attributes: BY_NAME

Value to store in all uninitialized data words. Options are:

ZERO (Z)

All zeroes.

FLOATING_POINT_INDEFINITE (FPI)

Floating-point indefinite value.

INFINITY (I)

Floating-point infinite value.

ALTERNATE_ONES (AO)

Alternating 0 and 1 bits; the leftmost (highest order) bit is 1.

UNSPECIFIED

The PRESET_VALUE parameter is removed from the program description.

If the PRESET_VALUE parameter is omitted, the parameter of the program description is not changed.

STACK_SIZE or *SS*

Parameter Attributes: BY_NAME

Maximum number of bytes in the run-time stack. The program uses the run-time stack for procedure call linkages and local variables. If STACK_SIZE is omitted, the system default value is used. You can display the default stack size by entering a DISPLAY_PROGRAM_ATTRIBUTE command. If $UNSPECIFIED is used, the STACK_SIZE parameter is removed from the program description.

ABORT_FILE or *AF*

Parameter Attributes: BY_NAME

File containing Debug commands to be processed if the program aborts. The commands are executed only if the program is not executed in Debug mode. A file value is evaluated when the object library is generated.

Path values containing $FAMILY, $USER, $WORKING_CATALOG, or $SYSTEM elements can be supplied as strings to be evaluated when the program description is used.

If ABORT_FILE is omitted, the program description for the ABORT_FILE parameter is not changed. If $UNSPECIFIED is used, the ABORT_FILE parameter is removed from the program description.

DEBUG_INPUT or *DI*

Parameter Attributes: BY_NAME

File containing Debug commands. The commands are read only if the program is executed under the control of Debug (refer to the DEBUG_MODE parameter). This file can be positioned. A file value is evaluated when the object library is generated.

Path values containing $FAMILY, $USER, $WORKING_CATALOG, or $SYSTEM elements can be supplied as strings to be evaluated when the program description is used.

If DEBUG_INPUT is omitted, the DEBUG_INPUT parameter of the program description is not changed. If $UNSPECIFIED is used, the DEBUG_INPUT parameter is removed from the program description.

DEBUG_OUTPUT or *DO*

Parameter Attributes: BY_NAME

File on which Debug output is written. Output is written only if the program is executed in Debug mode. This file can be positioned. A file value is evaluated when the object library is generated.

Path values containing $FAMILY, $USER, $WORKING_CATALOG, or $SYSTEM elements can be supplied as strings to be evaluated when the program description is used.

If DEBUG_OUTPUT is omitted, the DEBUG_OUTPUT parameter of the program description is not changed. If $UNSPECIFIED is used, the DEBUG_OUTPUT parameter is removed from the program description.

DEBUG_MODE or *DM*

Parameter Attributes: BY_NAME

Indicates whether the program is to be run under the control of Debug. (For information on using Debug, refer to the program's specific source language manual.) Options are:

ON

Program executed under control of the Debug program.

OFF

Program executed without the Debug program.

If the DEBUG_MODE parameter is omitted, the DEBUG_MODE parameter of the program description is not changed. If $UNSPECIFIED is used, the DEBUG_MODE parameter is removed from the program description.

AVAILABILITY or *A*

Parameter Attributes: BY_NAME

Specifies whether or not the program description is made known to users as a command. Options are:

NORMAL_USAGE (ADVERTISED, A, or NU)

Program description appears in the output produced by the DISPLAY_COMMAND_LIST_ENTRY command (and in similar situations).

ADVANCED_USAGE (AU)

The command is included in displays of the user's command list if the user specifies the ADVANCED_USAGE display option for the DISPLAY_COMMAND_LIST_ENTRY command.

HIDDEN (H)

Program description is suppressed from the output produced by DISPLAY_COMMAND_LIST_ENTRY command (and in similar situations).

If this parameter is omitted, the AVAILABILITY parameter of the program description is not changed.

SCOPE or *S*

Parameter Attributes: BY_NAME

The manner in which the command processor may be called. The keyword options are:

XDCL (X)

The command is externally declared and may be called from outside the object library on which it resides.

GATE (G)

The program can be invoked from rings less privileged than the program's execution ring brackets. The GATE attribute implies the XDCL attribute.

LOCAL (L)

Reserved.

The default is XDCL.

LOG_OPTION or *LO*

Parameter Attributes: BY_NAME

Determines the manner in which calls to the program are logged. The keyword options are:

AUTOMATIC (A)

The logging is performed by the SCL Interpreter.

MANUAL (M)

Logging is performed by the program. Use this option to suppress the logging of secure information that should not be written to a log.

The default is AUTOMATIC. If you omit this parameter, the logging option for the program description is not changed.

APPLICATION_IDENTIFIER or *AI*

Parameter Attributes: BY_NAME, ADVANCED

Name of application associated with the program. When the program is executed, accounting statistics will be emitted for the application. The application identifier is stored in the module header.

Only a user with APPLICATION_ADMINISTRATION capability can specify an application identifier.

If the keyword $UNSPECIFIED is used, the application identifier is removed. If this parameter is omitted, the application identifier is not changed.

ARITHMETIC_OVERFLOW or *AO*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition ARITHMETIC_OVERFLOW causes an interrupt. Valid specifications are:

ON

ARITHMETIC_OVERFLOW is enabled. The condition causes an interrupt.

OFF

ARITHMETIC_OVERFLOW is disabled. The condition does not cause an interrupt.

ARITHMETIC_LOSS_OF_SIGNIFICANCE or *ALOS*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition ARITHMETIC_LOSS_OF_SIGNIFICANCE causes an interrupt. Valid specifications are:

ON

ARITHMETIC_LOSS_OF_SIGNIFICANCE is enabled. The condition causes an interrupt.

OFF

ARITHMETIC_LOSS_OF_SIGNIFICANCE is disabled. The condition does not cause an interrupt.

DIVIDE_FAULT or *DF*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition DIVIDE_FAULT causes an interrupt. Valid specifications are:

ON

DIVIDE_FAULT is enabled. The condition causes an interrupt.

OFF

DIVIDE_FAULT is disabled. The condition does not cause an interrupt.

EXPONENT_OVERFLOW or *EO*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition EXPONENT_OVERFLOW causes an interrupt. Valid specifications are:

ON

EXPONENT_OVERFLOW is enabled. The condition causes an interrupt.

OFF

EXPONENT_OVERFLOW is disabled. The condition does not cause an interrupt.

EXPONENT_UNDERFLOW or *EU*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition EXPONENT_UNDERFLOW causes an interrupt. Valid specifications are:

ON

EXPONENT_UNDERFLOW is enabled. The condition causes an interrupt.

OFF

EXPONENT_UNDERFLOW is disabled. The condition does not cause an interrupt.

FP_INDEFINITE or *FPI* or *FI*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition FP_INDEFINITE causes an interrupt. Valid specifications are:

ON

FP_INDEFINITE is enabled. The condition causes an interrupt.

OFF

FP_INDEFINITE is disabled. The condition does not cause an interrupt.

FP_LOSS_OF_SIGNIFICANCE or *FPLOS* or *FLOS*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition FP_LOSS_OF_SIGNIFICANCE causes an interrupt. Valid specifications are:

ON

FP_LOSS_OF_SIGNIFICANCE is enabled. The condition causes an interrupt.

OFF

FP_LOSS_OF_SIGNIFICANCE is disabled. The condition does not cause an interrupt.

INVALID_BDP_DATA or *IBDPD* or *IBD*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition INVALID_BDP_DATA causes an interrupt. Valid specifications are:

ON

INVALID_BDP_DATA is enabled. The condition causes an interrupt.

OFF

INVALID_BDP_DATA is disabled. The condition does not cause an interrupt.

Remarks

- To allow users the option of rescinding a previously specified value or of not including a given parameter in the CHAPD command, the keyword $UNSPECIFIED may be used for some parameters. This removes the parameter from the description. The result of using $UNSPECIFIED is the same as not supplying the parameter on the CREATE_PROGRAM_DESCRIPTION subcommand. When the program is executed, the corresponding job default program attribute value is used.
- For more information, see the NOS/VE Object Code Management manual.

Examples See the NOS/VE Object Code Management manual for a detailed example.

COMBINE_MODULE CREOL Subcommand

Purpose Adds new modules and replaces existing modules in the module list.

Format

COMBINE_MODULE or
COMBINE_MODULES or
COMM
LIBRARY=list of file
MODULE=list of program_name or *list of range of program_name*
PLACEMENT=keyword
DESTINATION=program_name
STATUS=status variable

Parameters

LIBRARY or **LIBRARIES** or **L**

Object files, SCL procedure files, or object library files containing the modules to be combined. This parameter is required.

MODULE or *MODULES* or *M*

Modules to be combined.

You use a string value for a module whose name is not an SCL name or a COBOL name. An example of such a module name is in the C language where lowercase is significant.

If MODULE is omitted, all modules on the specified files or libraries are combined.

PLACEMENT or *P*

Indicates whether the added modules are placed before or after the module specified on the DESTINATION parameter. Options are:

BEFORE (B)

Modules added before the destination module.

AFTER (A)

Modules added after the destination module.

If PLACEMENT is omitted, AFTER is used.

DESTINATION or *D*

Module before or after which the added modules are placed.

This parameter does not affect the location of replacement modules. A replacement module is always placed in the same location as the module it replaces.

If DESTINATION is omitted, added modules are placed according to the PLACEMENT parameter value. If the value of PLACEMENT is BEFORE, the modules are placed at the beginning of the library. If the value of PLACEMENT is AFTER, the modules are placed at the end of the library.

Remarks

- The COMBINE_MODULES subcommand can specify object files, SCL procedure files, or object libraries that are processed in the order you specify the files on the LIBRARY parameter.

- The COMBINE_MODULES subcommand checks for duplicate modules in the specified files and reports an error if duplicates are found.

  You can, however, combine modules in libraries with duplicate modules. You add one of the libraries to the module list with an ADD_MODULES subcommand and then perform a COMBINE_MODULES of the second library.

- If you do not want to use all modules in a file, specify the modules to be used on the MODULE parameter.

- A module to be combined replaces an existing module with the same name in the module list. If the name is not already in the module list, the module to be combined is added to the module list.

- A replacement module is placed in the same location as the module it replaces. An added module is added at the end of the list, unless you specify another location with the DESTINATION and PLACEMENT parameters. You can change the module order later with a REORDER_MODULES subcommand.

- If you specify an SCL procedure whose header references a non-standard type, you must make the type definition available. For instance, if you want to add the following procedure:

```
PROCEDURE show(
   p1: address_list = $required
   status)
  :
PROCEND
```

  then the type definition for ADDRESS_LIST must be created outside the procedure. This is accomplished by using the TYPE control statement, as in:

```
TYPE
   address_list : list 1..3 of string
TYPEND
```

- For more information, see the NOS/VE Object Code Management manual.

Examples

The following subcommand combines all modules in files MY_LIBRARY and YOUR_LIBRARY with the modules already in the module list.

```
COL/combine_module (my_library,your_library)
```

CREATE_APPLICATION_MENU CREMM Subcommand

Purpose Initiates the CREATE_APPLICATION_MENU utility session.

Format

CREATE_APPLICATION_MENU or
CREAM
NAME=name
STATUS=status variable

Parameters **NAME** or **N**

Specifies the name of the application menu. The NAME parameter is a string containing 1 through 31 characters. This parameter is required.

Remarks For more information, see the NOS/VE Object Code Management manual.

CREATE_BRIEF_HELP_MESSAGE CREMM Subcommand

Purpose Creates a brief description of a command. The complete description is generated by the CREATE_FULL_HELP_MESSAGE subcommand.

Format

CREATE_BRIEF_HELP_MESSAGE or
CREBHM
COLLECT_TEMPLATE_UNTIL=string
STATUS=status variable

Parameters *COLLECT_TEMPLATE_UNTIL* or *CTU*

Specifies the termination string to use when collecting the template of the brief help message. If the COLLECT_TEMPLATE_UNTIL parameter is omitted, the string '**' is used.

Remarks For more information, see the NOS/VE Object Code Management manual.

Examples The following example creates a brief help message.

```
CMM/create_brief_help_message
? The DISPLAY_FILE command displays information
? about the specified file.
? **
CMM/
```

CREATE_COMMAND_DESCRIPTION CREOL Subcommand

Purpose Defines the means of access to a command. This subcommand identifies either a program to be dynamically loaded or a system-supplied command.

NOTE

Do not create command descriptions for command processors written in languages other than CYBIL. The run-time library routines for other languages (such as FORTRAN) may depend on static initialization of data or may call the PMP$EXIT or PMP$ABORT procedures.

Format

CREATE_COMMAND_DESCRIPTION or
CRECD
NAME=record
STARTING_PROCEDURE=program_name
LIBRARY=keyword or *file* or *string*
SYSTEM_COMMAND_NAME=name
AVAILABILITY=keyword
SCOPE=keyword
LOG_OPTION=keyword
MERGE_OPTION=keyword
APPLICATION_IDENTIFIER=name
STATUS=status variable

Parameters

NAME or **NAMES** or **N**

The command name and its aliases; the name by which the user calls the command. Specify the names in a record with the following format:

```
record
  name: name
  aliases: list rest of name = $optional
recend
```

For example, specify this parameter as follows:

```
name = (tape_file_dump, tapfd, tfd)
```

The first name qualifies as the NAME field in the record followed by the names for the ALIASES field.

This parameter is required.

STARTING_PROCEDURE or *SP*

Parameter Attributes: BY_NAME

Name of the command processor's starting procedure (entry point in the module). Specify a name which conforms to the type PROGRAM_NAME. For names other than SCL or COBOL names, use a string value. An example is a C function name, in which lowercase is significant.

You must specify either this parameter or the SYSTEM_COMMAND_NAME parameter, but not both.

LIBRARY or *L*

Parameter Attributes: BY_NAME

Designates the library containing the command processor's starting procedure. Enter the name of the library, or enter the file path to the library as a string value. File paths containing $FAMILY, $USER, or $SYSTEM elements or file variable names should be entered as strings. The string is then evaluated at the time the command description is used.

The keyword, OSF$CURRENT_LIBRARY specifies the library containing the command description.

You may specify this parameter only if you specify the STARTING_PROCEDURE parameter.

SYSTEM_COMMAND_NAME or *SCN*

Parameter Attributes: BY_NAME

The name of a command in the $SYSTEM command list entry. This name need not be the same as the name specified on the NAME parameter.

When this parameter is specified, the command is called by means of the library containing the command description and not actually via $SYSTEM. To the user, however, this distinction is transparent.

You must specify either this parameter or the STARTING_PROCEDURE parameter, but not both.

AVAILABILITY or *A*

Parameter Attributes: BY_NAME

Specifies whether the command is included in a display of the command list. Keyword options are:

NORMAL_USAGE (NU)

The command is included in displays of the command list as output on the DISPLAY_COMMAND_LIST_ENTRY command and other similar situations.

ADVANCED_USAGE (AU)

The command is included in displays of the user's command list if the user specifies the ADVANCED_USAGE display option for the DISPLAY_COMMAND_LIST_ENTRY command.

HIDDEN (H)

The command is not included in displays of the user's command list.

The default is NORMAL_USAGE.

SCOPE or *S*

Parameter Attributes: BY_NAME

The manner in which the command processor may be called. The keyword options are:

XDCL (X)

The command is externally declared and may be called from outside the object library on which it resides.

GATE (G)

The command processor can be invoked from rings less privileged than the command processor's execution ring brackets. The GATE attribute implies the XDCL attribute.

LOCAL (L)

Reserved.

The default is XDCL.

LOG_OPTION or *LO*

Parameter Attributes: BY_NAME

Determines the manner in which calls to the command are logged. The keyword options are:

AUTOMATIC (A)

The logging is performed by the SCL Interpreter.

MANUAL (M)

Logging is performed by the command processor. Use this option to suppress the logging of secure information that should not be written to a log.

The default is AUTOMATIC.

MERGE_OPTION or *MO*

Parameter Attributes: BY_NAME

Indicates whether the module containing the command description will be added or replaced within the module. The keyword options are:

ADD (A)

Adds the module to the end of the module list.

REPLACE (R)

Replaces the module in the current module list which has the same name.

COMBINE (C)

Adds the new module to the end of the module list unless a module of the same name already exists, in which case that module is replaced with the new module.

The default is COMBINE.

APPLICATION_IDENTIFIER or *AI*

Parameter Attributes: BY_NAME, ADVANCED

Name of the application associated with the command. When the command is executed, accounting statistics for the application are gathered.

Only a user with APPLICATION_ADMINISTRATION capability can specify an application identifier.

If this parameter is not specified, no application is associated with the command.

Remarks

- This subcommand creates a command description similar to an abbreviated program description which is used to control access to the command.
- Access to the command processor is by means of either a $SYSTEM command or by a starting procedure as established with the parameters on this command.
- The command processor is loaded into the task from which the command is called. Because access to the command is through a command description, a separate task is not generated and the command processor has access to the data already in use in the current task. (Calls to commands defined within program descriptions always create new, separate tasks).
- For command descriptions which name a command in $SYSTEM, the SCL Interpreter executes the command processor. However, if the command description names a starting procedure, the loader is called to execute the command.
- For command descriptions naming a starting procedure and, optionally, an object library, the dynamic loading of the command processor occurs as follows:
  - If the starting procedure is already loaded in the current task, no loading occurs; the process is complete.
  - If a library was specified as part of the command description the library is the searched for the module containing the starting procedure.
  - If no library is specified as part of the command description, the program library list of the current task is searched for a module containing the starting procedure.

- If you choose to have your command processor implemented using a command description, take care not to call the CYBIL procedures PMP$EXIT or PMP$ABORT in the code for the command processor. Both CYBIL procedures will cause task termination.

- To read about the loading process and satisfying of external references for these methods of loading, see The Loading Process in Detail in the NOS/VE Object Code Management manual.

  - For more information, see the NOS/VE Object Code Management manual.

Examples

The following examples show how to create command descriptions that access commands either through a starting procedure or by means of a system command. The first example defines access to a TAPE_FILE_DUMP command (and its aliases) through procedure TAPE_FILE_PROG in library :NVE.SMITH.PROGRAM_LIBRARY.

```
COL/create_command_description ..
COL../name=(tape_file_dump, dump_tape, tapfd) ..
COL../starting_procedure=tape_file_prog ..
COL../library=:nve.smith.program_library ..
COL../scope=xdcl availability=normal_usage ..
COL../merge_option=add
        :
COL/generate_library library=:nve.smith.tape_command_library
```

The next example shows how to create access to a DELETE_FILE command and its aliases through the system command, DELETE_FILE.

```
COL/create_command_description ..
COL../name=(delete_file, delete, delf) ..
COL../system_command_name=delete_file
        :
COL/generate_library library=:nve.smith.delete_commands
```

At the completion of the session, you add the generated library to the command list, and, if you want to restrict users to the use of just the commands in the library, you can delete the $SYSTEM command list entry for those users.

For more information about command lists, see the NOS/VE System Usage manual.

13

CREATE_FORM_MODULE CREOL Subcommand

Purpose CREATE_FORM_MODULE starts the CREATE_FORM_MODULE utility, which is subordinate to the CREOL utility. Using the subcommands of CREATE_FORM_MODULE, you create a form.

Format

CREATE_FORM_MODULE or
CREFM
FORM_NAME=name
MERGE_OPTION=keyword
STATUS=status variable

Parameters

FORM_NAME or **FN**

The name of the form. The application program uses this name to open the form. This parameter is required.

MERGE_OPTION or *MO*

Specifies how to merge the form module you are creating with existing modules on the object library. The following values are valid:

ADD or A

Adds the module to the object library. If a module with the same name already exists on the library, ADD does not replace it. (To replace an existing module, use REPLACE or COMBINE.)

REPLACE or R

Replaces a module on the library with the module you are creating, if they have the same names. REPLACE does not add a new module. (To add a new module, use either ADD or COMBINE.)

COMBINE or C

Adds a new module or replaces an existing module.

The default is COMBINE.

Remarks

- The CREATE_FORM_MODULE subcommand establishes default attributes for the form. (For a list of the attributes and their defaults, see the description of the SET_FORM subcommand of the CREATE_FORM_MODULE utility.)

- The END_FORM_MODULE subcommand ends the creation of a form and quits the CREATE_FORM_MODULE utility.
- For information about screen formatting, form modules, and adding form modules to object libraries, see the NOS/VE Screen Formatting manual.

Examples The following example shows how to create form SELECT_FORM for use with a COBOL program.

```
COL/create_form_module form_name=select_form
CFM/set_form form_processor=cobol
CFM/add_event program_event=compute ..
CFM../terminal_event=next action=return_normal ..
CFM../label='COMP'
CFM/      more create_form_module subcommands
                    :
CFM/end_form_module
COL/generate_library library=:nve.bonnie.forms_library
```

CREATE_FULL_HELP_MESSAGE CREMM Subcommand

Purpose Creates a message containing a complete description of the command. A brief description is generated with the CREATE_BRIEF_HELP_MESSAGE subcommand.

Format **CREATE_FULL_HELP_MESSAGE** or **CREFHM**

COLLECT_TEMPLATE_UNTIL=string
STATUS=status variable

Parameters *COLLECT_TEMPLATE_UNTIL* or *CTU*

Specifies the termination string to use when collecting the template of the full help message. If the COLLECT_TEMPLATE_UNTIL parameter is omitted, the string '**' is used.

Remarks For more information, see the NOS/VE Object Code Management manual.

Examples The following example creates a full help message.

```
CMM/create_full_help_message
? The DISPLAY_FILE command displays information about
? the file onto your terminal screen.  You can
? specify the level of detail by entering a value
? for the DISPLAY_OPTIONS parameter of this command.
? **
CMM/
```

CREATE_FUNCTION_DESCRIPTION CREOL Subcommand

Purpose Creates a function description which is used to load the program containing the function processor when the function is called.

NOTE

Do not create function descriptions for function processors written in languages other than CYBIL. The run-time library routines for other languages (such as FORTRAN) may depend on static initialization of data or may call the PMP$EXIT or PMP$ABORT procedures.

Format

CREATE_FUNCTION_DESCRIPTION or
CREFD
NAME=data_name or **record**
STARTING_PROCEDURE=program_name
LIBRARY=keyword or *file* or *string*
AVAILABILITY=keyword
SCOPE=keyword
MERGE_OPTION=keyword
STATUS=status variable

Parameters

NAME or **NAMES** or **N**

The name and aliases by which the function is called. The name you specify must conform to the type DATA_NAME. SCL names are valid DATA_NAME types.

You can specify a single name or you can specify a function name and aliases, using a record which has the following format:

```
record
  name: data_name
  aliases: list rest of data_name = $optional
recend
```

For example, the following shows how to specify the name of a function and its aliases using this parameter:

```
name=($tape_dump_files, $tdf)
```

For more information about the DATA_NAME type, see the NOS/VE System Usage manual.

This parameter is required.

STARTING_PROCEDURE or SP

Parameter Attributes: BY_NAME

Name of the function processor's starting procedure (entry point in the module). Specify a name which conforms to the type PROGRAM_NAME. For names other than SCL or COBOL names, use a string value. An example is a C function name, in which lowercase is significant.

This parameter is required.

LIBRARY or *L*

Parameter Attributes: BY_NAME

Designates the library containing the function processor's starting procedure. Enter the name of the library, or enter the file path to the library as a string value. File paths containing $FAMILY, $USER, or $SYSTEM elements or file variable names should be entered as strings. The string is then evaluated at the time the function description used.

The keyword OSF$CURRENT_LIBRARY specifies the library containing the function description.

AVAILABILITY or *A*

Parameter Attributes: BY_NAME

Specifies whether the function is included in a display of the command list. Keyword options are:

NORMAL_USAGE (NU)

The command is included in displays of the command list as output on the DISPLAY_COMMAND_LIST_ENTRY command and other similar situations.

ADVANCED_USAGE (AU)

The command is included in displays of the user's command list but only if the user specifies the ADVANCED_USAGE display option for the DISPLAY_COMMAND_LIST_ENTRY command.

HIDDEN (H)

The command is not included in displays of the user's command list.

The default is NORMAL_USAGE.

SCOPE or *S*

Parameter Attributes: BY_NAME

The manner in which the function processor may be called. The keyword options are:

XDCL (X)

The function is externally declared and may be called from outside the object library on which it resides.

GATE (G)

The function processor can be invoked from rings less privileged than the processor's execution ring brackets. The GATE attribute implies the XDCL attribute.

LOCAL (L)

Reserved.

The default is XDCL.

MERGE_OPTION or *MO*

Parameter Attributes: BY_NAME

Indicates whether the module containing the function description will be added or replaced within the module list. The keyword options are:

ADD (A)

Added to the end of the module list.

REPLACE (R)

Replaces the module in the current module list which has the same name.

COMBINE (C)

Added to the end of the module list if a module having the same name does not exist. If a module having the same name does exist, this option acts the same as the REPLACE keyword option.

The default is COMBINE.

Remarks

- This subcommand creates a function description (similar to an abbreviated program description) which is used to control access to the function.
- The function is loaded into the task from which it is called. A separate task is not generated and the function processor has access to the data already in use in the current task.
- The dynamic loading of the function processor occurs as follows:
  - If the starting procedure is already loaded in the current task, no loading is required..
  - If a library was specified as part of the function description, the library is searched for the module containing the starting procedure.
  - If no library is specified as part of the function description, the current task's program library list is searched for a module containing the starting procedure.
- To read about the loading process and satisfying external references for these methods of loading, see the section titled The Loading Process in Detail in the NOS/VE Object Code Management manual.
  - For more information, see the NOS/VE Object Code Management manual.

Examples The following examples show how to create a function description that accesses a function processor through a starting procedure in a library. The description defines access to the $LIST_TAPE_FILES function (and its aliases) through procedure TAPE_FILE_FUNCTION in library :NVE.SMITH.PROGRAM_LIBRARY.

```
COL/create_function_description ..
COL../name=($list_tape_file, $ltf) ..
COL../starting_procedure=tape_file_function ..
COL../library=:nve.smith.program_library
  :
COL/generate_library library=:nve.smith.tape_command_library
```

To make the function available to users, the library of function descriptions must be added to the command list.

For more information about command lists, see the NOS/VE System Usage manual.

CREATE_LINKED_MODULE CREOL Subcommand

Purpose Creates a prelinked module from an existing module and adds it to the module list.

Format

CREATE_LINKED_MODULE or
CRELM
NAME=program_name
COMPONENT=list of record
RING_BRACKETS=record
RETAIN_COMMON_BLOCK=keyword or *list of program_name*
IGNORE_SECTION_NAMES=boolean
STARTING_SEGMENT=integer
OUTPUT=file
DEBUG_TABLE=file
NEXT_AVAILABLE_SEGMENT=integer variable
APPLICATION_IDENTIFIER=name
DEFER_ENTRY_POINTS=keyword or *record* or *list of program_name*
DEFER_COMMON_BLOCKS=keyword or *record* or *list of program_name*
STATUS=status variable

Parameters

NAME or N

Name of the new prelinked module.

For modules with other than SCL or COBOL names, use a string value. An example is a C function name, in which lowercase is significant.

This parameter is required.

COMPONENT or COMPONENTS or C

Component modules of the new module. Each item in the list is a record consisting of a file name followed by a series of module names which are to be used. A range of names may be specified. If no module names are specified for a file, all modules on the file are used.

For modules with other than SCL or COBOL names, use a string value. An example is a C function name, in which lowercase is significant.

NOTE

A component module can be only an object, load, or bound module.

This parameter is required. At least one file must be specified.

RING_BRACKETS or *RB*

Specifies three integers representing the r1, r2, and r3 ring execution values for the new module. The ring values can be from 3 through 15. If RING_BRACKETS is omitted, all three values will default to the current ring.

RETAIN_COMMON_BLOCK or *RETAIN_COMMON_BLOCKS* or *RCB*

Parameter Attributes: BY_NAME, ADVANCED

Specifies which common block names are retained in the new modules. The keyword ALL specifies that all common blocks are retained.

For a common block with other than an SCL or COBOL name, use a string value. An example is a C function name, in which lowercase is significant.

If RETAIN_COMMON_BLOCK is omitted, no common blocks are retained.

IGNORE_SECTION_NAMES or *ISN*

Parameter Attributes: BY_NAME, ADVANCED

Specifies whether working storage sections with different names should be placed in unique segments. If IGNORE_SECTION_NAMES is omitted or IGNORE_SECTION_NAMES=TRUE, sections with similar access attributes (read and write) are placed in the same segments, regardless of section name.

STARTING_SEGMENT or *SS*

Parameter Attributes: BY_NAME, ADVANCED

First segment number to use in prelinking this module. The STARTING_SEGMENT parameter provides a unique starting segment number. It is used only when creating multiple prelinked modules that are loaded together.

Use the NEXT_AVAILABLE_SEGMENT parameter to generate the integer value for the STARTING_SEGMENT parameter on the next CREATE_LINKED_MODULE subcommand.

Integer values are 0 through 4,095. The operating system reserves segments 36 through 63 for prelinked programs. Each program must fit into these segments. Do not use segments 0 through 35, and 64 through 4,095.

If STARTING_SEGMENT is omitted, the integer value 36 is used as the starting segment number.

OUTPUT or *O*

File to which the prelink information and diagnostics are written. This file can be positioned.

If OUTPUT is omitted, information is written to file $LOCAL.LINKMAP.

DEBUG_TABLE or *DT*

File to which the table containing binary debug information is written. This parameter is for Control Data internal use only.

If DEBUG_TABLE is omitted, no debug information is written.

NEXT_AVAILABLE_SEGMENT or *NAS*

Parameter Attributes: BY_NAME, ADVANCED

Integer variable which returns the address of the next available segment number. Use this parameter only for creating multiple prelinked modules which will be loaded together. This parameter generates unique segment numbers which will be used by the STARTING_SEGMENT parameter on the next CREATE_LINKED_MODULE subcommand.

If this parameter is omitted, no segment number is returned.

APPLICATION_IDENTIFIER or *AI*

Parameter Attributes: BY_NAME, ADVANCED

Name of the application identifier stored in the module header and included on the application accounting statistics when the software is executed.

Only a user with APPLICATION_ADMINISTRATION capability can specify an application identifier.

If an application identifier is placed on a load module, the module is assumed to be a unit-measured application.

If APPLICATION_IDENTIFIER is omitted, no application identifier is assigned to the module.

DEFER_ENTRY_POINTS or *DEP*

Parameter Attributes: BY_NAME, ADVANCED

Defers the loading of entry points in the module. Specify the keywords ALL, NONE, or $NOT_RETAINED ($NR) for this parameter, or specify one or more entry points whose names conform to the type PROGRAM_NAME.

For example, to defer the loading of entry points EP1 and EP2, enter them as a list of names as follows:

```
defer_entry_points=(ep1, ep2)
```

Alternatively, you can defer loading of all except certain specified entry points, using a record which has the following format:

```
record
  action: key
    ($defer_all_except, $dae)
    keyend
  entry_points: list rest of program_name
recend
```

For example, to defer loading of all entry points except EP1 and EP2, enter the following:

```
defer_entry_points=($defer_all_except (ep1, ep2))
```

The default is NONE; all entry points in the module are loaded.

DEFER_COMMON_BLOCKS or *DCB*

Parameter Attributes: BY_NAME, ADVANCED

Defers the loading of common blocks specified on this parameter. You can specify the keywords ALL or NONE, or you can provide a list of names that conform to the type PROGRAM_NAME. For example, to defer the loading of common blocks COM1 and COM2, specify them as a list of names as follows:

```
defer_common_blocks=(com1, com2)
```

Alternatively, you can defer the loading of all common blocks except those specified by using a record with the following format:

```
record
  action: key
    ($defer_all_except, $dae)
  keyend
  common_blocks: list rest of program_name
recend
```

For example, to defer loading of all common blocks except COM1 and COM2, enter the following:

```
defer_common_blocks=($defer_all_except (com1,com2))
```

The default is NONE; all common blocks in the module are loaded.

For more information about using the DEFERRED_ENTRY_POINTS and DEFERRED_COMMON_BLOCKS parameters, see Deferred Loading of Entry Points and Common Blocks in chapter 8.

Remarks

- When building programs that consist of multiple prelinked modules, all predefined segment numbers must be unique for the entire load sequence.
- Use the STARTING_SEGMENT parameter on the CREATE_LINKED_MODULE subcommand to specify the first reserved segment number for a module. This allows modules that are prelinked separately to be used together at execution time.
- The system issues a warning diagnostic message for all text-embedded libraries encountered during prelinking. If the warning is ignored, the loader attempts to satisfy text-embedded library references at load time.
- During prelinking, an output file is generated that contains diagnostics and information on how the program was prelinked. This link map's default file name is $LOCAL.LINKMAP.
- Do not prelink COBOL programs that use CALL and CANCEL into a single module because CALL will try to overlay a single component module that is no longer available.
- Once you have prelinked modules, they can no longer be debugged using the interactive debugger. The debug information written to the file specified by the DEBUG_TABLE parameter is not the same as the debug tables used by the interactive debugger.
- For more information, see the NOS/VE Object Code Management manual.

Examples

The following sequence creates a prelinked module named PRELINKED_MODULE from component BOUND_PRODUCT with ring brackets of (11,11,11). The module is then put in object library PRELINKED_PRODUCT and executed.

```
/create_object_library
COL/create_linked_module name=prelinked_module ..
COL../component=bound_product ..
COL../ring_brackets=(11,11,11)
COL/generate_library prelinked_product
COL/quit
/execute_task ..
../starting_procedure=product_entry_point ..
../library=prelinked_product
```

CREATE_MENU_CLASS CREAM Subcommand

Purpose Creates a class for an application menu. A class is defined as a name for a submenu. This subcommand allows you to identify a grouping of menu items. Up to 16 menu classes can be defined for a menu.

Format

CREATE_MENU_CLASS or
CREMC
NAME=string
STATUS=status variable

Parameters

NAME or **N**

Specifies the identification for the menu class being defined. Menu class names must be unique within a menu. The NAME parameter is a string containing 1 through 31 characters. This parameter is required.

Remarks For more information, see the NOS/VE Object Code Management manual.

CREATE_MENU_ITEM CREAM Subcommand

Purpose Creates an item for an application menu. A menu represents a particular action to be performed by the application program, or a particular option for such an action. Up to 20 menu items can be defined for each menu class.

Format

CREATE_MENU_ITEM or
CREMI
KEY=keyword
SHIFT=boolean
CLASS=string
SHORT_LABEL=string
ALTERNATE_SHORT_LABEL=string
LONG_LABEL=string
ALTERNATE_LONG_LABEL=string
PAIR_WITH_PREVIOUS=boolean
STATUS=status variable

Parameters

KEY or *K*

Specifies the key on a terminal keyboard that is associated with the menu item. The name of the key and the associated SHIFT parameter must be unique within the menu. Selectable keys are f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16, next, help, stop, back, up, down, forward, backward, edit, data, insert_line, delete_line, home, clear, clear_eol_menu_item, delete_char_menu_item, insert_char_menu_item, and undo.

Omission of the KEY parameter causes no assignment of the menu item to a key. The menu item is automatically assigned to a key, however, when the menu is used.

SHIFT

Indicates whether the menu item is associated with a shifted key (YES) or an unshifted key (NO).

Omission of the SHIFT parameter assumes an unshifted key. If the KEY parameter is omitted, the SHIFT parameter is ignored.

CLASS or *C*

Specifies the menu class for this menu item. The CLASS parameter is a string containing 1 through 31 characters.

Omission of the CLASS parameter causes the most recently created menu class to be used. If no menu classes have been defined, an error results.

SHORT_LABEL or **SL**

Provides a short label to represent this menu item for the application user. The SHORT_LABEL parameter is a string containing 1 through 6 characters. this parameter is required.

ALTERNATE_SHORT_LABEL or *ASL*

Provides a short label when the meaning of the menu item is toggled. The ALTERNATE_SHORT_LABEL parameter is a string containing 1 through 6 characters.

Omission of the ALTERNATE_SHORT_LABEL parameter causes the value for the SHORT_LABEL parameter to be used; the menu item's meaning does not toggle.

LONG_LABEL or *LL*

Provides a long label to represent this menu item for the application user. The LONG_LABEL parameter is a string containing 1 through 31 characters.

Omission of the LONG_LABEL parameter causes the SHORT_LABEL parameter to be used.

ALTERNATE_LONG_LABEL or *ALL*

Provides a long label when the meaning of the menu item is toggled. The ALTERNATE_LONG_LABEL parameter is a string containing 1 through 31 characters.

Omission of the ALTERNATE_LONG_LABEL parameter causes the value for the LONG_LABEL parameter to be used; the menu item's meaning does not toggle.

PAIR_WITH_PREVIOUS or *PWP*

Indicates (YES) that this menu item is to be paired with the most recently created menu item during automatic assignment of menu items to keys.

Omission causes NO to be assumed; that is, there is no assignment preference in pairing this menu item with other menu items.

Remarks For more information, see the NOS/VE Object Code Management manual.

CREATE_MESSAGE_MODULE CREOL Subcommand

Purpose Starts the CREATE_MESSAGE_MODULE utility.

Format

CREATE_MESSAGE_MODULE or
CREMM
NAME=program_name
MANUAL=program_name
NATURAL_LANGUAGE=keyword or *name*
MERGE_OPTION=keyword
STATUS=status variable

Parameters

NAME or **N**.

Specifies the name of the message module to create. This parameter is required.

For status messages, you can specify any name.

For help and prompt messages, the name references the procedure or command for which the message module is being created. It must be in the form:

name$language

name is the name of the message module specified on either the parameter description table (PDT) or the procedure header. *language* is the natural language used to compose the messages in this module (it should be the same as the language that is specified by the NATURAL_LANGUAGE parameter).

Use a string value for a message module whose name is not an SCL name or a COBOL name.

MANUAL or *M*

Specifies the name of the online manual that describes the command or function for which this module contains help information.

NATURAL_LANGUAGE or *NL*

Specifies the name of the natural language used to compose the messages for the message module. Specify an SCL name or one of the following keywords:

DANISH	DUTCH
ENGLISH	FINNISH
FLEMISH	FRENCH

GERMAN	ITALIAN
NORWEGIAN	PORTUGUESE
SPANISH	SWEDISH
US_ENGLISH	

If NATURAL_LANGUAGE is omitted, US_ENGLISH is used.

MERGE_OPTION or *MO*

Specifies whether to add, replace, or combine the message module with the new object library. Options are:

ADD (A)

Message module is added to the new library.

REPLACE (R)

Message module replaces an existing module on the library.

COMBINE (C)

Message module is placed in the new library whether a module with the same name is present or not. If one is present, it is replaced with the new module.

If MERGE_OPTION is omitted, COMBINE is used.

Remarks

- CYBIL programmers can use GENERATE_MESSAGE_TEMPLATE to create message modules for status messages. This is described in the CYBIL System Interface manual.
- For more information, see the NOS/VE Object Code Management manual.

Examples

The following example creates a message module containing the status message: +P is not a command.

```
/create_object_library
COL/create_message_module name=a_message_module
CMM/create_status_message name=cle$unknown..
CMM../_command code=790 severity=error
? +P is not a command.
? **
CMM/end_message_module
COL/
```

CREATE_MODULE CREOL Subcommand

Purpose Creates a new load module from existing modules and adds it to the module list.

Format

CREATE_MODULE or
CREM
NAME=program_name
COMPONENT=list of record
GATE=keyword or *list of program_name*
RETAIN=keyword or *list of program_name*
STARTING_PROCEDURE=program_name
PRESET_VALUE=keyword
INCLUDE_BINARY_SECTION_MAPS=boolean
OUTPUT=file
APPLICATION_IDENTIFIER=name
STATUS=status variable

Parameters

NAME or N

Name of the new module.

You use a string value for a module whose name is not an SCL name or a COBOL name. An example of such a module name is the name of a C function, where lowercase is significant.

This parameter is required.

COMPONENT or COMPONENTS or C

Component modules of the new module. Each item in the list is a record consisting of a file name followed by a series of module names which are to be used. A range of names may be specified. If no module names are specified for a file, all modules on the file are used.

For modules with names which are not SCL or COBOL names, use a string value. An example is a C function name, in which lowercase is significant.

13

NOTE

A component module can only be a load or object module.

The component modules are combined within the new module in the order you list them on the COMPONENT parameter.

This parameter is required. At least one file must be specified.

GATE or *GATES* or *G*

Parameter Attributes: BY_NAME

List of additional entry points to be given the gate attribute in the new module.

You use a string value for an entry point whose name is not an SCL name or a COBOL name.

If GATE is omitted, the gated entry points in the new module are the entry points gated in the component modules.

RETAIN or *R*

Parameter Attributes: BY_NAME

List of additional entry points given the retain attribute.

You use a string value for an entry point whose name is not an SCL name or a COBOL name.

If RETAIN is omitted, the new module retains gated entry points, entry points assigned the retain attribute in the component modules, and entry points not referenced by any other component module.

STARTING_PROCEDURE or *SP*

Parameter Attributes: BY_NAME

Starting procedure for the new module.

You use a string value for an entry point whose name is not an SCL name or a COBOL name.

If STARTING_PROCEDURE is omitted, the starting procedure is the last transfer symbol in the last module specified in the COMPONENT parameter value list.

PRESET_VALUE or *PV*

Parameter Attributes: BY_NAME

Specifies text record reduction.

ZERO (Z)

Reduces the number of individual text records in an object module. Reducing the number of records reduces the module loading time.

If PRESET_VALUE is omitted, the number of text records is not reduced.

INCLUDE_BINARY_SECTION_MAPS or *IBSM*

Parameter Attributes: BY_NAME

Indicates whether the binary section map is included in the information element for the bound module.

If INCLUDE_BINARY_SECTION_MAPS is omitted, binary section maps are not included.

OUTPUT or *O*

Parameter Attributes: BY_NAME

File to which the section map for the new module is written. This file can be positioned.

If OUTPUT is omitted, no section map is written.

APPLICATION_IDENTIFIER or *AI*

Parameter Attributes: BY_NAME, ADVANCED

Name of the application identifier stored in the module header and included on the application accounting statistics when the software is executed.

Only a user with APPLICATION_ADMINISTRATION capability can specify an application identifier.

If an application identifier is placed on a load module, the module is assumed to be a unit-measured application.

If APPLICATION_IDENTIFIER is omitted, no application identifier is assigned to the module.

Remarks

- The new module is not generated until you enter a GENERATE_LIBRARY subcommand. Therefore, the section map for the module is not written on the file specified on the OUTPUT parameter until the module is generated.

- The existing modules to be combined are referred to as the component modules of the new module. The module type of the new module is a bound module because it is created by the combination of other modules.

- If a component module contains an external reference to another component module, CREOL links the modules.

- Although the command adds the new module to the module list, and stores information from the component module headers in the bound module header, it does not add the component modules to the module list. You can display component module information with the subcommand DISPLAY_NEW_LIBRARY or the SCL command DISPLAY_OBJECT_LIBRARY.

- The following entry points are kept in the bound module.

  - The starting procedure entry point for the bound module.

  - Entry points with the gate attribute. (The gate attribute indicates that a procedure executing in a ring within the call bracket of the module can call the entry point.)

  - Entry points with the retain attribute. (The retain attribute indicates the entry point is to be kept in a new module created by combining the module with other modules.)

  - Entry points not referenced by any other component module.

- You can assign the gate and retain attributes with the CREOL subcommands CREATE_MODULE or CHANGE_MODULE_ATTRIBUTES. You can also assign the gate attribute within the CYBIL source code (the #GATE attribute on the declaration).

- Do not bind COBOL programs that use CALL and CANCEL into a single module, because CALL will try to overlay a single module that is no longer available.

- For more information, see the NOS/VE Object Code Management manual.

Examples The following subcommand sequence creates a module, displays the module information, then generates a new object library. The new module is named NEW_MODULE and combines modules EXAMPLE and NAND on file OBJ1. When the new library is generated, it writes the section map on file $OUTPUT.

```
COL/create_module name=new_module component=..
COL../((obj1,example,nand)),output=$output
COL/display_new_library module=new_module ..
COL../display_options=(header,component)
 NEW_MODULE          - load module          - 18:05:32 1988-03-28
 kind: MI_VIRTUAL_STATE  generator: OBJECT_LIBRARY_GENERATOR
 generator name version: OBJECT LIBRARY GENERATOR V1.1
 components
 ~~~~~~~~~~

   component: EXAMPLE
   created:   18:05:32 1986-03-28
   generator: FORTRAN
   generator name version: FTN
   commentary: VS FORTRAN - level 86063

   component: NAND
   created:   18:05:32 1986-03-28
   generator: FORTRAN
   generator name version: FTN
   commentary: VS FORTRAN - level 86063

COL/generate_library library=my_new_library
 Section map for module  NEW_MODULE          created: 18:05:32  1988-03-28
 kind: CODE  length: 6E

   offset: 0          length: 54        module: EXAMPLE     section: EXAMPLE
   offset: 58         length: 16        module: NAND        section: NAND
 kind: BINDING  length: 50

 kind: WORKING STORAGE  length: 180

   offset: 0          length: 0EF       module: EXAMPLE
   offset: 0F0        length: 90        module: NAND
 kind: WORKING STORAGE  length: 88

   offset: 0          length: 78        module: EXAMPLE
   offset: 78         length: 10        module: NAND
```

CREATE_PARAMETER_ASSIST_MESSAGE CREMM Subcommand

Purpose Creates a help message that is displayed when a user enters an incorrect value for the parameter specified by this command.

13

Format **CREATE_PARAMETER_ASSIST_MESSAGE** or **CREPAM**

NAME=**name**
COLLECT_TEMPLATE_UNTIL=*string*
STATUS=*status variable*

Parameters **NAME** or **N**

Name of the command parameter for which the assist message is defined. If the command parameter has aliases, you must specify the first name listed in the PDT or PROCEDURE header. This parameter is required.

COLLECT_TEMPLATE_UNTIL or *CTU*

Specifies the termination string to use when collecting the template of the parameter assist message. If the COLLECT_TEMPLATE_UNTIL parameter is omitted, the string '**' is used.

Remarks For more information, see the NOS/VE Object Code Management manual.

Examples The following example creates a parameter assist message.

```
CMM/create_parameter_assist_message ..
CMM../name=display_options
? The DISPLAY_OPTIONS parameter must specify
? one of the following keyword values:
? BRIEF or FULL.
? **
CMM/
```

CREATE_PARAMETER_HELP_MESSAGE CREMM Subcommand

Purpose Creates a help message for a parameter. The help message is displayed when a user requests help for the parameter specified by this command.

Format **CREATE_PARAMETER_HELP_MESSAGE** or **CREPHM**

NAME=**name**
COLLECT_TEMPLATE_UNTIL=*string*
STATUS=*status variable*

Parameters **NAME or N**

Name of the command parameter for which the help message is defined. If the command parameter has aliases, you must specify the first name listed in the PDT or PROCEDURE header. This parameter is required.

COLLECT_TEMPLATE_UNTIL or *CTU*

Specifies the termination string to use when collecting the template of the parameter help message. If the COLLECT_TEMPLATE_UNTIL parameter is omitted, the string '**' is used.

Remarks For more information, see the NOS/VE Object Code Management manual.

Examples The following example creates a parameter help message.

```
CMM/create_parameter_help_message ..
CMM../name=display_options
? The DISPLAY_OPTIONS parameter determines the
? amount of information that is displayed for
? this file. Keyword values are BRIEF and FULL.
? **
CMM/
```

CREATE_PARAMETER_PROMPT_MESSAGE CREMM Subcommand

Purpose Creates the prompt message that appears for the parameter specified by this command.

Format **CREATE_PARAMETER_PROMPT_MESSAGE** or **CREPPM**

NAME=name
COLLECT_TEMPLATE_UNTIL=string
STATUS=status variable

Parameters **NAME or N**

Name of the command parameter for which the prompt message is defined. If the command parameter has aliases, you must specify the first name listed in the PDT or PROCEDURE header. This parameter is required.

COLLECT_TEMPLATE_UNTIL or *CTU*

Specifies the termination string to use when collecting the template of the parameter prompt message. If the COLLECT_TEMPLATE_UNTIL parameter is omitted, the string '**' is used.

Remarks

- NOS/VE allows you to name time zones, and to enter names for months and days in any natural language. Use the text of the parameter prompt message to define the names.
- Once a MONTHS_AND_DAYS module for a particular language has been referenced in a job, modifying the module on the object library or adding a different object library with a MONTHS_AND_DAYS module for the same language to your command list will have no effect on the current job.
- For more information, see the NOS/VE Object Code Management manual.

Examples

The following example creates a parameter prompt message.

```
CMM/create_parameter_prompt_message ..
CMM../name=display_options
? Display_options for the file?
? **
CMM/
```

The following example creates a parameter prompt template for time zones in US_ENGLISH.

```
/create_object_library
COL/create_message_module ..
COL../name=time_zones$us_english
CMM/create_parameter_prompt_message ..
CMM../name=standard_time$0
? Coordinated Universal Time
? **
CMM/create_parameter_prompt_message ..
CMM../name=standard_time$_6
? Central Standard Time
? **
CMM/create_parameter_prompt_message ..
CMM../name=daylight_saving_time$_6
```

```
? Central Daylight Saving Time, CDT
? **
CMM/end_message_module
COL/generate_library library=my_times
COL/quit
/
```

The following example extracts the released French month and day names.

```
/create_object_library
COL/add_module $system.osf$command_library ..
COL../module=months_and_days$french
COL/generate_library $local.months_days$..
COL../french format=message_module
COL/quit
```

The example DAYS_MONTHS_AND_TIME_ZONES in the online NOS/VE Examples manual demonstrates the definition of message modules for time zones, month names, and day names.

CREATE_PROGRAM_DESCRIPTION CREOL Subcommand

Purpose Defines a program description module and adds it to the module list.

Format **CREATE_PROGRAM_DESCRIPTION** or **CREPD**

NAME=record
FILE=list of: file or *string*
LIBRARY=list of: keyword or *file* or *string*
MODULE=list of program_name
STARTING_PROCEDURE=program_name
LOAD_MAP=file or *string*
LOAD_MAP_OPTION=keyword or *list of keyword*
TERMINATION_ERROR_LEVEL=keyword
PRESET_VALUE=keyword
STACK_SIZE=integer
ABORT_FILE=file or *string*
DEBUG_INPUT=file or *string*
DEBUG_OUTPUT=file or *string*
DEBUG_MODE=boolean
AVAILABILITY=keyword
SCOPE=keyword

LOG_OPTION=keyword
MERGE_OPTION=keyword
APPLICATION_IDENTIFIER=name
ARITHMETIC_OVERFLOW=boolean
ARITHMETIC_LOSS_OF_SIGNIFICANCE=boolean
DIVIDE_FAULT=boolean
EXPONENT_OVERFLOW=boolean
EXPONENT_UNDERFLOW=boolean
FP_INDEFINITE=boolean
FP_LOSS_OF_SIGNIFICANCE=boolean
INVALID_BDP_DATA=boolean
STATUS=status variable

Parameters

NAME or NAMES or N

The program name and its aliases. The first name is the module name. Subsequent names are aliases. Specify the names in a record with the following format:

```
record
  name: program_name
  aliases: list rest of program_name=$optional
recend
```

This parameter is required.

FILE or *FILES* or *F*

List of object files or object library files to be unconditionally loaded when the program is executed. A file value is evaluated when the library is generated.

Path values containing $FAMILY, $USER, or $SYSTEM elements, and file variable names can be supplied as strings to be evaluated when the program description is used.

LIBRARY or *LIBRARIES* or *L*

List of library files added to the program library list when the program is executed. A file value is evaluated when the object library is generated.

Path values containing $FAMILY, $USER, or $SYSTEM elements, and file variable names can be supplied as strings to be evaluated when the program description is used.

The keyword OSF$TASK_SERVICES_LIBRARY specifies the system table, and keyword OSF$CURRENT_LIBRARY represents the library containing the program description.

MODULE or *MODULES* or *M*

List of modules loaded from the program library list when the program is executed.

For modules with other than SCL or COBOL names, use a string value. An example is a C function name, in which lowercase is significant.

If MODULE is omitted, no additional modules are loaded when the program is executed.

STARTING_PROCEDURE or *SP*

Name of the entry point where execution begins.

For an entry point whose name is other than an SCL or COBOL name, use a string value. An example is a C function name, in which lowercase is significant.

If STARTING_PROCEDURE is omitted, the last transfer symbol loaded is used.

LOAD_MAP or *LM*

Parameter Attributes: BY_NAME

File on which the load map is written. A file value is evaluated when the library is generated.

Path values containing $FAMILY, $USER, or $SYSTEM elements can be supplied as strings to be evaluated when the program description is used.

This file can be positioned.

LOAD_MAP_OPTION or *LOAD_MAP_OPTIONS* or *LMO*

Parameter Attributes: BY_NAME

Set of one or more keywords indicating the information included in the load map. Options are:

ALL

Segment map, block map, entry point map, and entry point cross-reference.

NONE

No load map is written.

SEGMENT (S)
Segment map.

BLOCK (B)
Block map.

ENTRY_POINT (EP)
Entry point map.

CROSS_REFERENCE (CR)
Entry point cross-reference.

TERMINATION_ERROR_LEVEL or *TEL*
Parameter Attributes: BY_NAME
Error level that terminates program loading. Options are:

WARNING (W)
Warning, error, or fatal error.

ERROR (E)
Error or fatal error only.

FATAL (F)
Fatal error only.

PRESET_VALUE or *PV*
Parameter Attributes: BY_NAME
Value stored in all uninitialized words of program space. Options are:

ZERO (Z)
All zeros.

FLOATING_POINT_INDEFINITE (FPI)
Floating-point indefinite value.

INFINITY (I)
Floating-point infinite value.

ALTERNATE_ONES (AO)
Alternating 0 and 1 bits. The leftmost (highest order) bit is 1.

STACK_SIZE or *SS*

Parameter Attributes: BY_NAME

Maximum number of bytes in the run-time stack. The program uses the run-time stack for procedure call linkages and local variables.

ABORT_FILE or *AF*

Parameter Attributes: BY_NAME

File containing Debug commands to be processed if the program aborts. The commands are used only if the program is not executed in Debug mode. A file value is evaluated when the object library is generated.

Path values containing $FAMILY, $USER, or $SYSTEM elements can be supplied as strings to be evaluated when the program description is used. This file can be positioned.

DEBUG_INPUT or *DI*

Parameter Attributes: BY_NAME

File containing Debug commands. The commands are read only if the program is executed under control of Debug (refer to DEBUG_MODE parameter). This file can be positioned. A file value is evaluated when the object library is generated.

Path values containing $FAMILY, $USER, or $SYSTEM elements can be supplied as strings to be evaluated when the program description is used.

DEBUG_OUTPUT or *DO*

Parameter Attributes: BY_NAME

File on which Debug output is written. Output is written only if the program is executed in Debug mode. This file can be positioned. A file value is evaluated when the object library is generated.

Path values containing $FAMILY, $USER, or $SYSTEM elements can be supplied as strings to be evaluated when the program description is used.

DEBUG_MODE or *DM*

Parameter Attributes: BY_NAME

Indicates whether the program is to be run under the control of Debug. (For information on using Debug, refer to the program's specific source language manual). Options are:

ON

Program executed under control of the Debug program.

OFF

Program executed without the Debug program.

AVAILABILITY or *A*

Parameter Attributes: BY_NAME

Specifies whether the program description is made known to users as a command or not. Options are:

NORMAL_USAGE (ADVERTISED A NU)

The program description appears in the output produced by the DISPLAY_COMMAND_LIST_ENTRY command (and in other similar situations).

ADVANCED_USAGE (AU)

The command is included in displays of the user's command list but only if the user specifies the ADVANCED_USAGE display option for the DISPLAY_COMMAND_LIST_ENTRY command.

HIDDEN (H)

The program description is suppressed from the output produced by DISPLAY_COMMAND_LIST_ENTRY command (and in other similar situations).

Omission causes ADVERTISED to be used.

SCOPE or *S*

Parameter Attributes: BY_NAME

The manner in which the command processor may be called. The keyword options are:

XDCL (X)

The command is externally declared and may be called from outside the object library on which it resides.

GATE (G)

The program can be called from rings less privileged than the program's execution ring brackets. The GATE attribute implies the XDCL attribute.

LOCAL (L)

Reserved.

The default is XDCL.

LOG_OPTION or *LO*

Parameter Attributes: BY_NAME

Determines the manner in which calls to the program are logged. The keyword options are:

AUTOMATIC (A)

The logging is performed by the SCL Interpreter.

MANUAL (M)

Logging is performed by the program. Use this option to suppress the logging of secure information that should not be written to a log.

The default is AUTOMATIC.

MERGE_OPTION or *MO*

Parameter Attributes: BY_NAME

Indicates whether the program description module is added or replaced within the module list. Keyword options are:

ADD (A)

Added to the end of the module list.

REPLACE (R)

Replaces the module with the same name in the module list, if one exists.

COMBINE (C)

Added to the end of the module list if a module of the same name does not exist; replaces the module if it does exist.

If MERGE_OPTION is omitted, COMBINE is used.

APPLICATION_IDENTIFIER or *AI*

Parameter Attributes: BY_NAME, ADVANCED

Name of the application identifier stored in the module header and included on the application accounting statistics when the software is executed.

Only a user with APPLICATION_ADMINISTRATION capability can specify an application identifier.

If APPLICATION_IDENTIFIER is omitted, no application is identified with the program description.

ARITHMETIC_OVERFLOW or *AO*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition ARITHMETIC_OVERFLOW causes an interrupt. Valid specifications are:

ON

ARITHMETIC_OVERFLOW is enabled. The condition causes an interrupt.

OFF

ARITHMETIC_OVERFLOW is disabled. The condition does not cause an interrupt.

ARITHMETIC_LOSS_OF_SIGNIFICANCE or *ALOS*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition ARITHMETIC_LOSS_OF_SIGNIFICANCE causes an interrupt. Valid specifications are:

ON

ARITHMETIC_LOSS_OF_SIGNIFICANCE is enabled. The condition causes an interrupt.

OFF

ARITHMETIC_LOSS_OF_SIGNIFICANCE is disabled. The condition does not cause an interrupt.

DIVIDE_FAULT or *DF*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition DIVIDE_FAULT causes an interrupt. Valid specifications are:

ON

DIVIDE_FAULT is enabled. The condition causes an interrupt.

OFF

DIVIDE_FAULT is disabled. The condition does not cause an interrupt.

EXPONENT_OVERFLOW or *EO*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition EXPONENT_OVERFLOW causes an interrupt. Valid specifications are:

ON

EXPONENT_OVERFLOW is enabled. The condition causes an interrupt.

OFF

EXPONENT_OVERFLOW is disabled. The condition does not cause an interrupt.

EXPONENT_UNDERFLOW or *EU*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition EXPONENT_UNDERFLOW causes an interrupt. Valid specifications are:

ON

EXPONENT_UNDERFLOW is enabled. The condition causes an interrupt.

OFF

EXPONENT_UNDERFLOW is disabled. The condition does not cause an interrupt.

FP_INDEFINITE or *FPI* or *FI*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition FP_INDEFINITE causes an interrupt. Valid specifications are:

ON

FP_INDEFINITE is enabled. The condition causes an interrupt.

OFF

FP_INDEFINITE is disabled. The condition does not cause an interrupt.

FP_LOSS_OF_SIGNIFICANCE or *FPLOS* or *FLOS*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition FP_LOSS_OF_SIGNIFICANCE causes an interrupt. Valid specifications are:

ON

FP_LOSS_OF_SIGNIFICANCE is enabled. The condition causes an interrupt.

OFF

FP_LOSS_OF_SIGNIFICANCE is disabled. The condition does not cause an interrupt.

INVALID_BDP_DATA or *IBDPD* or *IBD*

Parameter Attributes: BY_NAME, ADVANCED

This parameter specifies whether or not the hardware condition INVALID_BDP_DATA causes an interrupt. Valid specifications are:

ON

INVALID_BDP_DATA is enabled. The condition causes an interrupt.

OFF

INVALID_BDP_DATA is disabled. The condition does not cause an interrupt.

Remarks

- You can execute the program described by the program description module with a command reference that specifies the module.
- Except where otherwise noted, omitting a parameter from the CREATE_PROGRAM_DESCRIPTION subcommand omits a corresponding attribute from the program description. This causes the corresponding job default program attribute value to be used when the program is executed. You can display the job default attributes by entering the DISPLAY_PROGRAM_ATTRIBUTES command.
- For more information, see the NOS/VE Object Code Management manual.

Examples

- The following subcommand creates a program description for a FORTRAN program.

```
COL/create_program_description ..
COL../name=fortran_program file=$local.lgo ..
COL../library='flf$library'
COL/generate_library library=.bjs.audit_library
COL/quit
/.bjs.audit_library.fortran_program
```

The command reference, .BJS.AUDIT_LIBRARY.FORTRAN_PROGRAM, loads all modules in file $LOCAL.LGO, and uses the file variable FLF$LIBRARY to add the object library FLF$LIBRARY to the program library list.

The value FLF$LIBRARY is specified as a string instead of a file reference or a variable name because this results in the string FLF$LIBRARY being evaluated each time the program description is executed.

- The following sequence demonstrates using OSF$CURRENT_LIBRARY in a program description. The NAME parameter on the program description defines SHOW_OFF and SHOO as two aliases for the

FORTRAN program P9939. The LIBRARY parameter specifies that the library on which this program description resides is to be added to the program library list when the program is executed.

Thus, if the object library is copied to another file, the program description does not have to be updated. The program description always specifies the library on which it resides.

```
/collect_text ftn_pgm
ct?       program p9939
ct?       print *,'In P9939'
ct?       end
ct?**
/fortran input=ftn_pgm
/create_object_library
COL/add_module library=lgo
COL/create_program_description name=..
COL../(show_off shoo) starting_procedure=p9939 ..
COL../library=osf$current_library
COL/generate_library library=my_lib
COL/quit
/my_lib.show_off
In P9939
/copy_file input=my_lib output=diff_lib
/diff_lib.show_off
In P9939
/
```

CREATE_STATUS_MESSAGE CREMM Subcommand

Purpose Creates a status message for the specified status condition code.

Format

CREATE_STATUS_MESSAGE or
CRESM
NAME=name
CODE=integer
IDENTIFIER=string
SEVERITY=keyword
COLLECT_TEMPLATE_UNTIL=string
STATUS=status variable

Parameters **NAME or N**

Specifies the condition identifier. This parameter is required.

CODE or C

Specifies the status condition code. If the CODE parameter is less than or equal to 16,777,215, the IDENTIFIER parameter must be specified and is combined with CODE to form the condition code. If CODE is greater than 16,777,215, it represents the complete status condition code. The IDENTIFIER parameter, if specified, is ignored.

Codes 0 through 9,999 for every possible product identifier are reserved for Control Data use. Codes 10,000 through 19,999 for every possible product identifier are for user-developed products. The remainder of each range is reserved for future use.

The CODE parameter is required.

IDENTIFIER or *I*

Specifies the two-character product identifier that is combined with CODE to form the status condition code.

SEVERITY or *S*

Specifies the severity level of the status condition. Options are:

NON_STANDARD (NS)

Non-standard condition. This flags non-standard extensions to the language specification.

DEPENDENT (D)

Dependent condition. This flags machine dependent usage in code. It is intended primarily for use by the implementation language (CYBIL), but other products with similar needs may also use it.

INFORMATIVE (I)

Information condition. These messages report conditions encountered during command processing that do not cause incorrect or incomplete operation of a command.

WARNING (W)

Warning condition. These messages report conditions encountered during command processing that may have caused incorrect or incomplete operation of a command or of subsequent commands.

ERROR (E)

Error condition. These messages report that the last command was not completed correctly. By default, a batch job is terminated. For an interactive session, additional input is requested from the user to direct continued job processing.

FATAL (F)

Fatal condition. These messages report that the last command or subcommand was not completed correctly. Subsequent processing is usually provided to discover additional problems.

CATASTROPHIC (C)

Catastrophic condition. These messages report that the last command or subcommand was not completed correctly. No further processing for the requested function is possible.

If SEVERITY is omitted, ERROR is used.

COLLECT_TEMPLATE_UNTIL or *CTU*

Specifies the termination string to use when collecting the template of the status message. If the COLLECT_TEMPLATE_UNTIL parameter is omitted, the string '**' is used.

Remarks For more information, see the NOS/VE Object Code Management manual.

Examples See the NOS/VE Object Code Management manual for a detailed example.

DELETE_MODULE CREOL Subcommand

Purpose Deletes one or more modules from the module list.

Format

DELETE_MODULE or
DELETE_MODULES or
DELM
MODULE=keyword or **list of program_name** or
list of range of program_name
STATUS=status variable

Parameters

MODULE or **MODULES** or **M**

Modules deleted. If ALL is specified, all modules in the module list are deleted. This parameter is required.

Remarks For more information, see the NOS/VE Object Code Management manual.

Examples The following session generates a new object library from a subset of the modules in an existing object library.

```
/create_object_library
COL/add_modules library=old_library
COL/delete_module (sort4,merge5)
COL/generate_library library=new_library
COL/quit
/
```

The object library generated on file NEW_LIBRARY contains all modules from file OLD_LIBRARY except modules SORT4 and MERGE5.

DISPLAY_NEW_LIBRARY CREOL Subcommand

Purpose Displays information about modules in the module list.

Format

DISPLAY_NEW_LIBRARY or
DISNL
MODULE=list of program_name or *list of range of program_name*
DISPLAY_OPTION=keyword or *list of keyword*
OUTPUT=file
ALPHABETICAL_ORDER=boolean
STATUS=status variable

Parameters *MODULE* or *MODULES* or *M*

List of modules for which information is displayed.

You use a string value for a module whose name is not an SCL name or a COBOL name. An example of such a module name is in the C language, where lowercase is significant.

If MODULE is omitted, information for all modules in the module list is displayed.

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Set of keywords indicating the information displayed in addition to the module type and name. Options are:

NONE

No information other than the module type and name.

DATE_TIME (DT)

Creation date and time.

ENTRY_POINT (EP)

Entry point names.

HEADER (H)

Module header information. This includes:

- Module type, name, creation date and time, kind, generator, generator name version, and commentary.
- Formal parameters, availability, scope, and log option for SCL command procedures.
- Entire program description, its availability, scope, log option, and application identifier for program description modules.
- Entire command description, its availability, scope, log option, and application identifier for command description modules.
- Entire function description, its availability, and scope for function description modules.

- Natural language for online manuals and message modules.
- The lowest and highest condition codes for message modules that contain status message information.

LIBRARIES or LIBRARY (L)

Local file names within the object text of the modules that are added to the program library list when the module is loaded (i.e., text-embedded libraries).

REFERENCE (R)

External references.

COMPONENT (C)

Module headers of the component modules if the module is a bound module.

ALL

All of the listed options.

If DISPLAY_OPTION is omitted, the default set by the last SET_DISPLAY_OPTIONS subcommand is used. The initial default is DATE_TIME.

OUTPUT or *O*

Output file. This file can be positioned. If OUTPUT is omitted, file $OUTPUT is used.

ALPHABETICAL_ORDER or *AO*

Indicates the display order for the module information. Options are:

TRUE

Alphabetical order by module name.

FALSE

Order in which modules exist on the library or file.

If ALPHABETICAL_ORDER is omitted, FALSE is used.

Remarks

- The DISPLAY_NEW_LIBRARY subcommand displays the contents of the new library that would be generated if the subcommand GENERATE_LIBRARY were entered.
- To change and display the default display options for subsequent DISPLAY_NEW_LIBRARY subcommands, enter a SET_DISPLAY_OPTION subcommand.
- For more information, see the NOS/VE Object Code Management manual.

Examples The following subcommand lists the module header and entry point information for module EXAMPLE.

```
COL/display_new_library example display_options=(h,ep)
 EXAMPLE                          - object module - 15:40:31 1986-04-09
 kind: MI_VIRTUAL_STATE  generator: CYBIL
 generator name version: C180 CYBIL/II 1.0 LEVEL 85302
 commentary: DA=NONE RC=NONE OPT=LOW
 entry points
 ~~~~~~~~~~~~

   EXAMPLE
 starting procedure: EXAMPLE
```

END_APPLICATION_MENU CREAM Subcommand

Purpose Terminates creation of the application and ends the CREATE_APPLICATION_MENU utility session.

Format **END_APPLICATION_MENU** or
QUIT or
ENDAM

Parameters None.

Remarks For more information, see the NOS/VE Object Code Management manual.

END_FORM_MODULE
CREFM Subcommand

Purpose END_FORM_MODULE ends the creation of a form and quits the CREATE_FORM_MODULE utility.

Format

END_FORM_MODULE or
QUI or
QUIT or
ENDFM
CREATE_MODULE=boolean
STATUS=status variable

Parameters *CREATE_MODULE* or *CM*

A boolean value indicating whether Screen Formatting is to create the form. The default is TRUE.

Remarks

- After quitting the CREATE_FORM_MODULE utility, you can write a form object definition and a form variable definition to files.
- For more information, see the NOS/VE Screen Formatting manual.

END_MESSAGE_MODULE
CREMM Subcommand

Purpose Ends the CREATE_MESSAGE_MODULE utility session.

Format

END_MESSAGE_MODULE or
QUI or
QUIT or
ENDMM
CREATE_MODULE=boolean
STATUS=status variable

Parameters *CREATE_MODULE* or *CM*

Specifies whether the message module should be created. If CREATE_MODULE is omitted, YES is used and the message module is created.

Remarks For more information, see the NOS/VE Object Code Management manual.

GENERATE_LIBRARY CREOL Subcommand

Purpose Generates a new object library using the information in the module list. This subcommand can also write an object file, SCL procedure text file, form source file, form variable file, or CREATE_MESSAGE_MODULE subcommands.

Format

GENERATE_LIBRARY or
GENL
LIBRARY=**file**
FORMAT=*keyword*
STATUS=*status variable*

Parameters

LIBRARY or **L**

File on which the modules are written. This parameter is required.

FORMAT or *F*

Specifies the format written. Options are:

LIBRARY (L)

Object library. Dictionaries are generated, and each object module in the module list is converted to the load module format. A module dictionary is written on the file.

FILE (F)

Object file. All modules in the module list must be object or load modules. All load modules are converted back to object modules. No dictionaries are generated.

FORM_SOURCE (FS)

Creates a file containing the CREATE_FORM_MODULE subcommands used to establish form modules in the module list. All form modules in the module list are written to the file.

FORM_VARIABLE (FV)

Creates a file containing the program variable definitions for every form module in the module list. The variable definitions are written in the language that processes the form.

SCL_PROCEDURE (SCL_PROC, SP)

SCL procedure text file. All command procedure modules in the module list are written to the file. This option allows command procedures to be edited on libraries.

MESSAGE_MODULE (MM)

Creates a file containing the CREOL subcommands for building message template modules. When the MESSAGE_MODULE parameter is specified, all message modules in the module list are written to the file. This option allows message module definitions to be edited on libraries.

If FORMAT is omitted, LIBRARY is used.

Remarks

- For a listing of the file contents, file processor, and file structure attributes created for the various kinds of object library files, see the CREATE_OBJECT_LIBRARY command.
- GENERATE_LIBRARY always discards the contents of the module list after it has used it.
- You can reference the library file written using GENL within the same CREOL session using subsequent CREOL subcommands.
- GENERATE_LIBRARY requires append and shorten access to write an object library file. If this access cannot be obtained, the file is written to a uniquely named file, and the subcommand reports the file name.

  When changing an object library that is already an entry in the command list, keep in mind that all object libraries in the command list are considered open and can not be accessed for any kind of write operation. Because GENL will not be able to write to the library in the command list, you should create a higher cycle of the same library and add that library to the command list. An example which shows how to do this is included in the examples section that follows.
- For more information about form modules, see the NOS/VE Screen Formatting manual.

- For more information, see the NOS/VE Object Code Management manual.

Examples

- The following sequence generates an object library that contains the modules from object files OBJ1 and OBJ2.

```
/create_object_library
COL/add_module (obj1,obj2)
COL/generate_library $user.library_1
COL/quit
```

- The following sequence extracts the text in a command procedure stored in the object library on file $USER.MY_PROCED. The SCL command COPY_FILE lists the contents of the text file.

```
/create_object_library
COL/add_module library=$user.my_proced ..
COL../module=proc1
COL/generate_library library=text_file ..
COL../format=scl_proc
COL/copy_file input=text_file
PROCEDURE proc1
    attach_file $system.library
    detach_file $system.library2
PROCEND proc1
COL/quit
/
```

- The following sequence demonstrates how to update an object library that is in a command list. It makes the object library $USER.MY_PROCED.1 an entry in the command list, extracts a procedure from the object library, edits the procedure, puts the edited procedure on the new object library $USER.MY_PROCED.2, removes the command list entry for $USER.MY_PROCED.1, and adds the command list entry for $USER.MY_PROCED.2.

```
/create_object_library
COL/add_module library=$user.my_proced.1 ..
COL../module=proc1
COL/generate_library library=proc1_source ..
COL../format=scl_proc
COL/edit_file file=proc1_source
   .
   .   "Use EDIT_FILE to make changes.
   .
COL/add_module library=$user.my_proced.1
COL/replace_module library=proc1_source
COL/generate_library library=$user.my_proced.2
COL/quit
/delete_command_list_entry ..
../entry=$user.my_proced.1
/create_command_list_entry ..
../entry=$user.my_proced.2
```

QUIT
CREOL Subcommand

Purpose Ends a CREATE_OBJECT_LIBRARY utility session.

Format **QUIT** or
QUI

Parameters None.

Remarks For more information, see the NOS/VE Object Code Management manual.

REORDER_MODULE
CREOL Subcommand

Purpose Changes the order of one or more modules in the module list.

Format

REORDER_MODULE or
REORDER_MODULES or
REOM
MODULE=list of program_name or list of range of program_name
PLACEMENT=keyword
DESTINATION=program_name
STATUS=status variable

Parameters

MODULE or **MODULES** or **M**

List of modules in the order the modules are to appear in the module list.

You use a string value for a module whose name is not an SCL name or a COBOL name.

This parameter is required.

PLACEMENT or *P*

Indicates whether the ordered modules are placed before or after the module specified on the DESTINATION parameter. Options are:

BEFORE (B)

Modules placed before the destination module.

AFTER (A)

Modules placed after the destination module.

If PLACEMENT is omitted, AFTER is used.

DESTINATION or *D*

Module before or after which the ordered modules are placed.

If DESTINATION is omitted, the location depends on the PLACEMENT parameter value. If PLACEMENT=BEFORE is specified, the modules are placed at the beginning of the module list; if PLACEMENT=AFTER is specified, the modules are placed at the end of the module list.

Remarks

- To reorder modules, list the modules on the MODULE parameter of the subcommand in the order the modules are to appear in the module list. Then specify the location where CREOL is to insert the reordered modules into the module list, using the DESTINATION and PLACEMENT parameters.

  - For more information, see the NOS/VE Object Code Management manual.

Examples

The following subcommand reorders the modules START, MIDDLE, and END, and places them at the end of the module list.

```
COL/reorder_module modules=(start,middle,end)
```

REPLACE_MODULE CREOL Subcommand

Purpose

Replaces one or more modules in the module list.

Format

REPLACE_MODULE or
REPLACE_MODULES or
REPM
LIBRARY=list of file
MODULE=list of program_name or *list of range of program_name*
STATUS=status variable

Parameters

LIBRARY or **LIBRARIES** or **L**

Object files, SCL procedure files, or object library files containing the replacement modules. This parameter is required.

MODULE or *MODULES* or *M*

Replacement modules.

You use a string value for a module whose name is not an SCL name or a COBOL name.

If MODULE is omitted, all modules contained in the files specified on the library parameter are used.

Remarks

- The REPLACE_MODULES subcommand can specify object files, SCL procedure files, or object library files. The files are replaced in the order you specify them on the LIBRARY parameter. If you do not want to use all modules in the files, specify the modules to be used on the MODULE parameter.

- If the name of a specified module matches a name in the module list, the specified module replaces the existing module. If no module exists with the same name, a warning status is returned and the module is not added to the module list.

- A replacement module is always placed in the module list at the same location as the module it replaces.

- The REPLACE_MODULES subcommand does not add modules to the module list. To add modules, enter an ADD_MODULES subcommand. To both add and replace modules, enter a COMBINE_MODULES subcommand.

- If you specify an SCL procedure whose header references a non-standard type, you must make the type definition available. For instance, if you want to add the following procedure:

```
PROCEDURE show(
   p1: address_list = $required
   status)
  :
PROCEND show
```

  then the type definition of ADDRESS_LIST must be created outside the procedure. This is accomplished by using the TYPE control statement, as in:

```
TYPE
   address_list : list 1..3 of string
TYPEND
```

- For more information, see the NOS/VE Object Code Management manual.

Examples The following subcommand uses all modules on file BINARY to replace modules in the current module list.

```
COL/replace_module library=binary
```

SATISFY_EXTERNAL_REFERENCE CREOL Subcommand

Purpose Adds modules to the module list that satisfy external references.

Format

SATISFY_EXTERNAL_REFERENCE or
SATISFY_EXTERNAL_REFERENCES or
SATER
LIBRARY=list of file
STATUS=status variable

Parameters

LIBRARY or **LIBRARIES** or **L**

Object library files that are searched for modules containing referenced entry points. The libraries are searched in the order specified on the parameter. This parameter is required.

Remarks

- You should enter the SATISFY_EXTERNAL_REFERENCES subcommand after you have entered the ADD_MODULE, REPLACE_MODULE, COMBINE_MODULE, and CREATE_MODULE subcommands that specify the initial module list, so that a single SATISFY_EXTERNAL_REFERENCES subcommand is effective for the entire new object library.

- If none of the procedures in the new object library request that entry points be loaded dynamically, the SATISFY_EXTERNAL_REFERENCES subcommand ensures that the object library files specified on the subcommand need not be specified in the program library list when a module from the new object library is loaded because all modules required from these libraries are part of the new object library.

  For example, if MODA in the module list references FTNMOD1 and FTNMOD2 from file FTNLIB, a SATISFY_EXTERNAL_REFERENCES subcommand that specifies FTNLIB adds FTNMOD1 and FTNMOD2 to the module list. Later, after the new object library

is generated, a subcommand to execute MODA need not specify FTNLIB in the program library list. The loader can load modules FTNMOD1 and FTNMOD2 from the same file as MODA.

- To process a SATISFY_EXTERNAL_REFERENCES subcommand, the CREOL utility generates an external reference list and an entry point list for all modules currently in the module list. It then attempts to match each external reference to an entry point. If the entry point to satisfy an external reference is not in the entry point list, CREOL searches the files specified on the subcommand for a module containing the entry point. The files are searched in the order listed on the subcommand.

  If, after searching each specified file, the CREOL utility does not find the entry point, it continues with the next external reference in the list. No abnormal status is returned if an external reference is not matched.

  If the entry point is found, the module is added to the end of the module list. When a module is added to the module list, the entry points and external references within the module are also added to the entry point list and external reference list, respectively. Because the external references of the added modules are added to the external reference list, the SATISFY_EXTERNAL_REFERENCES subcommand also attempts to match the added external references.

  The process of matching external references continues until reaching the end of the external reference list, when the entry point and external reference lists are discarded.

- The NOS/VE task service library (OSF$TASK_SERVICES_LIBRARY) should not be used on the SATISFY_EXTERNAL_REFERENCES subcommand. If it is, an error status is returned. There is no way to bind the system entry points into a module such that external references to program interfaces FSP$OPEN_FILE, PMP$EXIT, or similar system routines, can be eliminated from the loading process.

- For more information, see the NOS/VE Object Code Management manual.

Examples The following sequence compiles a FORTRAN source program and then generates an object library. The object library contains the modules from file MY_LGO, and the modules referenced from the FORTRAN run-time and math libraries on files referenced by variables FLF$LIBRARY and MLF$LIBRARY. File variables which reference the run-time libraries for languages supported by Control Data are automatically created in a job at job initiation. No user intervention is required to access these libraries.

```
/fortran input=source binary=my_lgo
/create_object_library
COL/add_module library=my_lgo
COL/satisfy_external_references ..
COL../libraries=(flf$library,mlf$library)
COL/generate_library library=$user.my_library
COL/quit
/
```

SET_CHARACTER_INPUT CREFM Subcommand

Purpose SET_CHARACTER_INPUT specifies the user entry format and valid values for a character variable.

Format

SET_CHARACTER_INPUT or
SETCI
VARIABLE_NAME=name or **cobol_name**
VALID_VALUE=list of string
COMPARE_TO_SUBSTRING=boolean
ENTRY_FORMAT=keyword
STATUS=status variable

Parameters

VARIABLE_NAME or **VN**

The name of the variable. This parameter is required.

VALID_VALUE or *VALID_VALUES* or *VV*

A list of character strings valid for the user to enter. You can specify a string of from 0 through 65,535 characters. By default, all strings are valid.

COMPARE_TO_SUBSTRING or *CTS*

A boolean value specifying whether the user can enter a unique substring of one of the valid values. The default is TRUE.

ENTRY_FORMAT or *EF*

The data entry format for the user. The following values are valid:

Value	Meaning
CHARACTER	Allows any ASCII characters.
ALPHABETIC	Allows only alphabetic characters (a through z and A through Z).
DIGITS	Allows only unsigned numeric characters (0 through 9).
SIGNED	Allows numeric characters with or without a leading sign.

The default is CHARACTER.

Remarks

- You need to use this subcommand only to specify values other than the defaults.
- You can use this subcommand only when the value specified for the DATA_TYPE parameter of the ADD_VARIABLE subcommand is CHARACTER or COBOL.
- For more information, see the NOS/VE Screen Formatting manual.

SET_COBOL_DATA CREFM Subcommand

Purpose

SET_COBOL_DATA specifies the program format for a COBOL variable.

Format

SET_COBOL_DATA or
SETCD
VARIABLE_NAME=name or **cobol_name**
USAGE=keyword
PICTURE=string
STATUS=status variable

Parameters **VARIABLE_NAME or VN**

The name of the variable. This parameter is required.

USAGE or *U*

The COBOL USAGE clause. The application program cannot change the variable size set by this clause and the PICTURE clause.

The following values are valid:

BINARY
COMPUTATIONAL
COMP
COMPUTATIONAL-1
COMP-1
COMPUTATIONAL-3
COMP-3
DISPLAY
PACKED-DECIMAL

The default is DISPLAY.

PICTURE or *P*

The symbols that represent the picture. The application program cannot change the variable size set by this clause and the USAGE clause. You can specify a string of from 0 through 30 characters.

The default string is:

PIC X(n)

(n is the size of the variable text object.)

When you specify the string, the following rules apply:

- You cannot specify an edited numeric item or edited alphanumeric item.
- For an alphabetic item, you cannot specify the editing symbols *B* and *P*.
- You cannot specify the following combinations of items in the PICTURE clause for this subcommand and for the SET_COBOL_OUTPUT subcommand:
  - A numeric item for SET_COBOL_OUTPUT and an alphabetic or alphanumeric item for SET_COBOL_DATA.

- An alphabetic item or alphanumeric item for SET_COBOL_OUTPUT and a numeric item or signed numeric item for SET_COBOL_DATA.

Examples:

'999V99'
'X(13)'

Remarks

- You can use this subcommand only when the value of the PROCESSOR parameter of the SET_FORM subcommand is COBOL (COBOL is the default).
- Specifying this subcommand automatically sets the DATA_TYPE parameter of the ADD_VARIABLE subcommand to COBOL.
- You need to use this subcommand only to specify values other than the defaults.
- When the value specified for the DATA_TYPE parameter of the ADD_VARIABLE subcommand is CHARACTER, INTEGER, REAL, or UPPERCASE, Screen Formatting ignores values specified on this subcommand. The appropriate subcommands for these data types are the following:

  SET_CHARACTER_INPUT
  SET_INTEGER_INPUT
  SET_MONEY_INPUT
  SET_REAL_INPUT

- For more information, see the NOS/VE Screen Formatting manual.

SET_COBOL_OUTPUT CREFM Subcommand

Purpose SET_COBOL_OUTPUT specifies the display format for a COBOL variable.

Format

SET_COBOL_OUTPUT or
SETCO
VARIABLE_NAME=name or **cobol_name**
PICTURE=string
STATUS=status variable

Parameters

VARIABLE_NAME or VN

The name of the variable. This parameter is required.

PICTURE or *P*

The symbols that represent the picture. The number of characters required must be equal to or less than the variable text object. You can specify a string of from 0 through 30 characters.

The default string is:

PIC X(n)

(n is the size of the variable text object.)

When you specify the string, the following rules apply:

- You cannot specify the following items:
  - An edited alphanumeric item.
  - A signed numeric item.
- For an alphabetic item, you cannot use the editing symbols *B* and *P*.
- You cannot specify the following combinations of PICTURE clauses on this subcommand and the SET_COBOL_DATA subcommand:
  - A numeric item for SET_COBOL_OUTPUT and an alphabetic or alphanumeric item for SET_COBOL_DATA.
  - An alphabetic item or alphanumeric item for SET_COBOL_OUTPUT and a numeric item or signed numeric item for SET_COBOL_DATA.

Examples:

'$999.99'
'X(13)'

Remarks

- You can use this subcommand only when the value of the PROCESSOR parameter of the SET_FORM subcommand is COBOL (COBOL is the default).
- Specifying this subcommand automatically sets the DATA_TYPE parameter of the ADD_VARIABLE subcommand to COBOL.

- You need to use this subcommand only to specify values other than the default.
- When the value specified for the DATA_TYPE parameter of the ADD_VARIABLE subcommand is CHARACTER, INTEGER, REAL, or UPPERCASE, Screen Formatting ignores values specified on this subcommand. The appropriate subcommands for these data types are the following:

  SET_EXPONENT_OUTPUT
  SET_FLOAT_OUTPUT
  SET_INTEGER_OUTPUT
  SET_MONEY_OUTPUT

- For more information, see the NOS/VE Screen Formatting manual.

SET_DISPLAY_OPTION CREOL Subcommand

Purpose Changes and displays the default display options for subsequent DISPLAY_NEW_LIBRARY subcommands within the CREATE_OBJECT_LIBRARY session.

Format

SET_DISPLAY_OPTION or
SET_DISPLAY_OPTIONS or
SETDO
DISPLAY_OPTION=keyword or *list of keyword*
STATUS=status variable

Parameters *DISPLAY_OPTION* or *DISPLAY_OPTIONS* or *DO*

List of one or more keywords indicating the new default display options. The keywords indicate the information displayed in addition to the module type and name. Options are:

NONE

No information other than the module type and name.

DATE_TIME (DT)

Creation date and time.

ENTRY_POINT (EP)

Entry point names.

HEADER (H)

Module header information. This includes the following:

- Module type, name, creation date and time, kind, generator, generator name version, and commentary.
- Formal parameters, availability, scope, and log option for SCL command procedures.
- Entire command description, its availability, scope, log option, and application identifier for command description modules.
- Entire function description, its availability and scope for function description modules.
- Entire program description, its availability, scope, log option, and application identifier for program description modules.
- Natural language for online manuals and message modules.
- The lowest and highest condition codes for message modules that contain status message information.

LIBRARIES (L)

Local file names within the object text of the modules that are added to the program library list when the module is loaded (for example, text-embedded libraries).

REFERENCE (R)

External references.

COMPONENT (C)

Module headers of the component modules if the module is a bound module.

ALL

All of the listed options.

If DISPLAY_OPTION is omitted, the default display options are displayed without change.

13

Remarks

- The initial default display option is DATE_TIME.
- For more information, see the NOS/VE Object Code Management manual.

Examples The following subcommand changes and displays the default display option.

```
COL/set_display_options display_options=..
COL../(date_time,header,entry_point)
-- display option = (DATE_TIME,HEADER,ENTRY_POINT)
COL/
```

SET_EXPONENT_OUTPUT CREFM Subcommand

Purpose SET_EXPONENT_OUTPUT specifies the display format of the exponent for a real number variable.

Format

SET_EXPONENT_OUTPUT or
SETEO
VARIABLE_NAME=name or **cobol_name**
FORMAT=keyword
WIDTH=integer
DIGITS_RIGHT_OF_DECIMAL=integer
DIGITS_IN_EXPONENT=integer
SIGN=keyword
SUPPRESS_ZERO=boolean
STATUS=status variable

Parameters

VARIABLE_NAME or **VN**

The name of the real number variable. This parameter is required.

FORMAT or **F**

The FORTRAN format descriptor for displaying the real number. The valid values are EE and GE. This parameter is required.

WIDTH or **W**

The number of columns the variable text object for displaying the real number occupies. The valid values are from 1 through 19. This parameter is required.

DIGITS_RIGHT_OF_DECIMAL or DROD

The number of digits to display to the right of the decimal marker. The valid values are from 0 through 19. This parameter is required.

DIGITS_IN_EXPONENT or DIE

The number of digits in the exponent. The valid values are from 0 through 19. This parameter is required.

SIGN or *S*

The format of the sign for the real number. The following values are valid:

Value	Meaning
MINUS_IF_NEGATIVE	Displays a sign when the real number is negative.
ALWAYS_SIGNED	Displays signs for both positive and negative real numbers.

The default is MINUS_IF_NEGATIVE.

SUPPRESS_ZERO or *SZ*

Boolean value specifying how to display zero. TRUE displays a zero value as spaces; FALSE displays the zeros.

The default is TRUE.

Remarks

- When the value specified for the DATA_TYPE parameter of the ADD_VARIABLE subcommand is REAL, Screen Formatting accepts values specified on this subcommand.
- For more information, see the NOS/VE Screen Formatting manual.

SET_FLOAT_OUTPUT CREFM Subcommand

Purpose SET_FLOAT_OUTPUT specifies the floating format for the display of a real number variable.

Format

SET_FLOAT_OUTPUT or
SETFO
VARIABLE_NAME=name or **cobol_name**
FORMAT=keyword
WIDTH=integer
DIGITS_RIGHT_OF_DECIMAL=integer
SIGN=keyword
SUPPRESS_ZERO=boolean
STATUS=status variable

Parameters

VARIABLE_NAME or **VN**

The name of the real number variable. This parameter is required.

FORMAT or **F**

The FORTRAN format descriptor for displaying the real number. The valid values are F, G, and E. This parameter is required.

WIDTH or **W**

The number of columns the variable text object for displaying the real number occupies. The valid values are from 1 through 19. This parameter is required.

DIGITS_RIGHT_OF_DECIMAL or **DROD**

The number of digits to display to the right of the decimal marker. The valid values are from 0 through 19. This parameter is required.

SIGN or *S*

The format of the sign for the real number. The following values are valid:

Value	Meaning
MINUS_IF_NEGATIVE	Displays a sign when the real number is negative.
ALWAYS_SIGNED	Displays signs for both positive and negative real numbers.

The default is MINUS_IF_NEGATIVE.

SUPPRESS_ZERO or *SZ*

A boolean value specifying how to display zero. TRUE displays a zero value as spaces; FALSE displays the zeros. The default is TRUE.

Remarks

- When the value specified for the DATA_TYPE parameter of the ADD_VARIABLE subcommand is REAL, Screen Formatting accepts values specified on this subcommand.
- For more information, see the NOS/VE Screen Formatting manual.

SET_FORM CREFM Subcommand

Purpose

SET_FORM sets attributes that apply to the entire form. When you enter the CREATE_FORM_MODULE subcommand, Screen Formatting establishes a set of default attributes for the form. These defaults are listed with the parameter descriptions of the SET_FORM subcommand. By specifying parameters on this subcommand, you establish new attributes for the form.

Format

SET_FORM or
SETF
PROCESSOR=keyword
COLUMN=integer
LINE=integer
WIDTH=integer
HEIGHT=integer

DISPLAY=list of keyword
COMMENT=list of string
EVENT_FORM=keyword or *name*
HELP_PROCESSING=keyword or *name* or *string*
MESSAGE_FORM=name
VARIABLE_DECK_NAME=name
VARIABLE_RECORD_NAME=name
INVALID_DATA_CHARACTER=keyword or *string*
STATUS=status variable

Parameters *PROCESSOR* or *P*

The name of the application programming language that interacts with the user through the form. The following values are valid:

ANSI_FORTRAN
CDC_FORTRAN
COBOL
CYBIL
PASCAL
SCL

The default is COBOL.

If you specify COBOL as the processor, Screen Formatting converts underscores to hyphens for all names used by the application program.

COLUMN or *C*

The column position for the upper left corner of the form. Column 1 is the upper left corner of the screen. The valid values are from 1 through 256. The default is 1.

LINE or *L*

The line position for the upper left corner of the form. Line 1 is the upper left corner of the screen. The valid values are from 1 through 256. The default is 1.

WIDTH or *W*

The number of columns the form occupies. The valid values are from 1 through 256. If you specify this parameter, you must also specify the HEIGHT parameter. By default, the form occupies the entire screen.

HEIGHT or *H*

The number of lines the form occupies. The valid values are from 1 through 256. If you specify this parameter you must also specify the WIDTH parameter. By default, the form occupies the entire screen.

DISPLAY or *DISPLAYS* or *D*

A list of display attributes for areas of the form that have no objects and for objects that have no attributes specified. The following values are valid:

INVERSE
LOW_INTENSITY
HIGH_INTENSITY
BLINK
BLACK_BACKGROUND
BLUE_BACKGROUND
GREEN_BACKGROUND
MAGENTA_BACKGROUND
RED_BACKGROUND
CYAN_BACKGROUND
YELLOW_BACKGROUND
WHITE_BACKGROUND
BLACK_FOREGROUND
BLUE_FOREGROUND
GREEN_FOREGROUND
MAGENTA_FOREGROUND
RED_FOREGROUND
CYAN_FOREGROUND
YELLOW_FOREGROUND
WHITE_FOREGROUND
FINE_BORDER
MEDIUM_BORDER
BOLD_BORDER
DISPLAY_LEFT_TO_RIGHT
DISPLAY_RIGHT_TO_LEFT

The defaults are BLACK_BACKGROUND and WHITE_FOREGROUND.

COMMENT or *COMMENTS*

Parameter Attributes: BY_NAME

Comments to be saved with the form definition. By default, no comments are saved.

EVENT_FORM or *EF*

Parameter Attributes: BY_NAME

An associated event form to display the mapping of program events to terminal keys. The following values are valid:

Value	Meaning
name	Displays a form with the specified name.
SYSTEM	Displays the system event form (generated by Screen Formatting).
NONE	Displays no event form.

The default is SYSTEM.

HELP_PROCESSING or *HP*

Parameter Attributes: BY_NAME

Help processing for the form. Help processing occurs when the user executes a help event on an area that contains no object. The following values are valid:

Value	Meaning
name	Displays a form with the specified name.
string (0 through 256 characters)	Displays the specified string.
SYSTEM	Displays the system help form generated by Screen Formatting.
NONE	Displays no help information.

The default is SYSTEM.

MESSAGE_FORM or *MF*

Parameter Attributes: BY_NAME

The name of the form you designed for help error messages. This form must be in an object library in the user's command list. The default is a form created by Screen Formatting. (For a description of the form, see Using the Default Form for Error and Help Information earlier in this chapter.)

VARIABLE_DECK_NAME or *VDN*

Parameter Attributes: BY_NAME

The SOURCE_CODE_UTILITY deck name for the form variable. The default is the name of the form.

VARIABLE_RECORD_NAME or *VRN*

Parameter Attributes: BY_NAME

The record name for the form variable. The default is the value specified for the VARIABLE_DECK_NAME parameter.

INVALID_DATA_CHARACTER or *IDC*

Parameter Attributes: BY_NAME

The character to display on the form when the program is attempting to display invalid data. The valid value is either a single-character string or the keyword NONE. The default is NONE.

Remarks

- If the user's terminal screen is not large enough to display the entire form (as specified with the COLUMN, LINE, WIDTH, and HEIGHT parameters), the form is not displayed.

For more information, see the NOS/VE Screen Formatting manual.

SET_INTEGER_INPUT CREFM Subcommand

Purpose SET_INTEGER_INPUT specifies the user entry format and valid values for an integer variable.

Format

SET_INTEGER_INPUT or
SETII
VARIABLE_NAME=name or **cobol_name**
VALID_VALUE=list of range of integer
ENTRY_FORMAT=keyword
STATUS=status variable

Parameters

VARIABLE_NAME or VN

The name of the variable. This parameter is required.

VALID_VALUE or *VALID_VALUES* or *VV*

A list of integer ranges valid for the user to enter. The valid values are from 0 through 10000. By default, all integers are valid.

ENTRY_FORMAT or *EF*

The data entry format for the user. The following values are valid:

Value	Meaning
DIGITS	Allows only unsigned numeric characters (0 through 9).
SIGNED	Allows numeric characters with or without leading signs.

The default is SIGNED.

Remarks

- You need to use this subcommand only to specify values other than the defaults.
- When the value specified for the DATA_TYPE parameter of the ADD_VARIABLE subcommand is INTEGER or COBOL, Screen Formatting accepts values specified on this subcommand.
- For more information, see the NOS/VE Screen Formatting manual.

SET_INTEGER_OUTPUT CREFM Subcommand

Purpose

SET_INTEGER_OUTPUT specifies the display format for an integer variable.

Format

SET_INTEGER_OUTPUT or
SETIO
VARIABLE_NAME=name or **cobol_name**
WIDTH=integer
MINIMUM_DIGIT=integer
SIGN=keyword
STATUS=status variable

Parameters

VARIABLE_NAME or VN

The name of the variable. This parameter is required.

WIDTH or W

The number of columns the variable text object for displaying the integer occupies. The valid values are from 1 through 19. This parameter is required.

MINIMUM_DIGIT or *MINIMUM_DIGITS* or *MD*

The minimum number of digits to display. The valid values are from 0 through 19. The default is 0.

SIGN or *S*

The format of the sign for the integer. The following values are valid:

Value	Meaning
MINUS_IF_NEGATIVE	Displays a sign when the integer is negative.
ALWAYS_SIGNED	Displays signs for both positive and negative integers.

The default is MINUS_IF_NEGATIVE

Remarks

- When the value specified for the DATA_TYPE parameter of the ADD_VARIABLE subcommand is INTEGER, Screen Formatting accepts values specified on this subcommand.
- For more information, see the NOS/VE Screen Formatting manual.

SET_MONEY_INPUT CREFM Subcommand

Purpose

SET_MONEY_INPUT specifies the user entry format for money variables.

Format

SET_MONEY_INPUT or
SETMI
VARIABLE_NAME=**name** or **cobol_name**
MONEY_SYMBOL=*string*
THOUSANDS_SEPARATOR=*string*
DECIMAL_POINT=*string*
STATUS=*status variable*

Parameters

VARIABLE_NAME or **VN**

The name of the variable. This parameter is required.

MONEY_SYMBOL or *MS*

A character to use as the money symbol. Any single ASCII character is valid. The default is the dollar sign ($).

THOUSANDS_SEPARATOR or *TS*

A character to separate thousands in the currency. Any single ASCII character is valid. The default is the comma (,).

DECIMAL_POINT or *DP*

A character to represent the decimal marker. Any single ASCII character is valid. The default is the period (.).

Remarks

- You need to use this subcommand only to specify values other than the defaults.
- When the value specified for the DATA_TYPE parameter of the ADD_VARIABLE subcommand is INTEGER or REAL, Screen Formatting accepts values specified on this subcommand.
- For more information, see the NOS/VE Screen Formatting manual.

SET_MONEY_OUTPUT CREFM Subcommand

Purpose SET_MONEY_OUTPUT specifies the display format for money variables.

Format

SET_MONEY_OUTPUT or
SETMO
VARIABLE_NAME=name or **cobol_name**
MONEY_SYMBOL=string
THOUSANDS_SEPARATOR=string
DECIMAL_POINT=string
SIGN=keyword
SUPPRESS_ZERO=boolean
STATUS=status variable

Parameters

VARIABLE_NAME or **VN**

The name of the variable. This parameter is required.

MONEY_SYMBOL or *MS*

A character to use as the money symbol. Any single ASCII character is valid. The default is the dollar sign ($).

THOUSANDS_SEPARATOR or *TS*

A character to separate thousands in the currency. Any single ASCII character is valid. The default is the comma (,).

DECIMAL_POINT or *DP*

A character to represent the decimal marker. Any single ASCII character is valid. The default is the period (.).

SIGN or *S*

The format of the sign for the integer. The following values are valid:

Value	Meaning
MINUS_IF_NEGATIVE	Displays a sign when the integer is negative.
ALWAYS_SIGNED	Displays signs for both positive and negative integers.

The default is MINUS_IF_NEGATIVE.

SUPPRESS_ZERO or *SZ*

A boolean value specifying how to display zero. If TRUE, displays a zero value as spaces. If FALSE, displays the zeros. The default is TRUE.

Remarks

- You need to use this subcommand only to specify values other than the defaults.
- When the value specified for the DATA_TYPE parameter of the ADD_VARIABLE subcommand is INTEGER or REAL, Screen Formatting accepts values specified on this subcommand.
- For more information, see the NOS/VE Screen Formatting manual.

SET_REAL_INPUT CREFM Subcommand

Purpose SET_REAL_INPUT specifies the values the user can enter for a real variable.

Format

SET_REAL_INPUT or
SETRI
VARIABLE_NAME=name or **cobol_name**
VALID_VALUE=list of range of real
STATUS=status variable

Parameters

VARIABLE_NAME or **VN**

The name of the real variable. This parameter is required.

VALID_VALUE or *VALID_VALUES* or *VV*

A list of real ranges valid for the user to enter. The list can have from 0 through 10,000 values. By default, all values the user enters are valid.

Remarks

- You need to use this subcommand only to specify a value other than the default.
- When the value specified for the DATA_TYPE parameter of the ADD_VARIABLE subcommand is REAL or COBOL, Screen Formatting accepts the values specified on this subcommand.

- For more information, see the NOS/VE Screen Formatting manual.

Debug 14

ACTIVATE_SCREEN DEBUG Subcommand

Purpose Activates screen mode.

Format

ACTIVATE_SCREEN or
ACTS
SOURCE_FILES=list of file
STATUS=status variable

Parameters

SOURCE_FILES or *SOURCE_FILE* or *SF*

Specifies the file or files containing the source text of the compiled program to be debugged. If this parameter is omitted, screen mode is initiated with the current file.

Remarks

- To use this command, the FILE_PROCESSOR attribute of each file must contain the name of the compiler that compiled the file. This is done with the CHANGE_FILE_ATTRIBUTE command before you begin the DEBUG session.
- You can enter ACTIVATE_SCREEN any time during an interactive DEBUG session.
- When ACTIVATE_SCREEN is entered, step mode is turned off. The Debug input and Debug output files are changed to the files used by Debug for screen mode.
- During a Debug session in screen mode, screen mode functions, certain line mode commands, and NOS/VE commands are available. To return to line mode, use the DEACTIVATE_SCREEN (DEAS) screen function.
- ACTIVATE_SCREEN allows you to use screen debugging without changing the interactive style for the rest of your interactive terminal session.
- For more information, see the Debug for NOS/VE Usage manual.

Examples The following example changes the Debug session from line mode to screen mode. The source program to be debugged in screen mode is $USER.FORT.

```
DB/activate_screen sf=$user.fort
```

CHANGE_DEFAULT DEBUG Subcommand

Purpose Changes one or more default Debug settings.

Format

CHANGE_DEFAULT or
CHANGE_DEFAULTS or
CHAD
MODULE=keyword or *application*
PROCEDURE=keyword or *application*
DEBUG_INPUT=file
DEBUG_OUTPUT=file
STATUS=status variable

Parameters *MODULE* or *M*

New default name for the MODULE parameter on subsequent Debug commands. If you specify $CURRENT, the default module is reset to the module that was executing when Debug gained control. If this parameter is omitted, the current default module remains unchanged.

PROCEDURE or *P*

New default name for the PROCEDURE parameter on subsequent Debug commands. If you specify $CURRENT, the default procedure is reset to the procedure that was executing when Debug gained control. If this parameter is omitted, the current default procedure remains unchanged.

DEBUG_INPUT or *DI*

New default file from which Debug commands are read when Debug next gains control. If this parameter is omitted, the current DEBUG_INPUT file remains unchanged. Unless otherwise specified, the initial DEBUG_INPUT file is $COMMAND.

NOTE

Unless a file position is specified in the file reference, the DEBUG_INPUT and DEBUG_OUTPUT files are positioned at the beginning-of-information the first time it is used. The file is not repositioned the next time it is used. Commands are read from the file sequentially. If an end-of-partition or an end-of-file is reached on the input file, program execution resumes.

DEBUG_OUTPUT or *DO*

New default file on which Debug output is written. The change takes effect immediately. Both break report messages and command output are written to this file. If this parameter is omitted, the current DEBUG_OUTPUT file remains unchanged. Unless otherwise specified, the initial DEBUG_OUTPUT file is $OUTPUT.

Remarks For more information, see the Debug for NOS/VE Usage manual.

Examples Read commands from the file DBIN the next time Debug gains control:

```
DB/change_default debug_input=dbin
```

Write output to file $LIST:

```
DB/change_default debug_output=$list
```

Specify the default module name:

```
DB/chad module=main
```

CHANGE_MEMORY DEBUG Subcommand

Purpose Changes the contents of memory starting at the specified address. You can only change values in memory locations for which you have write permission.

Format

CHANGE_MEMORY or
CHAM
ADDRESS=application
VALUE=any
TYPE=keyword
REPEAT_COUNT=keyword or *integer*
STATUS=status variable

Parameters

ADDRESS or *A*

Address of the first byte of memory to be changed.

The form of an address is rssssooooooooo(16) where r is the ring number, sss is the segment number, and oooooooo is the offset from the beginning of the segment. You can get machine addresses from the cross-reference and load maps for your program. This parameter is required.

VALUE or **V**

New memory value. An integer value completely replaces the contents of eight bytes. A string value is interpreted as a hexadecimal or ASCII string depending on the TYPE parameter. This parameter is required.

TYPE or *T*

Type of data specified by the VALUE parameter. If this parameter is omitted, a string value is assumed to be a hexadecimal value.

ASCII (A)

ASCII string value.

HEX (H)

Hexadecimal string value.

INTEGER (I)

Integer value.

REPEAT_COUNT or *RC*

Number of times the value is to be repeated in memory. If you specify ALL, it repeats the value until the end of the data segment containing the address. If this parameter is omitted, a value of 1 is used.

Remarks

For more information, see the Debug for NOS/VE Usage manual.

Examples Replace four bytes of memory beginning at location B02200001112(16) with the hexadecimal string '1010aaab':

```
DB/change_memory address=b02200001112(16) ..
DB../value='1010aaab'
```

Replace six bytes of memory beginning at location B02200000055(16) with the ASCII string 'STRING':

```
DB/change_memory address=b02200000055(16) ..
DB../value='string' type=ascii
```

Replace eight bytes of memory beginning at location B02300000223(16) with the integer value 44:

```
DB/change_memory address=b02300000223(16) value=44
```


CHANGE_PROGRAM_VALUE DEBUG Subcommand

Purpose Changes the value of named program variables. Replacement values are entered in the same format as defined in your program, not as they are represented in memory.

Format

CHANGE_PROGRAM_VALUE or
CHAPV
NAME=application
VALUE=list of "value application
MODULE=application
PROCEDURE=application
RECURSION_LEVEL=integer
RECURSION_DIRECTION=keyword
STATUS=status variable

Parameters *NAME* or *N*

Name of the program variable in the source program whose value is to be changed.

VALUE or **V**

New value for the variable. This parameter is required.

MODULE or *M*

Name of the module that contains the variable. If this parameter is omitted, the default module (the module executing when Debug gained control or the module specified by the CHANGE_DEFAULTS command) is used.

PROCEDURE or *P*

Specifies the name of the procedure that contains the variable. If this parameter is omitted, the default procedure (the procedure executing when Debug gained control or the procedure specified by the CHANGE_DEFAULTS command) is used.

NOTE

The following two parameters, RECURSION_LEVEL and RECURSION_DIRECTION, are applicable only when debugging programs written in languages that support recursion (such as CYBIL and PASCAL). The parameter values are ignored for all other languages.

RECURSION_LEVEL or *RL*

Indicates the particular call of a recursive procedure to be used. If RECURSION_DIRECTION specifies FORWARD, the integer 1 specifies the first call, 2 the second call, and so forth; if RECURSION_DIRECTION is omitted or specifies BACKWARD, the integer 1 specifies the most recent call, 2 its predecessor, and so forth;

Recursion applies only to stack variables; it does not apply to variables stored in either a common block or the $STATIC section.

The default value is 1.

RECURSION_DIRECTION or *RD*

Indicates the order in which calls are counted by the RECURSION_LEVEL parameter. The default value is BACKWARD.

FORWARD

The integer 1 specifies the first call, 2 the second call, and so forth.

BACKWARD

The integer 1 specifies the most recent call, 2 its predecessor, and so forth.

Remarks For more information, see the Debug for NOS/VE Usage manual.

Examples The first example refers to the following definition:

```
COMMON /BLK/ DVAL, RVAL, IVAL, ZVAL
DATA DVAL, RVAL, IVAL, ZVAL /20.0D+0, 3.45E+01, 30,
*(+20.0,20.3)/
```

Display initial value of variable DVAL:

```
DB/display_program_value name=dval
dval = 20.
```

Change the value of variable DVAL to 30.0:

```
DB/change_program_value name=dval value=+30.0d+0
```

Display the new value of DVAL:

```
DB/dispv dval
dval = 30.
```

Change the value of variable INDEX:

```
DB/chapv name=index value=63 module=ff_pp
```

Change the value of logical variable VAR:

```
DB/change_program_value var value=true
```

CHANGE_REGISTER DEBUG Subcommand

Purpose Changes the value of the P, A, or X or state registers.

Format

CHANGE_REGISTER or
CHANGE_REGISTERS or
CHAR
KIND=keyword
NUMBER=list of range of: keyword or *integer*
VALUE=any
TYPE=keyword
STATUS=status variable

Parameters

KIND or *K*

Specifies the register to change. Options are:

Omitted

Same as KIND=P.

P

Changes the P register for the procedure or function currently being executed. Changing the P register changes the point in the program at which Debug resumes execution.

A

Changes the A registers for the procedure or function currently being executed.

X

Changes the X registers for the procedure or function currently being executed.

S

Changes the state registers. If the state register designates the P register, the user mask, the user condition, or the monitor condition, the copy of the register for the procedure or function currently being executed is changed. For other state registers, the current register is changed.

NUMBER or *N*

Indicates which A, X, or state registers specified by the KIND parameter will change. This parameter is ignored if KIND=P because there is only one P register.

The A and X registers can be saved and changed when Debug gains control. However, some registers are not always saved; a message is issued for each register that cannot be changed because is was not saved.

Access to some state registers is privileged; a message is issued when you try to change a register for which you are not privileged.

Options are:

Omitted

Changes the zero register.

Integer or range of integers

Changes a set of registers.

If KIND=A or X, integer can be 0 through 15.

If KIND=S, integer can be 0 through 255. Not all of these numbers in the integer range are assigned to state registers and the definition of some state registers is machine-dependent. For a complete list of state register numbers, see the Virtual State Hardware reference manual.

ALL

Changes all A (if KIND=A), all X (if KIND=X), or all state (if KIND=S) or all sets of (if KIND=ALL) registers.

VALUE or V

Specifies the new value of the register. Options are:

Omitted

Same as integer or string allowed for KIND=P.

Integer

If KIND=P or A, integer must be in the range 0 through 0FFFFFFFFFFF(16).

If KIND=X or S, integer must be in the range -7FFFFFFFFFFFFFFF(16) through 7FFFFFFFFFFFFFFF(16).

The upper 4 bits are ignored when changing the P register because the ring number in P cannot be changed.

The upper bits of the register are set to zero if an integer is negative or to 1 if an integer is positive when the value does not fill the register.

String

If KIND=P or A, string can be a hexadecimal string containing a maximum of 12 hexadecimal digits (spaces are ignored); each hexadecimal digit corresponds to 4 bits.

If KIND=X, string can be a hexadecimal string containing a maximum of 16 hexadecimal digits (spaces are ignored); each hexadecimal digit corresponds to 4 bits or an ASCII string containing a maximum of eight ASCII characters; each character corresponds to one byte.

If KIND=S, a string cannot be specified for VALUE.

If a string value does not fill the register (it is less than 16 hexadecimal digits or 8 ASCII characters), the string value is left-justified with remaining bytes unchanged.

TYPE or *T*

Type of data specified by the VALUE parameter. If this parameter is omitted, a string value is assumed to be a hexadecimal value and a numeric value is assumed to be an integer.

ASCII (A)

ASCII string value.

HEX (H)

Hexadecimal string value.

INTEGER (I)

Integer value.

Remarks For more information, see the Debug for NOS/VE Usage manual.

Examples Change the current value of the P register to 0A02200004500(16). The upper 4 bits for the ring number are ignored.

```
DB/change_register kind=p value=0a02200004500(16)
```

Change the current value of the X7 register to 'ABCDEFGH':

```
DB/char kind=x number=7 value='abcdefgh' type=ascii
```


CHANGE_USER_MASK DEBUG Subcommand

Purpose Enables or disables traps for certain system conditions that terminate program execution.

Format **CHANGE_USER_MASK** or **CHAUM**..

ARITHMETIC_OVERFLOW=boolean
ARITHMETIC_LOSS_OF_SIGNIFICANCE=boolean
DEBUG=boolean
DIVIDE_FAULT=boolean
EXPONENT_OVERFLOW=boolean
EXPONENT_UNDERFLOW=boolean
FP_INDEFINITE=boolean
FP_LOSS_OF_SIGNIFICANCE=boolean
INVALID_BDP_DATA=boolean
STATUS=status variable

Parameters *ARITHMETIC_OVERFLOW* or *AO*

Specifies the desired mode for the arithmetic overflow condition mask flag (condition bit). Options are:

Omitted

The arithmetic overflow condition mask flag is left unchanged.

ON

The arithmetic overflow condition mask flag is enabled (masked ON) for traps.

OFF

The arithmetic overflow condition mask flag is disabled (masked OFF) for traps.

ARITHMETIC_LOSS_OF_SIGNIFICANCE or *ALOS*

Specifies the desired mode for the arithmetic loss of significance condition mask flag (condition bit). Options are:

Omitted

The arithmetic loss of significance condition mask flag is left unchanged.

ON

The arithmetic loss of significance condition mask flag is enabled (masked ON) for traps.

OFF

The arithmetic loss of significance condition mask flag is disabled (masked OFF) for traps.

DEBUG or *D*

Specifies the desired mode for the debug condition mask flag (condition bit). Options are:

Omitted

The debug condition mask flag is left unchanged.

ON

The debug condition mask flag is enabled (masked ON) for traps. Break events that are detected by the Debug hardware are reported.

OFF

The debug condition mask flag is disabled (masked OFF) for traps. Break events that are detected by the Debug hardware are not reported.

DIVIDE_FAULT or *DF*

Specifies the desired mode for the divide fault condition mask flag (condition bit). Options are:

Omitted

The divide fault condition mask flag is left unchanged.

ON

The divide fault condition mask flag is enabled (masked ON) for traps.

OFF

The divide fault condition mask flag is disabled (masked OFF) for traps.

EXPONENT_OVERFLOW or *EO*

Specifies the desired mode for the exponent overflow condition mask flag (condition bit). Options are:

Omitted

The exponent overflow condition mask flag is left unchanged.

ON

The exponent overflow condition mask flag is enabled (masked ON) for traps.

OFF

The exponent overflow condition mask flag is disabled (masked OFF) for traps.

EXPONENT_UNDERFLOW or *EU*

Specifies the desired mode for the exponent underflow condition mask flag (condition bit). Options are:

Omitted

The exponent underflow condition mask flag is left unchanged.

ON

The exponent underflow condition mask flag is enabled (masked ON) for traps.

OFF

The exponent underflow condition mask flag is disabled (masked OFF) for traps.

FP_INDEFINITE or *FPI* or *FI*

Specifies the desired mode for the floating point indefinite condition mask flag (condition bit). Options are:

Omitted

The floating point indefinite condition mask flag is left unchanged.

ON

The floating point indefinite condition mask flag is enabled (masked ON) for traps.

OFF

The floating point indefinite condition mask flag is disabled (masked OFF) for traps.

FP_LOSS_OF_SIGNIFICANCE or *FPLOS* or *FLOS*

Specifies the desired mode for the floating point loss of significance condition mask flag (condition bit). Options are:

Omitted

The floating point loss of significance condition mask flag is left unchanged.

ON

The floating point loss of significance condition mask flag is enabled (masked ON) for traps.

OFF

The floating point loss of significance condition mask flag is disabled (masked OFF) for traps.

INVALID_BDP_DATA or *IBDPD* or *IBD*

Specifies the desired mode for the invalid business data processing data condition mask flag (condition bit). Options are:

Omitted

The invalid business data processing data condition mask flag is left unchanged.

ON

The invalid business data processing data condition mask flag is enabled (masked ON) for traps.

OFF

The invalid business data processing data condition mask flag is disabled (masked OFF) for traps.

Remarks For more information, see the Debug for NOS/VE Usage manual.

Examples The following example disables the floating point indefinite condition mask flag in the user mask register:

```
DB/change_user_mask fpi=off
```

The following command displays the new values in the user mask register:

```
DB/disum

privileged_instruction_fault      = true
unimplemented_instruction         = true
free_flag                         = true
process_interval_timer            = true
inter_ring_pop                    = true
critical_frame_flag               = true
not_assigned                      = true
divide_fault                      = true
debug                             = false
arithmetic_overflow               = false
exponent_overflow                 = true
exponent_underflow                = true
fp_loss_of_significance           = false
fp_indefinite                     = false
arithmetic_loss_of_significance   = true
invalid_bdp_data                  = true
```

The following example disables the arithmetic overflow and invalid business data processing data condition mask flags:

```
DB/chaum ao=off ibdpd=off
```

The following command displays the new values of the user mask register:

```
DB/disum

privileged_instruction_fault      = true
unimplemented_instruction         = true
free_flag                         = true
process_interval_timer            = true
inter_ring_pop                    = true
critical_frame_flag               = true
not_assigned                      = true
divide_fault                      = true
debug                             = false
arithmetic_overflow               = false
exponent_overflow                 = true
exponent_underflow                = true
fp_loss_of_significance           = false
fp_indefinite                     = false
arithmetic_loss_of_significance   = true
invalid_bdp_data                  = false
```


$CURRENT_LINE DEBUG Function

Purpose Returns an integer identifying the current line number in the program where Debug has control.

Format **$CURRENT_LINE** or
$CL

Parameters None.

Remarks For more information, see the Debug for NOS/VE Usage manual.

$CURRENT_MODULE DEBUG Function

Purpose Returns a string identifying the name of the module where execution stopped.

Format **$CURRENT_MODULE** or
$CM

Parameters None.

Remarks For more information, see the Debug for NOS/VE Usage manual.

$CURRENT_PROCEDURE DEBUG Function

Purpose Returns a string identifying the name of the procedure where execution is stopped.

Format **$CURRENT_PROCEDURE** or
$CP

Parameters None.

Remarks For more information, see the Debug for NOS/VE Usage manual.

$CURRENT_PVA DEBUG Function

Purpose Returns an integer identifying the process virtual address (PVA) where execution is stopped.

Format **$CURRENT_PVA** or
$CPVA

Parameters None.

Remarks For more information, see the Debug for NOS/VE Usage manual.

DELETE_BREAK DEBUG Subcommand

Purpose Deletes one or more break definitions.

Format **DELETE_BREAK** or
DELETE_BREAKS or
DELB
BREAK=list of name
STATUS=status variable

Parameters **BREAK** or **BREAKS** or **B**

Specifies the break definitions to be deleted. If the keyword ALL appears in the list of names, all breaks are deleted. This parameter is required.

Remarks For more information, see the Debug for NOS/VE Usage manual.

Examples Delete break definitions B1, B2, and B3:

```
DB/delete_breaks breaks=(b1,b2,b3)
```

Delete all break definitions:

```
DB/delete_breaks all
```

Delete break definition B4:

```
DB/delete_break b4
```

DISPLAY_BREAK DEBUG Subcommand

Purpose Displays specified break definitions. The break name, events, address, and any commands associated with the break are displayed.

Format

DISPLAY_BREAK or
DISPLAY_BREAKS or
DISB
BREAKS=list of name
OUTPUT=file
STATUS=status variable

Parameters *BREAKS* or *BREAK* or *B*

Break definitions to be displayed. If the keyword ALL appears in the list of names, all breaks are displayed. If this parameter is omitted, all breaks are displayed.

OUTPUT or *O*

File on which the break definitions are written. The default file is the current default Debug output file.

Remarks For more information, see the Debug for NOS/VE Usage manual.

Examples Display break definitions B1, B3:

```
DB/display_breaks breaks=(b1,b3)

    --  Break B1
    --  event(s) = execution
    --  location: M=TEST L=16

    --  Break B3
    --  event(s) = eu
    --  range: M=TEST L=18 to M=TEST L=19 BO=15
```

Display all break definitions:

```
DB/display_breaks

    --  Break B1
    --  event(s) = execution
    --  location: M=TEST L=16

    --  Break B2
    --  event(s) = execution
    --  range: M=TEST L=22 to M=test BO=39

    --  Break B3
    --  event(s) = eu
    --  range: M=TEST L=18 to M=TEST L=19 BO=15

    --  Break B4
    --  event(s) = execution
    --  location: M=TEST L=25
```


DISPLAY_CALL DEBUG Subcommand

Purpose Displays information about the dynamic call chain.

Format

DISPLAY_CALL or
DISPLAY_CALLS or
DISC
COUNT=keyword or *integer*
START=integer
DISPLAY_OPTION=list of keyword
OUTPUT=file
STATUS=status variable

Parameters *COUNT* or *C*

Number of calls to be displayed. If the keyword ALL is specified or this parameter is omitted, all calls are displayed.

START or *S*

Call in the chain to be displayed first. The integer 1 specifies the most recent call, 2 the predecessor to the most recent call, and so forth. The default value is 1, the most recent call.

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Type of information to be displayed. If this parameter is omitted, only calls which are in user code are displayed.

USER_CALLS (UC)

Displays only calls which are in user code.

SYSTEM_CALLS (SC)

Displays only calls which are not part of the user code.

ALL_CALLS (AC)

Displays both user calls and system calls.

VARIABLE_VALUES (VV)

Displays all variables known to the procedure.

OUTPUT or *O*

File on which the call information is written. If this parameter is omitted, the information is written to the current DEBUG_OUTPUT file.

Remarks For more information, see the Debug for NOS/VE Usage manual.

Examples Display all the calls on the call chain beginning with the second most recent call:

```
DB/display_calls start=2

 -- Called from procedure IO1 module IO1 at line 16
    byte offset 12
 -- Called from procedure TEST module TEST at line 54
    byte offset 12
```

Display system code calls on the call chain beginning with the third most recent call:

```
DB/display_calls start=3 display_option=system_calls

 -- Traceback from module MLM$OUTPUT_FLOATING_NUMBER
    byte offset 14C(16)
 -- Called from procedure FLP$SEQ_ACC_LST_OUT module
    FLM$LIST_DIRECTED_IO byte offset 61E(16)
```

Display all user code calls on the call chain, as well as all variables known to the procedure:

```
DB/display_calls display_options=(user_call, ..
DB../variable_values)

 -- Traceback from procedure TEST module TEST at
    line 42
 -- DISPLAY OF ALL VARIABLES IN TEST

        ARAY =  0( 20 OCCURRENCES)
        ARG = 0
        BASE = 2.
        DVAL = 20.
        I = 0
        IVAL = 30
        LOG1 = FALSE
        LOG2 = FALSE
        LOG3 = FALSE
        PIND = 70000000000000000(16)
        RESLT = 0.
        RVAL = 34.5
        X1 = 0
        XPWR = 4094.
        Y1 = 0.
        ZERO = 0.
        ZVAL = (20.,20.3)
```


DISPLAY_DEBUGGING _ENVIRONMENT DEBUG Subcommand

Purpose Displays the following:

- Current defaults for module, procedure, DEBUG_INPUT, DEBUG_OUTPUT
- Total number of breaks you have set and Debug has set
- STEP_MODE values
- Location in your program where execution has stopped

Format

DISPLAY_DEBUGGING_ENVIRONMENT or **DISDE**

DISPLAY_OPTION=list of keyword
OUTPUT=file
STATUS=status variable

Parameters

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Type of information to be displayed. If this parameter is omitted, defaults, breaks, STEP_MODE attributes, and user addresses are displayed.

DEFAULTS (D)

Current default values for module, procedure, DEBUG_INPUT, and DEBUG_OUTPUT.

Unless the CHANGE_DEFAULT subcommand has been specified, the default module and procedure is where execution has stopped in your task. The text $CURRENT is output if module or procedure has not been initialized.

BREAKS (B)

Number of breaks you have set, number of breaks currently in use by Debug, and the maximum number of allowed breaks.

STEP_MODE (SM)

Current STEP_MODE attributes.

USER_ADDRESS (UA)

Location where execution has stopped in your program.

ALL

Displays defaults, breaks, STEP_MODE attributes, and the user address.

OUTPUT or *O*

File where the debugging environment display is written. If this parameter is omitted, the current default Debug output file is written.

Remarks

For more information, see the Debug for NOS/VE Usage manual.

Examples

Display the number of breaks set, the number of breaks in use by Debug, the maximum number of allowed breaks, and the location where execution has stopped:

```
DB/display_debugging_environment do=(b,ua)

 -- The number of breaks set by the user is 1.
 -- The number of breaks in use by DEBUG is 0.
 -- The number of available breaks is 63.
 -- Execution is currently stopped at B 02E 0000013C
    which, in higher symbolic terms is M=TEST L=36
    BO=12.
```

Write defaults, breaks, STEP_MODE attributes, and location where execution has stopped to file FILE1 and returns the status to variable SS:

```
DB/disde do=all output=file1 status=ss
```

Write defaults, breaks, STEP_MODE attributes, and location where execution has stopped to the current default Debug output file:

```
DB/disde

 --  Default module is $CURRENT.
 --  Default procedure is $CURRENT.
 --  Default debug_input file is DBIN.
 --  Default debug_output file is $OUTPUT.
 --  The number of breaks set by the user is 5.
 --  The number of breaks in use by DEBUG is 0.
 --  The number of available breaks is 59.
 --  Step_mode is OFF.
 --  Execution is currently stopped at B 02E 0000013C
     which, in higher symbolic terms is M=TEST L=36
     BO=12
```

DISPLAY_DEBUG_TASK_STATUS DEBUG Subcommand

Purpose Displays task information for each task of a single or multi-task debugging session.

Format

DISPLAY_DEBUG_TASK_STATUS or
DISDTS
TASK_NUMBER=list of range of: keyword or *integer*
OUTPUT=file
STATUS=status variable

Parameters *TASK_NUMBER* or *TASK_NUMBERS* or *TN*

Specifies which tasks' information you want displayed. Options are:

Omitted

Same as TASK_NUMBER=ALL.

List of integer

Specifies the tasks' information you want displayed. Values can be a list of one or more positive integers representing the task number or numbers assigned to each task by Debug.

ALL

Information for all tasks is displayed.

OUTPUT or *O*

Specifies the file on which the task information is to be written. Options are:

Omitted

Writes to he current Debug output file.

File

Writes to the named file. You can position the file by appending a position indicator to the file name (.$BOI, .$ASIS, .$EOI).

Remarks For more information, see the Debug for NOS/VE Usage manual.

Examples The following example displays current task information for all tasks of a multi-task debugging session. Task 1 is in control; task 2 is suspended at line 186 of module PASSED_PARAMETERS and is waiting for control.

```
DB/display_debug_task_status task_number=all

 --  TASK  1: system id: 14
 --  Status: Controlling Debug executing
 --  Stopped for termination at an unknown address
 --  Terminated by returning
 --   The status at termination was: NORMAL.

 --  TASK  2: system id: 16
 --  Status: Waiting for control
 --  Stopped before task start at
       M=PASSED_PARAMETERS L=186

 --  Style: debugging all tasks
 --  Currently active task(s) - 2
 --  Total task(s) activated - 2
```


DISPLAY_MEMORY
DEBUG Subcommand

Purpose Displays information located at a location to which you have read access. The location can be specified by section and module or by address.

Format

DISPLAY_MEMORY or
DISM
ADDRESS=application
SECTION=keyword or *name*
MODULE=application
BYTE_OFFSET=application
BYTE_COUNT=integer
REPEAT_COUNT=keyword or *integer*
OUTPUT=file
STATUS=status variable

Parameters *ADDRESS* or *A*

Address of the first byte of memory to be displayed. If the ADDRESS parameter is omitted, the location must be specified by the SECTION and MODULE parameters.

The address has the format rsssoooooooo(16) where r is the ring number, sss is the segment number, and oooooooo is the offset from the beginning of the segment. You can use the BYTE_OFFSET parameter to modify the starting address of memory to be displayed. This parameter is required.

SECTION or *SEC*

Memory section containing the data to be displayed.

$BLANK

Section that contains unnamed common.

$BINDING

Section that contains the links to external procedures and the data for the module.

$LITERAL

Section containing the literal data (for example, long constants) of the module.

$STATIC

Section containing the static (not on the run-time stack) variables not explicitly allocated to a named section of the module.

When you use SECTION to specify a location, you must qualify it with the MODULE parameter. You can use the BYTE_OFFSET parameter to modify the starting address of memory to be displayed.

MODULE or *M*

Module containing the data to be displayed. The MODULE parameter cannot be specified unless the SECTION parameter is also specified. If MODULE and SECTION are omitted, the location must be specified by the ADDRESS parameter.

BYTE_OFFSET or *BO*

Offset to the location specified by the SECTION and MODULE parameters or the ADDRESS parameter. If BYTE_OFFSET is omitted, a zero offset is used.

The address generated by adding BYTE_OFFSET to the base address must be within the memory block implied by the base address. The block size is the length of the section when the SECTION parameter is specified, and the length of the segment containing the machine address when the ADDRESS parameter is specified.

BYTE_COUNT or *BC*

Number of bytes in the item to be displayed. The default value is eight bytes.

REPEAT_COUNT or *RC*

Number of items of length BYTE_COUNT to be displayed. If REPEAT_COUNT is omitted, only one item is displayed.

The maximum amount of memory that can be displayed is limited to the block size implied by address (section length for SECTION and segment length for ADDRESS). A large integer causes all memory from the specified address to the end of the memory block to be displayed.

The keyword ALL displays all memory from the specified address to the end of the memory block.

OUTPUT or *O*

File on which the displayed information is written. If OUTPUT is omitted, the display is written to the current Debug output file.

Remarks

- This command allows you to debug your program even when compiler-generated symbol tables are not available, and to display memory areas that do not correspond to program identifiers. Each display line shows the memory contents in hexadecimal and ASCII formats; the relative byte offset from the initial address is also shown.

- The compiler-generated attributes list shows the section name and offset for all variables. To reference static variables, specify the section name and byte offset. To reference variables on the stack, specify the machine address of the stack frame and byte offset.
- To get the address of the stack frame of the procedure executing when Debug got control, display register A1 (see the DISPLAY_REGISTER command description). To get the address of other stack frames, display the save area of the wanted stack frame using the DISPLAY_STACK_FRAME command and get the value of register A1 from that display.
- You can use the DISPLAY_PROGRAM_VALUE command to display program variables when symbol tables are available.
- For more information, see the Debug for NOS/VE Usage manual.

Examples Display the first three bytes of the literal memory section for module MOD1:

```
DB/display_memory section=$literal module=mod1 ..
DB../byte_count=3
```

Display the first 32 bytes of the memory section DATA1 for module MOD2 as separate items:

```
DB/display_memory sec=data1 module=mod2 rc=4
```

Display the first 200 bytes of memory starting from the specified address:

```
DB/dism a=0b02400000224(16) bo=8 rc=25
```

DISPLAY_PROGRAM_VALUE DEBUG Subcommand

Purpose Displays the value of a program variable.

Format

DISPLAY_PROGRAM_VALUE or
DISPLAY_PROGRAM_VALUES or
DISPV
NAME=list of "variable application
MODULE=application

PROCEDURE=application
RECURSION_LEVEL=integer
RECURSION_DIRECTION=keyword
TYPE=keyword
VARIANT_SELECTION=list of: keyword or *any*
NAME_OPTION=list of keyword
SCOPE=keyword
SECTION=keyword or *application*
OUTPUT=file
STATUS=status variable

Parameters *NAME* or *N*

Name of the program variable in the source program whose value is to be displayed, or the keyword $ALL to display all variables in the procedure. This parameter is required.

The program variable can be one of the following:

- Simple variable or constant name.
- Substring reference.
- Subscripted name.
- Field reference.
- Pointer reference or dereference.

Subscripts can be constants or variables but not expressions. NAME cannot be a substring.

SCL string variables can be used to name long program names. To do this, assign a string containing the identifier to the SCL variable. Then use the SCL variable preceded by a question mark as the value of the NAME parameter.

MODULE or *M*

Name of the module that contains the variable. The default module is the module executing when Debug gained control or the module specified by the CHANGE_DEFAULT subcommand.

PROCEDURE or *P*

Name of the procedure that contains the variable. The default procedure is the procedure executing when Debug gained control or the procedure specified by the CHANGE_DEFAULTS command.

NOTE

The following two parameters, RECURSION_LEVEL and RECURSION_DIRECTION, are applicable only when debugging programs written in languages that support recursion (such as CYBIL and PASCAL). The parameter values are ignored for all other languages.

RECURSION_LEVEL or *RL*

Indicates the particular call of a recursive procedure to be used. If RECURSION_DIRECTION specifies FORWARD, the integer 1 specifies the first call, 2 the second call, and so forth. If RECURSION_DIRECTION is omitted or specifies BACKWARD, the integer 1 specifies the most recent call, 2 its predecessor, and so forth.

Recursion applies only to stack variables; it does not apply to variables stored in either a common block or the $STATIC section.

The default value is 1.

RECURSION_DIRECTION or *RD*

Indicates the order in which calls are counted by the RECURSION_LEVEL parameter. The default value is BACKWARD.

FORWARD

The integer 1 specifies the first call, 2 the second call, and so forth.

BACKWARD

The integer 1 specifies the most recent call, 2 its predecessor, and so forth.

TYPE or *T*

Data representation used for the display.

HEX (H)

Hexadecimal dump. The display includes the variable name, its starting address, and the data displayed as hexadecimal digits and as ASCII characters.

INTEGER (I)

Decimal integer. The data length must be from 1 through 8 bytes. Each element of an array is displayed as a separate integer.

REAL (R)

Floating-point. The data length must be 8 bytes. Each element of an array is displayed as a separate floating-point number.

If TYPE is omitted, the data representation used corresponds to the data type as defined in the program.

VARIANT_SELECTION or *VS*

Selector value specifying the tagless variant to be displayed. The specified value can be an integer, boolean, name or one-character string, but it cannot be a string longer than one character. The value specifies the ordinal of the variant to be displayed.

Debug prompts the user when the VARIANT_SELECTION parameter is required, but has not been supplied.

NAME_OPTION or *NAME_OPTIONS* or *NO*

Qualifies the identifier(s) given for the NAME parameter. Options are:

Omitted

There is no default for the NAME_OPTION parameter when a single identifier is specified for the parameter. If $ALL is specified for the NAME parameter, the default for the NAME_OPTION parameter is VARIABLES.

CONSTANTS (C)

The identifier in the source program must be a constant.

VARIABLES (V)

The identifier in the source program must be a variable.

PARAMETERS (P)

The identifier in the source program must be a variable that was passed as a parameter to the default procedure or the procedure specified by the PROCEDURE parameter.

ALL

The identifier in the source program can be either a constant or a variable.

NOTE

NAME_OPTIONS=PARAMETERS cannot be used with the SECTION parameter.

SCOPE or *SCO*

Determines the type of search for identifiers specified by the NAME parameter. Options are:

GLOBAL (G)

The value of the NAME parameter must reference identifier(s) known outside the defining module. The Entry Point Table is searched to locate the identifier(s).

NOTE

GLOBAL cannot be used with the MODULE, PROCEDURE, RECURSION LEVEL, and RECURSION_DIRECTION parameters or with the NAME_OPTION =PARAMETERS.

MODULE (M)

The value of the NAME parameter must reference identifiers(s) defined at the outermost level of the module.

LOCAL (L)

The identifier(s) referenced by the NAME parameter must be defined in the procedure specified by the PROCEDURE parameter or by default.

SECTION or *SEC*

Displays a group of identifiers by specifying the section where they are stored. This parameter is valid only when the value of the NAME parameter is $ALL.

NOTE

The SECTION parameter cannot be used with the RECURSION_LEVEL and RECURSION_DIRECTION parameters or with NAME_OPTION=PARAMETERS or SCOPE=GLOBAL.

OUTPUT or *O*

File where the display information is written. The default is the current Debug output file.

Remarks For more information, see the Debug for NOS/VE Usage manual.

Examples The examples refer to the following definitions:

```
COMMON /BLK/ DVAL, RVAL, IVAL, ZVAL
DATA DVAL, RVAL, IVAL, ZVAL/20.0D+0, 3.45E+01, 30, (+20.0,20.3)/
```

Display the value of DVAL:

```
DB/display_program_value name=dval
dval = 20.
DB/
```

Display the value of RVAL:

```
DB/dispv name=rval
rval = 34.5
DB/
```

Display the value of IVAL:

```
DB/display_program_value ival
ival = 30
DB/
```

DISPLAY_REGISTER DEBUG Subcommand

Purpose Displays the contents of the P, A, X or state registers.

Format

DISPLAY_REGISTER or
DISPLAY_REGISTERS or
DISR
KIND=list of keyword
NUMBER=list of range of: keyword or *integer*
TYPE=keyword
OUTPUT=file
STATUS=status variable

Parameters *KIND* or *K*

Specifies the register to be displayed. Options are one or more of the following:

Omitted

Same as KIND=ALL_PROGRAM.

P

Displays the P register for the procedure or function currently being executed.

A

Displays the A registers for the procedure or function currently being executed.

X

Displays the X registers for the procedure or function currently being executed.

ALL_STATE or AS or S

Displays all state registers. For state registers which designate the P register, the user mask, the monitor condition, and user condition, the value of the register for the procedure or function currently being executed is displayed. For other state registers, the current values are displayed.

ALL_PROGRAM or AP

Displays all P, A, and X registers for the procedure or function currently being executed.

ALL

Displays all registers for the procedure or function currently being displayed.

NUMBER or *N*

Indicates which A, X, or state registers specified by the KIND parameter are displayed. This parameter is ignored if KIND=P because there is only one P register.

The A and X registers can be saved and displayed when Debug gains control. However, some registers are not always saved; a message is issued for each register that cannot be displayed because it was not saved.

Options are:

Omitted

Displays the zero register.

Integer or range of integers

Displays a set of registers.

If KIND=A or X, integer can be 0 through 15.

If KIND=S, integer can be 0 through 255. Not all of these numbers in integer range define state registers and access to some state registers is restricted. Debug will attempt to display undefined state registers, but will display a value of zero for restricted state registers.

ALL

Displays all A (if KIND=A), all X (if KIND=X), all state (if KIND=S), or all sets of (if KIND=ALL) registers.

TYPE or *T*

Type of data to be displayed. If this parameter is omitted, a string value is assumed to be a hexadecimal value and a numeric value is assumed to be an integer.

Options are:

ASCII (A)

ASCII string value.

HEX (H)

Hexadecimal string value.

INTEGER (I)

Integer value.

OUTPUT or *O*

File where the display information is written. The default file is the current Debug output file.

Remarks For more information, see the Debug for NOS/VE Usage manual.

Examples Display the contents of the P register in hexadecimal:

```
DB/display_register p

P=B 031 00000040
```

Display the contents of the A8 register in hexadecimal:

```
DB/display_register kind=a number=8 type=hex

A8=B 04E 000004A8
```

Display the contents of the X4, X5, X6, X7, X8, X9, and X10 registers in hexadecimal:

```
DB/disr kind=x number=4..10

X4=70000000   0000000
X5=40019482   53FC0CD1
X6=0000B01B   0000253A
X7=00000000   00000001
X8=00000000   00000064
X9=00000000   00000273
XA=00000000   000000CA
```

DISPLAY_STACK_FRAME DEBUG Subcommand

Purpose Displays the contents of one or more stack frames. Values are displayed in hexadecimal.

Format

DISPLAY_STACK_FRAME or
DISPLAY_STACK_FRAMES or
DISSF
COUNT=keyword or *integer*
START=integer
DISPLAY_OPTION=list of keyword
OUTPUT=file
STATUS=status variable

Parameters

COUNT or *C*

Number of stack frames to be displayed. The keyword ALL displays all stack frames. The default is one stack frame.

START or *S*

Frame on the stack to be displayed first. The integer 1 represents the most recent stack frame, 2 the predecessor of the most recent stack frame, and so forth. By default, the display begins with the most recent stack frame.

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Area of the stack frames to be displayed. By default, it displays both the automatic and save areas.

AUTO (A)

Displays the area that contains the automatic (dynamically allocated) variables of the procedure.

SAVE (S)

Displays the area that contains a copy of the registers of the procedure as they existed at the time of a call or trap.

ALL

Displays both the automatic and save areas.

OUTPUT or *O*

File on which the stack frame information is written. The default file is the current DEBUG_OUTPUT file.

Remarks For more information, see the Debug for NOS/VE Usage manual.

Examples Display the save area of the most recent stack frame:

```
DB/display_stack_frame display_option=save

SAVE AREA

P=B 035 00000026          VMID=0
UM=FFF7     UCR=0040     MCR=0000


A0=B 032 00000460       A1=B 032 00000408
A2=B 032 000003C0       A3=B 030 00000000
A4=B 032 00000390       A5=B 02F 00000020
A6=B 02E 00000000       A7=B 02F 00000000
A8=B 00F 00000018       A9=B 032 00000630
AA=B 032 00000A30       AB=F FFF 80000000
AC=F FFF 80000000       AD=B 032 00001058
AE=F FFF 80000000       AF=B 00B 000557F8

X0=0000B01D  00020060       X1=00000000  00000000
X2=0000FFFF  80000000       X3=000007FF  FFFFFFFF

X4=00000000  10000000       X5=00000000  00000008
X6=00000000  0000000D       X7=00000000  0000001D
X8=00000000  00000000       X9=00000000  00000008
XA=00000000  00000300       XB=00000000  00000000
XC=00000000  00000001       XD=00000000  00000022
XE=00000000  00010040       XF=00000000  0000004E
```

Display the automatic and save areas of the most recent stack frame:

```
DB/dissf  count=1

STACK FRAME 001        SEGMENT=032
00000000    00000000  00000000
00000008    00000000  00000000
00000010    30300000  00C0FFFF    00
00000018    80000000  00000000
00000020    B032B031  00000000     2 1
00000028    0000B01D  0009B346           F
00000030    0000B032  00000430       2   0
00000038    0040B032  00000400     @ 2
00000040    FF77B032  000003C0     w 2
00000048    FFFCB01B  00020F78           x
00000050    0000B032  00000390       2

SAVE AREA

P=B 035 00000026        VMID=0
UM=FFF7    UCR=0040    MCR=0000


A0=B 032 00000460      A1=B 032 00000408
A2=B 032 000003C0      A3=B 030 00000000
A4=B 032 00000390      A5=B 02F 00000020
A6=B 02E 00000000      A7=B 02F 00000000
A8=B 00F 00000018      A9=B 032 00000630
AA=B 032 00000A30      AB=F FFF 80000000
AC=F FFF 80000000      AD=B 032 00001058
AE=F FFF 80000000      AF=B 00B 000557F8

X0=0000B01D  00020060      X1=00000000  00000000
X2=0000FFFF  80000000      X3=000007FF  FFFFFFFF

X4=00000000  10000000      X5=00000000  00000008
X6=00000000  0000000D      X7=00000000  0000001D
X8=00000000  00000000      X9=00000000  00000008
XA=00000000  00000300      XB=00000000  00000000
XC=00000000  00000001      XD=00000000  00000022
XE=00000000  00010040      XF=00000000  0000004E
```

DISPLAY_USER_MASK DEBUG Subcommand

Purpose Symbolically displays the values of the condition mask flags (condition bits) of the user mask register for the procedure or function currently executing.

Format

DISPLAY_USER_MASK or
DISUM
OUTPUT=file
STATUS=status variable

Parameters *OUTPUT* or *O*

Specifies the file on which the user mask register information is to be written. Options are:

Omitted

Writes to the current Debug output file.

File

Writes to the named file. You can position the file by appending a position indicator to the file name (.$BOI, .$ASIS, .$EOI).

Remarks For more information, see the Debug for NOS/VE Usage manual.

Examples The following example displays the condition mask flag values of the user mask register.

```
DB/disum

privileged_instruction_fault     = true
unimplemented_instruction        = true
free_flag                        = true
process_interval_timer           = true
inter_ring_pop                   = true
critical_frame_flag              = true
not_assigned                     = true
divide_fault                     = true
debug                            = false
arithmetic_overflow              = false
exponent_overflow                = true
exponent_underflow               = true
```

```
fp_loss_of_significance          = false
fp_indefinite                    = true
arithmetic_loss_of_significance  = true
invalid_bdp_data                 = true
```

$MEMORY DEBUG Function

Purpose Returns the contents of a memory area.

Format

$MEMORY or
$MEM
(PARAMETER_1: application
PARAMETER_2: integer
PARAMETER_3: keyword)

Parameters

PARAMETER_1

Specifies the process virtual address. This parameter is required.

PARAMETER_2

The number of bytes to return. If parameter_3 is an integer, parameter_2 must be in the range of 1 through 8. If parameter_3 is a string, parameter_2 must be in the range of 1 through 256. If you omit this parameter, 6 bytes are returned.

PARAMETER_3

The type of value to return. If you specify an integer value, a hexadecimal integer with radix is returned. If you specify a string, a string is returned. If you omit this parameter, an integer is returned.

Remarks For more information, see the Debug for NOS/VE Usage manual.

$PROGRAM_VALUE DEBUG Function

Purpose Returns the value of the program element which is specified as the name parameter. Additional parameters for module, procedure, recursion level, and recursion direction can be specified to fully identify the named variable.

The $PROGRAM_VALUE function allows you to incorporate the values of program variables in SCL statements in order to enhance debugging capabilities.

Format

$PROGRAM_VALUE or
$PV
(PARAMETER_1: application
PARAMETER_2: application
PARAMETER_3: application
PARAMETER_4: integer
PARAMETER_5: keyword)

Parameters

PARAMETER_1

Name of the program element whose value is to be displayed. This parameter is required. Values can be one of the following types:

- Simple variable
- Subscripted name
- Field reference
- Pointer reference

The named variable must be used in your program.

Because names can be long, SCL string variables can be used as aliases for them. To do this, assign the SCL variable to a string containing the identifier. Then use the SCL variable preceded by a question mark as the value of the name parameter.

PARAMETER_2

Name of the module that contains the element specified by the name parameter. Omission causes the module executing when Debug gained control or the module specified by the CHANGE_DEFAULT subcommand to be used.

PARAMETER_3

Name of the procedure that contains the element specified by the name parameter. If you specify a procedure that is not in the active call chain, its automatic variables cannot be used because it has no stack frame. Omission causes the procedure executing when Debug gained control to be used if a module name is not specified. Otherwise, there

is no default procedure when a module name is specified and a procedure name is not specified; the element specified by the name parameter must exist at the module level.

PARAMETER_4

The particular call of a recursive procedure to be used. It must be a positive integer greater than zero. If the recursion direction parameter specified the keyword FORWARD, a value of 1 is the first call, 2 is the second call (the one called by the first call), and so on. If the recursion direction parameter is BACKWARD, 1 is the most recent call, 2 is the predecessor, and so on.

Omission causes a value of 1 to be used.

PARAMETER_5

Order in which calls to a recursive procedure are searched. It controls how the value of the recursion_level parameter is interpreted. It can be one of the following keywords:

FORWARD or F

If the RECURSION_LEVEL parameter specifies that the first call to the procedure is used, a 2 specifies the second call, and so on.

BACKWARD or B

If the RECURSION_LEVEL parameter specifies that the most recent call to the procedure is used, a 2 specifies its predecessor, and so on.

Omission causes BACKWARD to be used.

Remarks For more information, see the Debug for NOS/VE Usage manual.

Examples

```
DB/set_break name=b1 line=23 command= ..
DB../'if $program_value(index) <45 then; run; ifend'
```

QUIT DEBUG Subcommand

Purpose Ends the Debug session and returns control to NOS/VE. The session is terminated immediately; the program is not executed to completion.

Format **QUIT** or
QUI
STATUS=status variable

Remarks For more information, see the Debug for NOS/VE Usage manual.

$REGISTER
DEBUG Function

Purpose Returns the contents of a specified register in hexadecimal integer format, including radix. $REGISTER is useful when specified for the ADDRESS parameter value on the DISPLAY_MEMORY and CHANGE_MEMORY commands.

Format **$REGISTER** or
$REG
(PARAMETER_1: keyword
PARAMETER_2: integer)

Parameters **PARAMETER_1**

The type of register the value is returned from. P specifies a P register, A specifies an A register, X specifies an X register, and S specifies a state register.

PARAMETER_2

Specifies the register number the value is returned from.

Remarks For more information, see the Debug for NOS/VE Usage manual.

RUN
DEBUG Subcommand

Purpose Begins or resumes program execution once Debug has gained control. Execution continues until Debug again gains control. If the program has run to completion, entering the RUN command terminates program execution.

Format **RUN**
STATUS=status variable

Remarks

- Execution begins at the instruction whose address is stored in the P register of the program when the event that caused Debug to gain control occurred.
- If the P register points to the instruction that caused the event (such as division by zero), the same event will occur immediately after entering the RUN command. In this case, you must change the value in the P register with the CHANGE_REGISTER command or change the value of one of the operands with the CHANGE_PROGRAM_VALUE command before entering the RUN command.
- When Debug processes the RUN command, all previously created SCL blocks (except SET_BREAK command information and the name of the current DEBUG_INPUT and DEBUG_OUTPUT files) are lost. This means that all information about SCL commands, such as if-then blocks or while-for loops that span RUN commands are lost. Global variables must be recreated with XREF.
- For more information, see the Debug for NOS/VE Usage manual.

Examples The following sequence recreates the variable COUNT:

```
DB/crev  count  kind=integer  scope=job  v=0
DB/setb  b=one  l=one
DB/run  "BREAK ONE"
```

By specifying SCOPE=JOB, the variable COUNT will be retained past the RUN command.

SET_BREAK
DEBUG Subcommand

Purpose Defines a break.

Format

SET_BREAK or
SETB
BREAK=name
EVENT=list of name
LINE=integer
STATEMENT=integer
STATEMENT_LABEL=application

NAME=application
SECTION=keyword or *application*
MODULE=application
PROCEDURE=application
ENTRY_POINT=application
ADDRESS=application
BYTE_OFFSET=application
BYTE_COUNT=integer
COMMANDS=string
STATUS=status variable

Parameters *BREAK* or *B*

Name of the break. By default, Debug assigns a unique name and displays the name assignment to the user.

The name is used to reference the break definition in the DISPLAY_BREAK and DELETE_BREAK commands. The name is displayed in the break report message when the break occurs.

A break cannot be named ALL. The break name must not contain the character '$'. The form is:

EVENT or *EVENTS* or *E*

One or more events that will cause the break. If you specify more than one event, the break occurs for any of the events.

ARITHMETIC_OVERFLOW (AO)

Breaks when an arithmetic overflow occurs on an instruction in the specified address range. The P register points to the instruction that caused the overflow.

ARITHMETIC_SIGNIFICANCE (AS)

Breaks when arithmetic significance is lost on an instruction in the specified address range. The P register points to the instruction that caused the loss of significance.

BRANCH (B)

Breaks before either a branch to or a return from any location in the specified address range occurs.

CALL (C)

Breaks before a subprogram call occurs to any address in the specified address range.

DIVIDE_FAULT (DF)

Breaks when division by zero occurs in an instruction in the specified address range. The P register points to the instruction that caused the division by zero.

EXECUTION (E)

Breaks before the instruction in the specified address range is executed.

If the address is specified by the line number, not every line is usable. For example, breaks cannot be set at ENDIF statements because it is not obvious when control reaches them.

EXPONENT_OVERFLOW (EO)

Breaks when an exponent overflow occurs in an instruction in the specified address range. The P register points to the instruction following the one that caused the overflow.

EXPONENT_UNDERFLOW (EU)

Breaks when an exponent underflow occurs in an instruction in the specified address range. The P register points to the instruction following the one that caused the underflow.

FLOATING_POINT_INDEFINITE (FPI)

Breaks when the result of a floating-point operation is indefinite in an instruction in the specified address range. The P register points to the instruction following the one that caused the results to be indefinite.

FLOATING_POINT_SIGNIFICANCE (FPS)

Breaks when significance is lost during a floating-point operation in an instruction in the specified address range. The P register points to the instruction following the one that caused the loss of significance. This event will not occur unless your program sets the floating-point loss-of-significance bit in the user mask register.

INVALID_BDP_DATA (IBD)

Breaks when a business data processing (BDP) instruction fault occurs in an instruction in the specified address range. The P register points to the instruction that caused the fault. The BDP instructions are described in the Virtual State Hardware reference manual.

READ (R)

Breaks before a read occurs from the specified address range. The break occurs only if the first byte of the item to be read is within the address range.

READ_NEXT_INSTRUCTION (RNI)

Breaks before the instruction in the specified address range is executed.

WRITE (W)

Breaks before a write occurs into the specified address range. The break occurs only if the first byte of the item to be written is within the address range.

The default event is EXECUTION.

NOTE

The following optional parameters (up to the COMMAND parameter) specify the location at which the break occurs. For the break to occur, the specified event must occur within the range defined by the address parameters. If all of these parameters are omitted, an address range of one byte is used.

LINE or *L*

Line number in the module. The module is specified by the MODULE parameter.

STATEMENT or *S*

Statement in the multi-statement line specified by the LINE parameter. The statements are numbered in consecutive order beginning with 1.

If STATEMENT is omitted, the default is 1.

STATEMENT_LABEL or *SL*

Source statement label at which to set the break. The module is specified by the MODULE parameter. The procedure is specified by the PROCEDURE parameter.

The parameter value depends on the programming language.

For FORTRAN, BASIC, and PASCAL, a statement label is an integer.

For CYBIL, a statement label is a name enclosed in slashes (/name/).

For COBOL, a statement label is a COBOL_PARAGRAPH or COBOL_SECTION identifier.

NAME or *N*

Variable name on which a READ or WRITE break is set. You must also specify EVENT=READ or EVENT=WRITE. If NAME is omitted, the break is specified by another parameter.

SECTION or *SEC*

Memory section.

$BINDING

Section containing the links to external procedures and the data of the module.

$BLANK

Section containing unnamed common.

$LITERAL

Section containing the literal data (for example, long constants) of the module.

$STATIC

Section containing the static (not on the run-time stack) variables not explicitly allocated to a named section of the module.

Unless the MODULE parameter is also specified, the section must exist for the current default module.

The SECTION parameter cannot be specified for modules that are components of a bound module unless the section is a common block.

MODULE or *M*

Module name. The module may qualify another address parameter. Otherwise it specifies the first byte of the code section of the module.

If this parameter is omitted, the current default module is used. The default module can be specified by a CHANGE_DEFAULTS command. If not specified, it is the module executing when Debug gained control.

PROCEDURE or *P*

Procedure name specifies the first byte of the code section of the procedure. Unless the MODULE parameter is specified, the procedure must exist in the current default module.

You cannot specify the LINE or SECTION address parameters with PROCEDURE.

ENTRY_POINT or *EP*

Entry point name.

You can use the BYTE_OFFSET and BYTE_COUNT parameters to modify the ENTRY_POINT parameter. You cannot use other address parameters with this parameter.

ADDRESS or *A*

Address of the first byte of memory to be changed.

Its format is rsssoooooooo(16) where r is the ring number, sss is the segment number, and oooooooo is the offset from the beginning of the segment. You can get machine addresses from the cross-reference and load maps for your program.

BYTE_OFFSET or *BO*

Offset to the base address established by one of the address parameters. The default offset is zero.

The address generated by adding BYTE_OFFSET to the base address must be within the memory block implied by the base address. The block size is the length of the section when the SECTION parameter is specified, and the length of the segment containing the machine address when the ADDRESS parameter is specified.

BYTE_COUNT or *BC*

Number of bytes in the item. The default byte count is 1.

COMMANDS or *COMMAND* or *C*

Optional string of commands to be executed by Debug, SCL, or any other active command processor when the break is honored. After the commands in the string have been executed, commands are read from the current Debug input file unless the string contains a RUN command.

If a command in the string includes a quoted string, that string must be enclosed in two single apostrophes.

No break report message is issued before the commands in the string are executed. If you want a message to be displayed, include an SCL DISPLAY_VALUE command in the string.

If an error is detected in one of the commands in the string, the break report message is issued, the error is reported, and commands are read from the Debug input file. The remaining commands in the string are not executed.

Remarks

- You specify one or more events and the location at which Debug takes control. When a specified event occurs, program execution is suspended and a message informs you which break occurred. At this point, you can enter another Debug command that can be processed by the operating system command or other active command utility (such as an SCL command).

- Debug gains control when the following events occur, even if you do not set a break for them:

  ARITHMETIC_OVERFLOW
  ARITHMETIC_SIGNIFICANCE
  DIVIDE_FAULT
  EXPONENT_OVERFLOW
  EXPONENT_UNDERFLOW
  FLOATING_POINT_INDEFINITE
  FLOATING_POINT_SIGNIFICANCE
  INVALID_BDP_DATA

  Specific breaks can be set for these events so that the specified command string can be executed when Debug gains control.

- For more information, see the Debug for NOS/VE Usage manual.

Examples Cause a break to occur when execution reaches line 10 of module PROG1:

```
DB/set_break line=10 module=prog1
-- Break name DBB$1 assigned to this break
```

Cause a break when a branch or return occurs to line 40 (of the module executing when Debug gained control):

```
DB/set_break break=b2 event=branch line=40
```


SET_FUNCTION_KEY DEBUG Subcommand

Purpose Enables you to create your own set or sets of function keys for screen debugging.

Format

SET_FUNCTION_KEY or
SETFK
NUMBER=keyword or integer
COMMAND_STRING=any
SHIFT=boolean
LABEL=string
STATUS=status variable

Parameters **NUMBER or N**

Specifies the keys to be defined. Options are:

integer

Specifies the number of the key to be defined. Values can be any integer from 1 through 16. These numbers correspond to the highlighted boxes in the menu of functions at the bottom of the screen. The numbers 1 through 8 correspond to the first row of boxes; 9 through 16 correspond to the second row of boxes.

If you specify a number that exceeds the maximum function key defined for your terminal, the value of the COMMAND_STRING is saved; to execute the COMMAND_STRING, enter its LABEL on the home line.

keyword

You can also specify one of the following keywords: BACK, BKW, DATA, DOWN, EDIT, FWD, HELP, STOP, UNDO, UP.

The keywords relate to keys on some terminals. If your terminal has defined sequences that relate to these keywords, you can create your own function keys using these keywords.

COMMAND_STRING or CS

Specifies the function to be performed when the specified key is pressed. Options are:

string

Specifies the command(s) to be executed when the specified key is pressed. Values can be any Debug or NOS/VE command or a null string if the key is to be unassigned. When more than one command is specified, separate them with semicolons.

keyword

Specifies the Debug screen function to be executed when the specified key is pressed. Values can be any Debug screen function label name.

SHIFT or S

For those terminals that have one key identifier next to each highlighted box in the menu of functions, the SHIFT parameter specifies whether the key to be used is shifted. For those terminals that have two key identifiers next to each highlighted box in the menu of functions, the SHIFT parameter indicates which key you use. Options are:

Omitted

Same as SHIFT=FALSE.

TRUE

The key is used shifted or the key corresponds to the top key identifier.

FALSE

The key is used unshifted or the key corresponds to the bottom key identifier.

LABEL or *L*

Specifies the label that is to appear in the menu of functions for the specified key. Options are:

Omitted

The first six characters of the COMMAND_STRING value are used for the label.

string

The specified string is used for the label.

Remarks For more information, see the Debug for NOS/VE Usage manual.

Examples The following SET_FUNCTION_KEY command defines the shifted F5 key to display the date and time. The key has a screen label of time.

```
DB/set_function_key number=5 ..
..DB/command_string='disv $date;disv $time(ampm)' ..
..DB/shift=true label='time'
```

The following SET_FUNCTION_KEY command defines the DATA key of the CDC 721 terminal to execute the NOS/VE DISPLAY_CATALOG command. The key has a screen label of catlst.

```
DB/setfk n=data cs='display_catalog' l='catlst'
```

The following SET_FUNCTION_KEY command defines the unshifted F7 key to execute the Debug Locate function. The key has a screen label of locate.

```
DB/setfk n=7 cs=locate
```

SET_SCREEN_OPTIONS DEBUG Subcommand

Purpose Enables you to change the appearance of your screen for a screen mode Debug session.

Format

SET_SCREEN_OPTIONS or
SETSO
MENU_ROWS=integer
COLUMNS=integer
SPLIT_SIZES=list of integer
STATUS=status variable

Parameters

MENU_ROWS or *MENU_ROW* or *MR*

Specifies the number of rows of function key prompts to display on your screen. Options are:

Omitted

The number of rows of function key prompts remains the same. The default number of rows is one.

0

Displays no function key prompts.

1

Displays one row of function key prompts (functions 1 through 8).

2

Displays two rows of function key prompts (functions 1 through 16).

COLUMNS or *C*

Specifies the number of columns to be displayed for terminals that support multiple screen sizes. Options are:

Omitted

The number of columns displayed remains the same.

Integer

Specifies the number of columns to be displayed. Values can range from 40 to the maximum number allowed for your terminal screen (up to 256). The number you enter is compared to the screen sizes set up in the terminal definition for your terminal. The number of columns displayed is the closest number as large or larger than the number you enter on the COLUMNS parameter. When first entering Debug, it assumes a value of 80 columns.

SPLIT_SIZES or *SPLIT_SIZE* or *SS*

Specifies the number of lines displayed in the Source and Output windows. Options are:

Omitted

The number of lines displayed in the Source and Output windows remains the same.

List of integer

Specifies the number of lines displayed in the Source and Output windows. Values can be a list of at most two integers in the range 1 to the maximum number allowed for your terminal screen (up to 255). If two values are specified, they must be enclosed in parentheses and separated by commas or spaces. The first value specifies the number of lines displayed in the Source window and the second value specifies the number of lines displayed in the Output window (not including header information). Each window must contain at least one line.

Because the Source window is allocated first with the remainder of the screen allocated to the Output window, the second value is not necessary.

When first entering Debug, the source window occupies the top three-fourths of the screen and the Output window occupies the bottom one-fourth of the screen. The number of lines displayed is determined by the size of your terminal screen.

Remarks

- SET_SCREEN_OPTIONS, when entered in line mode, determines the screen characteristics to be displayed when screen mode Debug is activated with the ACTIVATE_SCREEN command.
- When SET_SCREEN_OPTIONS is entered on the home line while in screen mode, the screen is updated immediately according to the parameters specified.
- For all omitted parameters, Debug assumes you want the same value used the last time you entered the SET_SCREEN_OPTIONS command in the current Debug session. If SET_SCREEN_OPTIONS has not been entered, Debug assumes the default values.

- Screen characteristics remain in effect throughout the Debug session until they are changed by another SET_SCREEN_OPTIONS command or by the tailoring functions of screen mode Debug. When you end the Debug session, all screen characteristics return to their default values.
- You can also include the SET_SCREEN_OPTIONS command in the debug_input file so that your screen characteristics are modified immediately when you begin a Debug session.
- For more information, see the Debug for NOS/VE Usage manual.

Examples The following example displays (when screen mode is activated) two rows of function key prompts, 132 columns, and a source window as large as possible.

```
DB/set_screen_options menu_rows=2 column=132 split_size=255
```

SET_STEP_MODE DEBUG Subcommand

Purpose Activates or deactivates step mode. In step mode, control is returned after a specified subset of a task is executed.

Format

SET_STEP_MODE or
SETSM
MODE=keyword
UNIT=keyword
MODULE=list of: keyword or *"module"*
PROCEDURE=list of: keyword or *"procedure"*
SPAN=integer
COMMAND=string
STATUS=status variable

Parameters

MODE

Indicates whether to activate or deactivate step mode. This parameter is required.

ON

Activates step mode.

OFF

Deactivates step mode. When step mode is off, any remaining parameters are ignored.

If you specify MODE=ON and step mode is already on, all previous values are replaced with the new parameter values.

UNIT or *U*

Length of each step. The default value is LINE.

LINE (L)

The step is reported before the code is executed for each line, except for the procedure lines.

PROCEDURE (P)

The step is reported each time a new procedure begins and after any prolog code for the procedure has executed.

COBOL_SECTION (CS)

The step is reported each time a section header is reached (COBOL programs only).

COBOL_PARAGRAPH (CP)

The step is reported each time a paragraph is reached (COBOL programs only).

MODULE or *M*

Used with the UNIT parameter to specify the modules reported. If this parameter is omitted, the current default module is used.

$ALL

A step is reported that is in any module.

$CURRENT

A step is reported only if the step occurs in the module where the program is executing when step mode is activated.

list of names

A step is reported if the step occurs in any of the named modules.

You cannot specify both the MODULE and PROCEDURE parameters in the same SET_STEP_MODE command.

PROCEDURE or *P*

Used with the UNIT parameter to specify the procedure reported. If the parameter is omitted, the current default procedure is used.

$ALL

A step is reported that is in any procedure.

$CURRENT

A step is reported only if the step occurs in the procedure where the program is executing when step mode is activated.

list of names

A step is reported if the step occurs in any of the named procedures.

You cannot specify both the MODULE and PROCEDURE parameters in the same SET_STEP_MODE command.

SPAN or *S*

Number of steps to occur before execution stops and the step is reported. By default, every step that occurs is reported.

COMMAND or *COMMANDS* or *C*

Optional string of commands to be executed when the step occurs.

If the string of commands includes a RUN command, the task is resumed and the step is not reported.

If the string does not include a RUN command, command input will be requested from the current DEBUG_INPUT file after the string of commands has been executed.

Remarks

- If step mode is activated, a RUN command causes your program to execute for the specified unit. You are then prompted for further command input.
- A string of commands can be associated with the step and will be processed each time the step is completed. Stepping with a unit of line or procedure is only available if the source program was compiled with OPT=DEBUG.
- Activating step mode is an effective debugging aid, but it uses a lot of execution time.
- For more information, see the Debug for NOS/VE Usage manual.

Examples

The following command sequence shows a command to turn on step mode, two RUN commands to execute two steps, and a command to turn off step mode. The value of variable x is displayed at each step.

```
DB/set_step_mode,on,command='display_program_value,x'
DB/run
x =   2.00000000000000E+0000
 --  DEBUG: step at M=$MAIN L=34 BO=212
DB/run
x =   2.00000000000000E+0000
 --  DEBUG: step at M=$MAIN L=35 BO=6
DB/set_step_mode off
```

DISPLAY_STATION 15

DISPLAY_STATION Command

Purpose Starts the Display Station utility session and allows you to display status information about a CDCNET batch I/O station.

Format

DISPLAY_STATION or
DISS
STATION_NAME=name
STATUS=status variable

Parameters

STATION_NAME or SN

Specifies the name of the I/O station. If the I/O station has no name, (for example, a private I/O station), use the name of the Control Facility obtained from your site administrator. This parameter is required.

Remarks

- You may need validation to use this command utility. See your site administrator for the appropriate validation, if needed.
- For more information, see the CDCNET Batch Device manual.

Examples The following examples starts an Display Station Utility session and specifies I/O station WORK1.

```
/display_station sn=work1
diss/
```

DISPLAY_BATCH_DEVICE_STATUS DISS Subcommand

Purpose Displays the attributes and status of an I/O station's devices. However, when entered from within the DISS utility, certain attributes and status information are displayed only to the owner of the output queue file being processed by the device.

Format

DISPLAY_BATCH_DEVICE_STATUS or
DISBDS
DEVICE_NAME=list of: keyword or name
DISPLAY_OPTION=keyword
OUTPUT=file
STATUS=status variable

Parameters

DEVICE_NAME or DN

Specifies a list of one or more names of the devices whose status is to be displayed. You can also select the devices by device type or select all devices associated with the I/O station. Options are:

Keyword	Description
PRINTERS	Displays the status of the I/O station's printers.
ALL	When entered from within the OPES utility, displays the status of all devices associated with the I/O station.

This parameter is required.

DISPLAY_OPTION or *DO*

Specifies the amount of information to be displayed. Options are:

Keyword	Description
ALL (A)	When used within OPES, displays all items of information for the selected devices. When used within DISS, displays selected items of information for the selected devices. See Remarks.
BRIEF (B)	Displays only the following items of information for the selected devices: Device name Device status Transfer status Percentage of the file transfer complete Last unsolicited message

The default is BRIEF.

OUTPUT or *O*

Specifies the name of the output file where the status information is to be displayed and, optionally, specifies how the file is to be positioned prior to use. Refer to File Reference in the NOS/VE System Usage manual for a description of file positioning prior to use.

The default is file $OUTPUT.

Remarks

- The DISS utility displays the following information for all devices.
  - Device name
  - Device status (active/stopped/not ready/down)
  - Device type
  - External device characteristic strings

- File acknowledgement status (yes/no)
- File transfer status (idle/busy/suspended)
- Last unsolicited message concerning the device
- Page width
- Terminal model

- For output devices, the DISS utility displays the following additional information. However, it displays the information marked with an asterisk to only the owner of the output queue file being processed by the device.
  - Banner highlight field (comment_banner/routing_banner/site_banner/user_file_name/user_name)
  - Banner page count
  - Code set (ASCII/ASCII128/ASCII95/ASCII64/ASCII48/EBCDIC)
  - Forms size
  - Forms code strings
  - Device alias
  - Maximum file size in bytes
  - Suppress carriage control (yes/no)
  - Transmission block size in bytes
  - Undefined_FE_action (print_after_spacing/print_before_spacing/discard_print_line)
  - Unsupported_FE_action (print_after_spacing/print_before_spacing/discard_print_line)
  - Vertical print density (SIX_ONLY/EIGHT_ONLY/SIX_ANY/EIGHT_ANY)
  - VFU image load option (init/oper/user/none)
  - VFU load procedure

- Additional information about the file being transferred to the device. However, the DISS utility displays the information marked with an asterisk to only the owner of the output queue file being processed by the device.
- * Family name of generating job
- * Login user name of generating job
- Percent complete
- System-supplied file name
- * System-supplied job name
- * User-supplied file name
- * User-supplied job name

- For input devices, the following additional information is displayed:
  - Information about the file being transferred from the device.
  - * Job destination name
  - input bytes transferred

Examples The following example displays a brief status of all devices associated with the I/O station:

```
diss/display_batch_device_status dn=all do=brief
Device-Name                    : PRINT1
  Device_Status                : active
  File_Transfer_Status         : idle
  Last-Unsolicited_Message     : finished

Device_Name                    : PRINT2
  Device-Status                : active
  File_Transfer_Status         : idle
  Last_Unsolicited_Message     : finished
diss/
```

DISPLAY_STATION_QUEUE_ENTRY DISS Subcommand

Purpose Displays status information about one or more files in the I/O station's output queue. When entered from within the DISS utility, displays status information about only those files in the output queue that you own.

Format

DISPLAY_STATION_QUEUE_ENTRY or
DISPLAY_STATION_QUEUE_ENTRIES or
DISSQE
NAME=list of: keyword or **name**
DISPLAY_OPTION=keyword
OUTPUT=file
STATUS=status variable

Parameters

NAME or **NAMES** or **N**

Specifies a list of one or more file names for which information is to be displayed. Either the system-supplied or user-supplied name can be used. You can also request the top 10 files or all files. If you are using this subcommand within the DISS utility, you must be the owner of the specified file(s). Options are:

Keyword	Description
TOP_TEN	Displays information about the 10 files that are top candidates for transfer.
ALL	Displays information about all files in the I/O station's output queue.

This parameter is required.

DISPLAY_OPTION or *DO*

Specifies the amount of information to be displayed. Options are:

Keyword	Description
ALL (A)	Displays all items of information for the selected files. See Remarks.
BRIEF (B)	Displays only the following items of information for the selected files: System-supplied file name User-supplied file name File length File owner identification

The default is BRIEF.

OUTPUT or *O*

Specifies the name of the output file that the status information is written to and, optionally, specifies how the file is to be positioned prior to use. Refer to File Reference in the NOS/VE System Usage manual for a description of file positioning prior to use.

The default is file $OUTPUT.

Remarks

- The DISS utility only displays status information about files that you own that are in the I/O station output queue. you receive no display if you have no files in the queue.
- The display includes the following items of information for each file selected from the output queue:
  - System-supplied file name
  - Copies requested
  - Destination name (I/O station name)
  - Explicit device or alias name
  - Device type (printer/reader/plotter/punch)

- External device characteristic strings
- Family name of generating job
- File length in bytes
- Forms code strings
- Data mode (coded/transparent)
- Output state
- page format (continuous/burstable/nonburstable)
- Page length
- Page width
- Current position in queue
- Priority
- System-supplied job name
- Date and time the file was queued
- User name of generating job
- User-supplied file name
- User-supplied job name
- Vertical print density (SIX/EIGHT/NONE)
- VFU load procedure name

- Each file owner is responsible for establishing file attributes via the SET_FILE_ATTRIBUTES command (see the NOS/VE System Usage manual for more information). This is not a function of an I/O station operator.

Examples The following example displays the full status for file ABC in the I/O station's output queue as requested by a batch device user (the file owner) under the DISS utility.

```
diss/display_station_queue_entry n=all
System_Supplied_File_Name      : $0830_0631_AAA_0197
  Copies                       : 1
  Destination_Name             : URI
  Device_Name                  : PRINT1
  Device-Type                  : printer
  External_Characteristics     : NORMAL
  Family_Name                  : NVE
  File_Length                  : 16080
  Forms_Code                   : NORMAL
  Output_Data_Mode             : coded
  Page_Format                  : burstable
  Page_Length                  : 60
  Page_Width                   : 132
  Position_In_Queue            : 1
  Priority                     : 116
  System_Supplied_Job_Name     : $0830_0631_AAA_0195
  Time_Enqueued                : yyyy-mm-dd hh:mm:ss
  User_Name                    : LC
  User_Supplied_File_Name      : ABC
  User_Supplied_Job_Name       : LC
  Vertical_Print_Density       : six
diss/
```


DISPLAY_STATION_QUEUE_STATUS DISS Subcommand

Purpose Displays the status of the queue of output files destined for the I/O station.

Format **DISPLAY_STATION_QUEUE_STATUS** or **DISSQS**

DISPLAY_OPTION=keyword
OUTPUT=file
STATUS=status variable

Parameters *DISPLAY_OPTION* or *DO*

Specifies the amount of information to be displayed. Options are:

Keyword	Description
ALL (A)	Displays all categories of information for the I/O station's output queue. See Remarks.
BRIEF (B)	Displays all output queues.

The default is BRIEF

OUTPUT or *O*

Specifies the name of the output file where the status information is displayed and, optionally, specifies how the file is to be positioned prior to use. Refer to File Reference in the NOS/VE System usage manual for a description of file positioning prior to use.

The default is file $OUTPUT.

Remarks The following categories of information are displayed when DO=ALL:

- Number of files in the queue
- Each destination name (I/O station name) and the number of files queued for each destination name
- Explicitly requested device names in the queue and the number of files queued for each device
- Each device type and the number of files queued for each device type
- Each of the external device characteristic strings in the queue and the number of files queued for each string
- Each of the forms code strings in the queue and the number of files queued for each string

Examples The following example displays a brief status of all categories of information for the I/O station's output queue.

```
diss/display_station_queue_status do=brief
Station_Name                    : URI
Number_Of_Files                 : 1
  Age_Of_Oldest_File            : 127
  Average_Age_Of_Files          : 127
  File_Count                    : 322
  Total_File_Size               : 16080
diss/
```

DISPLAY_STATION_STATUS DISS Subcommand

Purpose Displays the status of the I/O station that you are operating.

Format **DISPLAY_STATION_STATUS** or **DISSS**

OUTPUT=file
STATUS=status variable

Parameters *OUTPUT* or *O*

Specifies the name of the output file where the status information is displayed and, optionally, specifies how the file is to be positioned prior to use. Refer to File Reference in the NOS/VE System Usage manual for a description of file positioning prior to use.

The default is file $OUTPUT.

Remarks

- The following items of information are displayed:
  - I/O station name
  - Control facility name
  - Default job destination
  - Destination unavailable action
  - File acknowledgement requested (yes/no)
  - Number of files queued for this I/O station

- PM message action (printer/display/discard)
- Required operator console name
- I/O station use (public/private)
- Count of devices
- List of devices showing device type, device status, and file transfer status for each printer

Examples The following example displays the status of an I/O station named IOSTATION_30009F0013

```
diss/display_station_status
Station_Name                        : IOSTATION_30009F0013
  Control_Facility_Name             : STATION_CONTROLLER_1
  Destination_Unavailable_Action    : stop input device
  Default_Job_Destination           : NVE
  File_Acknowledgement              : no
  Number_Of_Files_Queued            : 8
  PM_Message_Action                 : print
  Required_Console_Device           : $CONSOLE_30009F_7000000000
  Station_Usage                     : PRIVATE
  Count_Of_Devices                  : 2

  Device_Name           Type        Device_Status  File_Status
  ------------------------------------------------------------
  CR1                   reader      active         idle
  PRINT2                printer     stopped        suspended
diss/
```

QUIT
DISS Subcommand

Purpose Ends operator control of an I/O station and terminates the current execution of the Display Station utility.

Format

QUIT or
QUI or
END
STATUS=status variable

EDIT_CATALOG 16

EDIT_CATALOG Command

Purpose Initiates the EDIT_CATALOG (EDIC) utility. You can use this full screen application to create, move, copy, print, view, edit, and execute files.

Format

EDIT_CATALOG or
EDIC
CATALOG=file
DISPLAY_OPTIONS=keyword
KEEP_DISPLAY_CURRENT=boolean
NO_DOLLAR_FILES=boolean
STATUS=status variable

Parameters

CATALOG or *C*

Specifies the catalog to display. The default is the working catalog.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Specifies the file information to display. Specify one of the following values:

ALL or A

Displays all file attributes.

BRIEF or B

Displays only the name and entry type (file or catalog).

The default is BRIEF.

KEEP_DISPLAY_CURRENT or *KDC*

Specifies whether to update the catalog display after every EDIT_CATALOG operation, including carriage returns. Use this parameter only when a catalog is being accessed by multiple tasks or jobs. When a single task is accessing a catalog, the catalog display reflects the current state of the catalog, regardless of this parameter's value. Specify one of the following values:

TRUE

Updates the catalog display after the execution of every EDIT_CATALOG operation, carriage return, and home line command.

FALSE

Updates the catalog display only after the execution of home line commands and certain EDIT_CATALOG operations.

The default is FALSE.

NO_DOLLAR_FILES or *NDF*

Specifies whether to omit file names containing a $ character from the display. (By convention, $ appears only in CDC-defined file names.) Specify a boolean value. The default is FALSE.

Remarks

To access the EDIT_CATALOG online manual, enter the following command either at the NOS/VE system prompt or from the EDIC home line:

```
/help manual=edit_catalog
```

You can also access the online manual during an EDIC session by pressing HELP.

Examples

- The following example produces a full display of the $USER catalog:

```
/edit_catalog catalog=$user display_option=all
```

- The following example displays the default working catalog, excluding file names containing a $ character:

```
/edit_catalog no_dollar_files=true
```

$CURRENT_FILE
EDIC Function

Purpose Specifies the current file. This function can be used instead of explicitly naming a file within an SCL command you enter from within EDIT_CATALOG. The current file is considered to be the file at which the cursor was positioned before you pressed HOME. If you use this function within an SCL command and did not previously position the cursor on a file name, an error occurs.

Format **$CURRENT_FILE** or
$CF

Parameters None.

Remarks

- Evaluation of the $CURRENT_FILE function must occur within EDIT_CATALOG. Consequently, if you enter a command which initiates the execution of another task and specify the $CURRENT_FILE as a parameter, the $CURRENT_FILE will not be evaluated and you will receive an error message regarding the filename $CURRENT_FILE.

  If you execute a procedure and specify $CURRENT_FILE as one of the parameters, the $CURRENT_FILE will be evaluated.

- For more information, see the NOS/VE System Usage manual.

SET_DISPLAY_OPTION
EDIC Subcommand

Purpose Specifies the display option you wish to see while using the EDIT_CATALOG command.

Format **SET_DISPLAY_OPTION** or
SET_DISPLAY_OPTIONS or
SETDO
DISPLAY_OPTIONS=keyword
KEEP_DISPLAY_CURRENT=boolean
STATUS=status variable

Parameters *DISPLAY_OPTIONS* or *DISPLAY_OPTION* or *DO*

Specifies the amount of information that you want to have displayed. The following are possible entries:

BRIEF (B)

Selects a display in which file and catalog names are displayed. The display also includes the notation (catalog) indicating that a displayed name is a catalog.

ALL (A)

Selects a display showing all file and catalog information.

If the DISPLAY_OPTION parameter is omitted, BRIEF is used.

KEEP_DISPLAY_CURRENT or *KDC*

Specifies whether the catalog display is to be updated after every EDIT_CATALOG operation, including carriage returns. This parameter only has significance when a catalog is being accessed by multiple tasks or jobs. In the situation where a single task is accessing a catalog, the catalog display will always reflect the current state of the catalog, regardless of this parameter's value. Options are:

TRUE

Catalog display is updated after the execution of every EDIT_CATALOG operation, carriage return, and home line command.

FALSE

Catalog display is updated only after the execution of home line commands and some EDIT_CATALOG operations.

Omission causes FALSE to be used.

Remarks For more information, see the NOS/VE System Usage manual.

SET_SCREEN_OPTION EDIC Subcommand

Purpose Specifies the number of rows of keys to be displayed at the bottom of your screen with the SET_SCREEN_OPTION subcommand.

Format

SET_SCREEN_OPTION or
SET_SCREEN_OPTIONS or
SETSO
MENU_ROWS=integer
STATUS=status variable

Parameters *MENU_ROWS* or *MR*

Specifies the number of rows of keys to be displayed. You may specify a value of zero, one, or two. If omitted, a value of one is assumed.

Remarks For more information, see the NOS/VE System Usage manual.

EDIT_DECK 17

EDIT_DECK
EDID Subcommand

Purpose Opens the specified deck in the working library for editing while maintaining your current position in other decks.

Format

EDIT_DECK or
EDID
DECK=name
STATUS=status variable

Parameters

DECK or **D**

Specifies the deck to be edited. If the deck does not exist, it is created.

This parameter is required.

Remarks

- To discard decks created unintentionally, enter:

```
end_deck write_deck=false
```

- The maximum deck size is 16,777,214 lines.
- For more information, see the NOS/VE File Editor manual.

EDIT_FIRST_DECK
EDID Subcommand

Purpose Opens the first deck on the working library for editing while maintaining your current position in other decks.

Format

EDIT_FIRST_DECK or
EDIFD
STATUS=status variable

Remarks

- Decks are always in alphabetical order in the working library.
- For more information, see the NOS/VE File Editor manual.

EDIT_LAST_DECK
EDID Subcommand

Purpose Opens the last deck in the working library for editing while maintaining your current position in other decks.

Format

EDIT_LAST_DECK or
EDILD
STATUS=status variable

Remarks

- Decks are always in alphabetical order in the working library.
- For more information, see the NOS/VE File Editor manual.

EDIT_NEXT_DECK
EDID Subcommand

Purpose Opens the next deck on the working library for editing while maintaining your current position in other decks.

Format

EDIT_NEXT_DECK or
EDIND
STATUS=status variable

Remarks

- Decks are always in alphabetical order in the working library.
- For more information, see the NOS/VE File Editor manual.

END_DECK
EDID Subcommand

Purpose Closes editing on the current deck.

Format

END_DECK or
ENDD
WRITE_DECK=boolean
STATUS=status variable

Parameters *WRITE_DECK* or *WD* or *WRITE_FILE* or *WF*

Specifies whether the changes made to the deck since it was opened for editing are to be written to the working library.

TRUE indicates that the deck is to be rewritten.

FALSE indicates that the deck remains unchanged (the edited copy is discarded). FALSE also discards a deck that has been created during the current editing session provided that you have not closed the deck. This is the easiest way to delete decks that were unintentionally created.

If omitted, TRUE is assumed and the results are written to the working library.

Remarks For more information, see the NOS/VE File Editor manual.

RESET_DECK
EDID Subcommand

Purpose Discards changes made to the current deck being edited. All changes made since the last time the deck was opened for editing are discarded. The editor obtains a new copy of the deck from the working library.

Format

RESET_DECK or
RESD
STATUS=status variable

Remarks For more information, see the NOS/VE File Editor manual.

SELECT_DECK
EDID Subcommand

Purpose Opens the specified deck on the working library for editing and closes the previous deck (if any).

Format

SELECT_DECK or
SELD
DECK=name
STATUS=status variable

Parameters **DECK or D**

Specifies the name of the deck to be edited. If the deck does not exist, it is created.

This parameter is required.

Remarks

- Decks are always in alphabetical order in the working library.
- For more information, see the NOS/VE File Editor manual.

SELECT_FIRST_DECK EDID Subcommand

Purpose Opens the first deck on the working library for editing and closes the previous deck (if any).

Format

SELECT_FIRST_DECK or
SELFD
STATUS=status variable

Remarks

- Decks are always in alphabetical order in the working library.
- For more information, see the NOS/VE File Editor manual.

SELECT_LAST_DECK EDID Subcommand

Purpose Opens the last deck on the working library for editing and closes the previous deck (if any).

Format

SELECT_LAST_DECK or
SELLD
STATUS=status variable

Remarks

- Decks are always in alphabetical order in the working library.
- For more information, see the NOS/VE File Editor manual.

SELECT_NEXT_DECK
EDID Subcommand

Purpose Opens the next deck on the working library for editing and closes the previous deck (if any).

Format

SELECT_NEXT_DECK or
SELND
STATUS=status variable

Remarks

- Decks are always in alphabetical order in the working library.
- For more information, see the NOS/VE File Editor manual.

EDIT_FILE 18

EDIT_FILE Command

Purpose Starts a file editor (EDIT_FILE utility) session.

Format

EDIT_FILE or
EDIF
FILE=file
INPUT=file
OUTPUT=file
PROLOG=file
DISPLAY_UNPRINTABLE_CHARACTERS=boolean
STATUS=status variable

Parameters

FILE or **F**

Specifies the name of the file you want to edit. If the file you specify does not exist, a new file is created.

The file cannot be an object file.

This parameter is required.

INPUT or *I*

Specifies the file to be used as input to the editor. This file can be positioned. This file contains optional editor subcommands used to manipulate the working file. If omitted, $COMMAND is assumed.

OUTPUT or *O*

Specifies the file to which you want to write any output that may result from your editing session. This file can be positioned.

If OUTPUT is omitted, $OUTPUT is assumed. File $OUTPUT is usually connected to the terminal.

PROLOG or *P*

File the system executes when you start an editing session. If PROLOG is omitted, file $USER.SCU_EDITOR_PROLOG is used. You can establish a different default prolog file by using the CREATE_DEFAULT_VARIABLE command to set the variable ESD$EDIF_PROLOG to the file you want to be your default prolog.

For more information on the CREATE_DEFAULT_VARIABLE command, refer to the NOS/VE System Usage manual.

DISPLAY_UNPRINTABLE_CHARACTERS or *DUC*

Specifies whether unprintable ASCII characters are replaced by mnemonics when the file is displayed at the terminal. Options are:

TRUE

Unprintable characters (ASCII values 127 and 0 through 31) are replaced by their respective mnemonic values enclosed within the less than and greater than characters, < >. The mnemonics are replaced by the ASCII characters when the file is replaced.

FALSE

Unprintable characters are replaced by a single space and a warning message is issued. If the file is written when you exit the editing session, the unprintable characters are replaced by spaces.

If DISPLAY_UNPRINTABLE_CHARACTERS is omitted, FALSE is used.

ASCII characters and their corresponding mnemonic values are listed in appendix C.

Remarks

- If you would like to specify a file containing editor subcommands to be executed when you leave the editor (an epilog file), use the SET_EPILOG subcommand. If you want this done each time, include the SET_EPILOG subcommand in the file you specify for the PROLOG parameter.
- The following prompt appears for line editing:

```
ef/
```

- To edit a second file while in the editor, enter the EDIT_FILE subcommand. The FILE and STATUS parameters are the only parameters allowed on the EDIT_FILE subcommand.
- Commands following the EDIT_FILE command on the same physical line are processed in the context of the editor.

- The maximum file size is 16,777,214 lines.
- For more information, see the NOS/VE File Editor manual.

Examples

- The following command starts the EDIT_FILE utility with file $USER.MY_FILE:

```
edit_file file=$user.my_file
```

- The following command calls the editor for file ALPHA. The screen mode user will seen an initial display with two data splits, the first being 16 lines long:

```
edif f=alpha; setso s=2 ss=16
```

ACTIVATE_SCREEN EDIF Subcommand

Purpose Activates screen mode; specifies terminal type.

Format

ACTIVATE_SCREEN or
ACTS
MODEL=name
STATUS=status variable

Parameters *MODEL* or *M*

Specifies the type of terminal you are using. Valid entries are:

Entry	Terminal
MAC_CONNECT_10	Apple Macintosh running version 1.0 or 1.0+ of Control Data CONNECT for the Macintosh
MAC_CONNECT_11	Apple Macintosh running version 1.1 of Control Data CONNECT for the Macintosh
MAC_CONNECT_20	Apple Macintosh running version 2.0 of Control Data CONNECT for the Macintosh

MAC_CONNECT_21	Apple Macintosh running version 2.1 of Control Data CONNECT for the Macintosh
MAC_CONNECT_22	Apple Macintosh running version 2.2 of Control Data CONNECT for the Macintosh
PC_CONNECT_10	IBM PC or equivalent running version 1.0 of Control Data CONNECT for the IBM PC
PC_CONNECT_11	IBM PC or equivalent running version 1.1 of Control Data CONNECT for the IBM PC
PC_CONNECT_12	IBM PC or equivalent running version 1.2 of Control Data CONNECT for the IBM PC
PC_CONNECT_13	IBM PC or equivalent running version 1.3 or 1.4 of Control Data CONNECT for the IBM PC
PC_CONNECT_20	IBM PC or equivalent running version 2.0 of Control Data CONNECT for the IBM PC
IBM_3270	IBM 3270 with 24 x 80 screen
IBM_3270_2	IBM 3270 with 24 x 80 screen
IBM_3270_3	IBM 3270 with 32 x 80 screen
IBM_3270_4	IBM 3270 with 43 x 80 screen
IBM_3270_5	IBM 3270 with 27 x 132 screen
DEC_VT100	Digital Equipment VT100 with 18 function keys
DEC_VT100_GOLD	Digital Equipment VT100 with 32 function keys
DEC_VT220	Digital Equipment VT220 for users logging in through CDCNET

TV_955	Televideo 955 with full editing capability
TV_955_PROTECTED	Televideo 955 with form entry access only

If the terminal you are using is not on this list, use the DISPLAY_TERMINAL_MODEL TERMINAL_MODEL=ALL subcommand.

You must specify the MODEL parameter either on an earlier ACTIVATE_SCREEN or SET_SCREEN_OPTIONS subcommand or on the TERMINAL_MODEL parameter of the CHANGE_TERMINAL_ATTRIBUTES command.

Remarks

- The recommended method for preparing your session for screen editing is to enter the CHANGE_TERMINAL_ATTRIBUTES and CHANGE_INTERACTION_STYLE commands, described in the NOS/VE System Usage manual, before you start an editing session. To do this automatically, include these commands in your user prolog.
- Inside procedures, you can use the ACTIVATE_SCREEN subcommand to allow the user of the procedure to enter editor subcommands.
- Executing this subcommand causes the firmware of some terminals to be reinitialized. Refer to your terminal's documentation for more information.
- Use the $SCREEN_ACTIVE function to determine whether screen mode is active.
- For more information, see the NOS/VE File Editor manual.

Examples

To switch from line mode to screen mode in an editing session, enter:

```
activate_screen
```

$ACTIVE_IDENTIFIER
EDIF Function

Purpose Returns a value of line type identifier unless a string is expected in the context of the function call. The value identifies the nearest active line to the line given as the argument.

Format

$ACTIVE_IDENTIFIER or
$AI
(LINE_IDENTIFIER: line_identifier or **string)**

Parameters

LINE_IDENTIFIER

Identifies the line for which you want to find the status. If the line you specify is active, the same line identifier or string is returned. If the line is not active, the line identifier for the nearest active line is returned. If no lines are active, FIRST is returned.

This parameter is required.

Remarks For more information, see the NOS/VE File Editor manual.

ALIGN_SCREEN
EDIF Subcommand

Purpose Enables you to change the alignment of your screen.

Format

ALIGN_SCREEN or
ALIS or
A
MIDDLE=keyword or *integer* or *line_identifier*
TOP=keyword or *integer* or *line_identifier*
BOTTOM=keyword or *integer* or *line_identifier*
OFFSET=integer
STATUS=status variable

Parameters

MIDDLE or *M*

Specifies a line to be centered vertically on the screen. Values can be an integer, line identifier, or one of the keywords: CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. You cannot use this parameter with the TOP and BOTTOM parameters.

If you omit this parameter, CURRENT is assumed.

TOP or *T*

Specifies a line to be positioned at the top of the screen. The resulting middle line of the screen becomes the current line. Values can be an integer, line identifier, or one of the keywords: CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. You cannot use this parameter with the MIDDLE and BOTTOM parameters.

If you omit this parameter, no value is supplied.

BOTTOM or *B*

Specifies a line to appear at the bottom of the screen. The resulting middle line of the screen becomes the current line. Values can be an integer, line identifier, or one of the keywords: CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. You cannot use this parameter with the TOP and MIDDLE parameters.

If you omit this parameter, no value is supplied.

OFFSET or *O*

Specifies the number of columns to offset your view of the file on the screen. The number can be an integer from 0 through 216. The number you specify is added to column 1 and the last column displayed. For example, if the rightmost column is 80 and you specify an OFFSET value of 20, the leftmost column becomes 21 and the rightmost column becomes 100. The offset value is displayed on the title line of your screen when it is a non-zero value.

Remarks

- You can use $OFFSET to return the current OFFSET value.
- For more information, see the NOS/VE File Editor manual.

Examples

- The following example moves the current line to the bottom of the screen (same as the LinDn operation):

```
align_screen bottom=current
```

- The following example displays column 51 as the leftmost column:

```
alis o=50
```

BREAK_TEXT
EDIF Subcommand

Purpose Breaks a line at a specific point in the line to make one line into two lines.

Format

BREAK_TEXT or
BRET or
B
LINES=keyword or *integer* or *line_identifier*
COLUMN=keyword or *integer*
STATUS=status variable

Parameters

LINES or *LINE* or *L*

Identifies the line to be broken. Values can be an integer, line identifier, or one of the keywords: CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. Ranges are not allowed.

If omitted, CURRENT is assumed.

COLUMN or *C*

Specifies the column before which the break is to occur. In other words, the break occurs just before the column specified. Values can be an integer from 1 through 256, or one of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MAXIMUM.

If omitted, CURRENT is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

CENTER_LINES
EDIF Subcommand

Purpose Centers a line or lines between margins set with the SET_ PARAGRAPH_MARGINS subcommand.

Format

CENTER_LINES or
CENTER_LINE or
CENL
NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
STATUS=status variable

Parameters *NUMBER* or *N*

Specifies the number of lines to be centered.

If you omit this parameter and specify a range for the LINE parameter, NUMBER assumes a value of ALL.

If you omit this parameter without specifying a range of lines, NUMBER assumes a value of 1.

If NUMBER and LINES are both omitted, CURRENT is assumed.

LINES or *LINE* or *L*

Specifies a range of lines to be centered.

If one line is specified, the centering is limited to that line. Values can be an integer, line identifier, or one of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. MARK, SCREEN.

If LINES is omitted, the lines to be centered are determined by the NUMBER parameter. If LINES and NUMBER are both omitted, CURRENT is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following example centers the next five lines.

```
center_lines number=5
```

- The following example centers all lines between lines 15 and 23.

```
cenl 1=15..23
```

CLEAR_TABS
EDIF Subcommand

Purpose Deletes all or some of the tab columns.

Format

CLEAR_TABS or
CLEAR_TAB or
CLET
TAB_COLUMN=keyword or *list of range of integer*
STATUS=status variable

Parameters *TAB_COLUMN* or *TAB_COLUMNS* or *TC*

Specifies the columns to delete as tab columns. Values can be the keyword ALL or a list of a range of integers from 1 through 256.

If TAB_COLUMN is omitted, all tabs are cleared.

Remarks For more information, see the NOS/VE File Editor manual.

Examples The following CLEAR_TAB subcommand clears columns 7 and 65 as tab columns:

```
clear_tab tab_column=(7,65)
```

COPY_TEXT
EDIF Subcommand

Purpose Copies a block of text from one place to another within your working files.

Format

COPY_TEXT or
COPT or
C
TEXT=range of string
NUMBER=keyword or *integer*

LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
COLUMNS=keyword or *range of: keyword* or *integer*
INSERTION_LOCATION=keyword or *integer* or *line_identifier*
INSERTION_COLUMN=keyword or *integer*
PLACEMENT=keyword
BOUNDARY=keyword
UPPER_CASE=boolean
WORD=boolean
REPEAT_SEARCH=boolean
STATUS=status variable

Parameters

TEXT or *T*

Specifies strings of text in the first and last lines of a block of text to be copied. If you enter only one string, the block of text to be copied will contain only one line. If you enter two strings, the search for the second begins immediately after the first is found.

If TEXT is specified, the INSERTION_COLUMN and BOUNDARY parameters are ignored and line boundaries are used.

If omitted, the lines to be copied will be determined by the NUMBER and LINES parameters or the REPEAT_SEARCH parameter.

NUMBER or *N*

Specifies the number of blocks of text to be copied. Values for this parameter can be numbers or the keyword ALL (A).

If omitted and a range is specified for the LINES parameter, NUMBER assumes a value of ALL. Otherwise the assumed value is 1.

LINES or *LINE* or *L*

Specifies the range of lines to be searched for the text to be copied. If a single value is specified, only that line is searched. Values can be an integer, line identifier, or one of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN.

If omitted, CURRENT..LAST is assumed.

COLUMNS or *COLUMN* or *C*

Specifies the range of columns to be searched for text to be copied. The integers can be from 1 through 256 or any of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MARK, MAXIMUM.

If omitted, CURRENT is assumed. If omitted and you have specified LINE=MARK, the marked lines provide the column boundaries. If COLUMN is omitted and you have specified a LINE parameter other than MARK, all columns will be searched.

INSERTION_LOCATION or *IL*

Specifies the line before or after which the text is to be copied (depending on the value of the PLACEMENT parameter). Values can be an integer, line identifier, or one of the LINE keywords: CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. Ranges are not allowed.

If omitted, CURRENT is assumed.

INSERTION_COLUMN or *IC*

Specifies the column before or after which the text is to be copied (depending on the value of the PLACEMENT parameter). Values can be an integer from 1 through 256, or any of the COLUMN keywords: CURRENT, FIRST_MARK, LAST_MARK, MAXIMUM. Ranges are not allowed.

If omitted, CURRENT is assumed.

If a value for TEXT is specified, INSERTION_COLUMN is ignored.

PLACEMENT or *P*

Specifies if the copied lines are to appear BEFORE (B) or AFTER (A) the location specified by the INSERTION_LOCATION parameter.

If omitted, AFTER is assumed.

BOUNDARY or *B*

Specifies the type of boundary that will limit the search. Values can be BOX, LINE, or STREAM.

If BOUNDARY and COLUMNS are both omitted, LINE is assumed.

If BOUNDARY is omitted but COLUMNS is specified, STREAM is assumed.

If a value for TEXT is specified, BOUNDARY is ignored; line boundaries are used.

UPPER_CASE or *UC*

Determines the significance of capitalization in the search.

If you specify TRUE, the editor searches the file as if it were all uppercase.

If you specify FALSE, the editor searches for the text exactly as it was entered.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for UPPER_CASE is used.

WORD or *W*

Determines whether the editor searches for the specified text string as a word (the text you want to search for is surrounded by nonalphanumeric characters).

If you specify TRUE, the editor searches for the text as a word. If you specify FALSE, it doesn't.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for WORD is used.

REPEAT_SEARCH or *RS*

Instructs the editor how to use the values entered for the last TEXT, UPPER_CASE, and WORD parameters.

If you specify TRUE, the editor uses the same TEXT, UPPER_CASE, and WORD parameters as the last time you entered them for any subcommand (unless you have specified values for this subcommand).

If you specify FALSE, the editor uses the parameters entered with the current subcommand.

If omitted, FALSE is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following copies lines 30 through 40 to immediately after the current line:

```
copy_text line=30..40
```

- The following copies the next occurrence of a block of text beginning with the line containing *one* and ending with the line containing *five* to immediately before line 71:

```
copt t='one'..'five' il=71 p=b
```

- The following inserts a box of characters before column 5 of lines starting with line 5:

```
copt l=1..3 c=3..5 il=5 ic=5 b=box
```

Before copy:

```
abcdefghijklmnop
 abcdefghijklmnop
  abcdefghijklmnop
   abcdefghijklmnop
12345678901234567890
12345678901234567890
12345678901234567890
12345678901234567890
```

After copy:

```
abcdefghijklmnop
 abcdefghijklmnop
  abcdefghijklmnop
   abcdefghijklmnop
1234cde5678901234567890
1234bcd5678901234567890
1234abc5678901234567890
12345678901234567890
```

$CURRENT_COLUMN EDIF Function

Purpose Returns the value of the current column number.

Format $CURRENT_COLUMN or
$CC

Parameters None.

Remarks

- If the POSITION_CURSOR subcommand is used to specify a column on a row that is not part of the file text, the value returned is the column at which the cursor was positioned before the POSITION_CURSOR subcommand was entered.
- For more information, see the NOS/VE File Editor manual.

$CURRENT_DECK_NAME EDIF Function

Purpose Returns the name of the current deck (for editing decks only).

Format **$CURRENT_DECK_NAME** or
$CDN

Parameters None.

Remarks

- You can also use the $CURRENT_OBJECT function for this purpose.
- For more information, see the NOS/VE File Editor manual.

$CURRENT_LINE EDIF Function

Purpose Returns an integer specifying the current line number.

Format **$CURRENT_LINE** or
$CL

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

Examples The following statements assign variable LAST_LINE an integer value reflecting the number of lines in a file or deck:

```
position_cursor l=last
last_line = $current_line
```

$CURRENT_OBJECT EDIF Function

Purpose Returns a string identifying the current file name or deck name.

Format **$CURRENT_OBJECT** or
$CO

Parameters None.

Remarks

- You can use the $CURRENT_OBJECT_TYPE function to determine if the string is a file name or a deck name.
- For more information, see the NOS/VE File Editor manual.

Examples The following procedure rewrites the current file, or if you are editing a deck or a local file, the procedure places a copy of the deck or file in the catalog $USER.SAVE_EDITOR_FILES. This catalog must be present in your $USER catalog.

```
PROCEDURE checkpoint_file, chef (
  status)

  current_file=$fname($co)
  IF $file(current_file, permanent) THEN
    write_file file=current_file
  ELSE
    write_file file=$user.save_editor_files..
//$name($path(current_file, last))
  IFEND

PROCEND checkpoint_file
```

$CURRENT_OBJECT_TYPE
EDIF Function

Purpose Returns a string identifying the current object being edited. Possible values are FILE, DECK, or NULL.

Format **$CURRENT_OBJECT_TYPE** or
$COT

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

$CURRENT_ROW
EDIF Function

Purpose Returns an integer identifying the row on the screen where the cursor is positioned (as opposed to the current line number of a file).

Format **$CURRENT_ROW** or
$CR

Parameters None.

Remarks

- Zero is returned if the current row is not within screen boundaries or if you are in line mode.
- For more information, see the NOS/VE File Editor manual.

$CURRENT_SPLIT
EDIF Function

Purpose Returns an integer specifying the split of the screen in which the cursor is positioned.

Format **$CURRENT_SPLIT** or
$CS

Parameters None.

Remarks

- If you are in line mode, zero is returned.
- Values returned can be from 1 through 16. The top split of the screen is 1, the next lower is 2, and so on.
- For more information, see the NOS/VE File Editor manual.

$CURRENT_WORD EDIF Function

Purpose Returns the current word as a string.

Format **$CURRENT_WORD** or **$CW**

Parameters None.

Remarks

- This function is particularly useful when supplied as the value for the TEXT parameter for the LOCATE_TEXT and REPLACE_TEXT subcommands.
- For more information, see the NOS/VE File Editor manual.

Examples

- The following example converts the characters in the current word to lowercase:

```
replace_text t=$cw ..
   nt=$translate(upper_to_lower,$cw)
```

- If the current word is a deck name, you can edit that deck by entering the following command:

```
edit_deck d=$name($cw)
```

- If the current word is a command name, you can display information about that command by entering the following command:

```
include_command ..
    c='display_command_information c='//$cw
```

$CURRENT_WORD_COLUMN
EDIF Function

Purpose Returns an integer specifying the column in which the current word begins.

Format **$CURRENT_WORD_COLUMN** or
$CWC

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

Examples The following marks the current word:

```
mark_character c=$cwc..$strlen($cw)+$cwc-1
```

DEACTIVATE_SCREEN
EDIF Subcommand

Purpose Stops screen mode without stopping the editor.

Format **DEACTIVATE_SCREEN** or
DEAS
STATUS=status variable

Remarks

- When you enter this subcommand, the screen is cleared and the line mode prompt appears:

  ```
  ef/
  ```

- Use DEACTIVATE_SCREEN to enter commands that must be continued over more than one input line.
- For more information, see the NOS/VE File Editor manual.

DELETE_CHARACTERS EDIF Subcommand

Purpose Enables you to delete characters.

Format

DELETE_CHARACTERS or
DELC or
DELETE_CHARACTER or
DC

NUMBER=keyword or *integer*
LINES=keyword or *integer* or *line_identifier*
COLUMNS=range of: keyword or *integer*
STATUS=status variable

Parameters

NUMBER or *N*

Specifies the number of characters to be deleted. Values may be an integer or the keyword ALL.

If you omit this parameter without specifying LINE or COLUMN, a value of 1 is assumed.

If you omit this parameter and specify a range for COLUMN, ALL is assumed.

LINES or *LINE* or *L*

Specifies the line in which characters will be deleted. Values can be an integer, line identifier, or any of the keywords: CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. Ranges are not allowed.

If omitted, CURRENT is assumed.

COLUMNS or *COLUMN* or *C*

Specifies the columns to be deleted within the specified line(s). Values can be an integer from 1 through 256, or one of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MAXIMUM.

If omitted, CURRENT is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples The following deletes the characters in columns 1 through 17 of the current line.

```
delete_characters columns=(1..17)
```

DELETE_EMPTY_LINES EDIF Subcommand

Purpose Deletes a block of blank lines until a nonblank line is encountered.

Format

DELETE_EMPTY_LINES or
DELETE_EMPTY_LINE or
DELEL
LINES=keyword or *integer* or *line_identifier*
STATUS=status variable

Parameters *LINES* or *LINE* or *L*

Specifies the line at which the deletion of blank lines is to begin. Values can be an integer, line identifier, or any of the keywords: CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. Ranges are not allowed.

If the line you specify is not a blank line, nothing happens.

If LINES is omitted, CURRENT is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

DELETE_LINES EDIF Subcommand

Purpose Enables you to delete a line or range of lines.

Format

DELETE_LINES or
DELETE_LINE or
DELL
TEXT=range of string
NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
UPPER_CASE=boolean
WORD=boolean
REPEAT_SEARCH=boolean
STATUS=status variable

Parameters *TEXT* or *T*

Specifies a block of text to be deleted, beginning with the line containing the first string and ending with the line containing the second string.

If TEXT is omitted, the editor does not supply a value and the NUMBER and LINE parameters determine which text will be deleted.

NUMBER or *N*

Specifies the number of lines to be deleted, or the number of blocks of text to be deleted depending on the values you specify for the LINES and TEXT parameter. Values can be an integer or the keyword ALL.

If you omit this parameter and specify a range for the LINE parameter, NUMBER assumes a value of ALL.

If you omit this parameter without specifying a range of lines, NUMBER assumes a value of 1.

LINES or *LINE* or *L*

Specifies a range of lines to be deleted. Values can be an integer, line identifier, or any of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN.

If a single integer or keyword is specified, only that line is deleted.

If LINE=MARK is specified, marked lines are deleted in their entirety (even if the boundary implied by the mark is STREAM).

If LINE is omitted, CURRENT..LAST is assumed.

UPPER_CASE or *UC*

Determines the significance of capitalization in the search.

If you specify TRUE, the editor searches the file as if it were all uppercase.

If you specify FALSE, the editor searches for the text exactly as it was entered.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for UPPER_CASE is used.

WORD or *W*

Determines whether the editor searches for the specified text string as a word (the text you want to search for is surrounded by nonalphanumeric characters).

If you specify TRUE, the editor searches for the text as a word. If you specify FALSE, it doesn't.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for WORD is used.

REPEAT_SEARCH or *RS*

Instructs the editor how to use the values entered for the last TEXT, UPPER_CASE, and WORD parameters.

If you specify TRUE, the editor uses the same TEXT, UPPER_CASE, and WORD parameters as the last time you entered them for any subcommand (unless you have specified values for this subcommand).

If you specify FALSE, the editor uses the parameters entered with the current subcommand.

If omitted, FALSE is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples The following subcommand deletes marked lines.

```
delete_lines line=mark
```

DELETE_TEXT
EDIF Subcommand

Purpose Deletes text delimited by the subcommand parameters.

Format

DELETE_TEXT or
DELT or
D
TEXT=range of string
NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
COLUMNS=keyword or *range of: keyword* or *integer*

BOUNDARY=keyword
UPPER_CASE=boolean
WORD=boolean
REPEAT_SEARCH=boolean
STATUS=status variable

Parameters *TEXT* or *T*

Specifies strings of text in the first and last lines of a block of text to be deleted. If you enter only one string, the block of text to be deleted will contain only one line. If you enter two strings, the search for the second begins immediately after the first is found.

If omitted, the lines to be deleted will be determined by the NUMBER, BOUNDARY, COLUMNS, and LINES parameters or the REPEAT_SEARCH parameter.

NUMBER or *N*

Specifies the number of lines to be deleted. Values can be numbers or the keyword ALL.

If you omit this parameter and specify a range for the LINE parameter, NUMBER assumes a value of ALL.

If you omit this parameter without specifying a range of lines, NUMBER assumes a value of 1.

LINES or *LINE* or *L*

Specifies a range of lines to be deleted.

If a single value is specified, only that line is deleted.

If omitted, CURRENT..LAST is assumed.

COLUMNS or *COLUMN* or *C*

Specifies the columns to be deleted in a specified line.

If a range of lines is specified in the LINES parameter, whole lines are deleted between the first and last lines.

If no text is specified and the REPEAT_SEARCH parameter is FALSE, the specified columns are deleted. If text is specified or the REPEAT_SEARCH parameter is TRUE, whole lines are deleted.

If omitted, the entire line is deleted.

BOUNDARY or *B*

Specifies the type of boundary that will limit the search. Values can be BOX, LINE, or STREAM. If BOUNDARY is omitted, LINE is assumed.

If COLUMN is specified, STREAM is assumed.

UPPER_CASE or *UC*

Determines the significance of capitalization in the search.

If you specify TRUE, the editor searches the file as if it were all uppercase.

If you specify FALSE, the editor searches for the text exactly as it was entered.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for UPPER_CASE is used.

WORD or *W*

Determines whether the editor searches for the specified text string as a word (the text you want to search for is surrounded by nonalphanumeric characters).

If you specify TRUE, the editor searches for the text as a word. If you specify FALSE, it doesn't.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for WORD is used.

REPEAT_SEARCH or *RS*

Instructs the editor on how to use the values entered for the last TEXT, UPPER_CASE, and WORD parameters.

TRUE instructs the editor to use the same TEXT, UPPER_CASE, and WORD parameters as the last time you entered them on any subcommand, unless you have specified values for them on this subcommand.

FALSE instructs the editor to use the parameters entered with the current DELETE_TEXT subcommand.

If omitted, FALSE is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following deletes all lines from the line containing *first* to the line containing *last*:

```
delete_text text='first'..'last'
```

- The following deletes a box of characters in lines 1 through 3 and columns 3 through 5:

```
delt l=1..3 c=3..5 b=box
```

Text before deletion:

```
abcdefghijklmnop
 abcdefghijklmnop
  abcdefghijklmnop
   abcdefghijklmnop
```

Text after deletion:

```
abfghijklmnop
 aefghijklmnop
  defghijklmnop
   abcdefghijklmnop
```

DELETE_WORD EDIF Subcommand

Purpose Deletes words, blanks, or characters, depending on the position in the file you specify.

Format

DELETE_WORD or
DELW or
DW
LINES=keyword or *integer* or *line_identifier*
COLUMN=keyword or *integer*
STATUS=status variable

Parameters

LINES or *LINE* or *L*

Specifies a line in which the deletion is to occur. Values can be an integer, line identifier, or any of the keywords: CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. Ranges are not allowed.

If omitted, CURRENT is assumed.

COLUMN or *C*

Specifies the column to begin the deletion. Values can be an integer or any of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MAXIMUM. Ranges are not allowed.

If omitted, CURRENT is assumed.

Remarks

- For the editor, a word is a string of letters, numbers, or the special characters $, #, @, and _, surrounded by any other characters. The end of a line or beginning of a line is also considered a word boundary.
- If you specify a position that is part of a word, the entire word is deleted.
- If you specify a position that is a blank character, it and all following blanks are deleted.
- If you specify a position that is not part of a word or a blank character, only that character is deleted.
- For more information, see the NOS/VE File Editor manual.

Examples The following deletes the first word in line 50:

```
delete_word line=50 column=1
```

DISPLAY_COLUMN_NUMBERS EDIF Subcommand

Purpose Enables you to list column numbers, which will temporarily overwrite the specified line.

Format

DISPLAY_COLUMN_NUMBERS or
DISCN
ROW=integer
STATUS=status variable

Parameters *ROW* or *ROWS* or *R*

Specifies which row on the screen is to show the column numbers.

If omitted, the column numbers temporarily overwrite the current line.

Remarks

- The column numbers shown correspond to columns in the file and not column numbers on the screen.
- If the offset is currently nonzero, it is set to zero.
- For more information, see the NOS/VE File Editor manual.

Examples In screen mode, to display the column numbers on the third line, position the cursor on the third line, press Home, and enter:

```
display_column_numbers
```

The following appears:

```
FIRST LINE
SECOND LINE
123456789A123456789B123456789C123456789D123 ...
FOURTH LINE
```

DISPLAY_EDITOR_STATUS EDIF Subcommand

Purpose Enables you to check the status of a number of editor variables including the current tab character, tab columns, and function key definitions.

Format

DISPLAY_EDITOR_STATUS or
DISES
OUTPUT=file
STATUS=status variable

Parameters

OUTPUT or *O*

Specifies the file to which the status is written. If omitted, the status is written to the output destination for the session.

Remarks For more information, see the NOS/VE File Editor manual.

Examples The following example displays the first screen of editor status. The user is editing in screen mode using a Digital Equipment VT220 terminal.

```
dises
Press Next/Return for more
 Displaying Editor Status
--------------------------------------------------------------------------------
SCU Editor version is 88193
Modification name is EDIT_FILE
Current file is :NVE.SCU.TEACH
Line width is 0. Search margins are 1 to 256
Set verify option FALSE. State FALSE. No mask character. Tab character is \
Tab columns are:  1 7 72

Function Keys:
  Key       Label       Commands
       F1   Copy        copy_text l=m p=b
 Shift F1   Move        move_text l=m p=b
       F2   Mark        mark_lines
 Shift F2   Unmrk       unmark
       F3   MrkCh       mark_characters
 Shift F3   MrkBx       mark_boxes
       F4   Locate      locate_text t=$si('Enter search string')
 Shift F4   LocNxt      locate_next
       F5   Undo        undo
 Shift F5
```

DISPLAY_POSITION
EDIF Subcommand

Purpose Displays the current line number, current column number, size of the file, and, for screen mode, the line number of the top and bottom line of the screen on the message line.

Format **DISPLAY_POSITION** or
DISP
STATUS=status variable

Remarks For more information, see the NOS/VE File Editor manual.

Examples If, in screen mode, you enter:

```
display_position
```

A display similar to the following appears:

```
Current Line:12 Column:10 Size:109 Top:10 Bottom:18
```

$DISPLAY_UNPRINTABLE_CHARACTERS
EDIF Function

Purpose Returns a boolean value indicating whether unprintable ASCII characters displayed at the terminal are replaced by their corresponding mnemonic values (TRUE) or not (FALSE).

Format **$DISPLAY_UNPRINTABLE_CHARACTERS** or **$DUC**

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

EDIT_FILE
EDIF Subcommand

Purpose Enables you to edit multiple files within the editor.

Format

EDIT_FILE or
EDIF
FILE=**file**
STATUS=*status variable*

Parameters **FILE** or **F**

Specifies the name of the file you want to edit. If the file you specify does not exist, a new file is created.

The file must be a sequential file. By default, files created by NOS/VE have this attribute.

The file cannot be an object file.

This parameter is required.

Remarks

- Unlike the EDIT_FILE command, the EDIT_FILE subcommand does not have INPUT, OUTPUT, or PROLOG parameters. Once you are in the editor, you can specify only another file to edit.
- To edit two files on the same screen, use the SET_SCREEN_OPTIONS and EDIT_FILE subcommands.
- The maximum file size is 16,777,214 lines.

- For more information, see the NOS/VE File Editor manual.

Examples To edit file OMEGA, after editing another file, enter:

```
edit_file file=$user.omega
```

ENABLE_LINES
EDIF Subcommand

Purpose Specifies whether active lines or all lines are accepted as values for the LINES or INSERTION_LOCATION parameters. These parameters are associated with the DELETE_LINES, INSERT_LINES, and REPLACE_LINES subcommands.

Format

ENABLE_LINES or
ENAL
LINES=keyword
STATUS=status variable

Parameters

LINES or **LINE** or **L**

Specifies whether all lines or only active lines associated with a file or deck are accepted when either a DELETE_LINES, INSERT_LINES, or REPLACE_LINES command is entered.

Keyword values for this parameter are ACTIVE and ALL. Unless you change the keyword value by using this command, only active lines are accepted by the editor.

Remarks

- You can use the $LINES_ENABLED function to return the current value of the LINES parameter.
- ENABLE_LINES may be used in the file generated by the SCU EXTRACT_MODIFICATION subcommand. Refer to the SCL Source Code Management manual for more information on the EXTRACT_MODIFICATION subcommand.
- For more information, see the NOS/VE File Editor manual.

Examples Refer to the ENABLE_LINES example in the online Examples manual. You access this manual by entering:

```
help manual=examples
```

QUIT
EDIF Subcommand

Purpose Stops the processing of the current source of command input to the editor.

Format

QUIT or
END or
QUI
WRITE_FILE=boolean
STATUS=status variable

Parameters *WRITE_FILE* or *WD* or *WRITE_DECK* or *WF*

Specifies whether you want changes to all open files made permanent. The initial value is TRUE.

If FALSE is specified, no changes are made.

If omitted, the previous selection remains in effect.

Remarks

- Changes are made permanent after the QUIT command executes.
- Any error that occurs while files are written is reported in the status parameter of the command that invoked the editor.
- For more information, see the NOS/VE File Editor manual.

END_FILE
EDIF Subcommand

Purpose Enables you to close the current file, and continue editing other files.

Format

END_FILE or
ENDF
WRITE_DECK=boolean
STATUS=status variable

Parameters *WRITE_DECK* or *WD* or *WRITE_FILE* or *WF*

Specifies whether to make changes to the current file permanent.

If FALSE is specified, changes are not made permanent.

If omitted, TRUE is assumed.

Remarks

- The END_FILE and END_DECK subcommands can be used interchangeably. (The editor decides on the appropriate action based on the object type (FILE or DECK).
- For more information, see the NOS/VE File Editor manual.

ERASE_TEXT EDIF Subcommand

Purpose Erases text at the specified location leaving blank characters in its place.

Format

ERASE_TEXT or
ERAT
NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
COLUMNS=keyword or *range of: keyword* or *integer*
BOUNDARY=keyword
STATUS=status variable

Parameters

NUMBER or *N*

Specifies the number of lines to be erased. Values can be integers or the keyword ALL.

If you omit this parameter and specify a range for the LINE parameter, NUMBER assumes a value of ALL.

If you omit this parameter without specifying a range of lines, NUMBER assumes a value of 1.

LINES or *LINE* or *L*

Specifies the range of lines affected by the erasure. Values can be an integer, line identifier, or one of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN.

If omitted, the current line is erased.

COLUMNS or *COLUMN* or *C*

Specifies the range of columns. The integers can be from 1 through 256 or any of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MARK, MAXIMUM.

When you specify a boundary of STREAM, erasing starts at the beginning column on the beginning line, continues through all columns of the next lines, and stops at the end column of the ending line.

If COLUMN, BOUNDARY, and LINE are omitted and NUMBER=ALL, erasing starts at the current line and ends at the last line. If COLUMN is omitted and LINE is specified, the specified lines are erased. If COLUMN, BOUNDARY, LINE, and NUMBER are omitted, the current line is erased.

BOUNDARY or *B*

Specifies the type of boundary that will limit the erasure. Values can be LINE, STREAM, or BOX. If you omit this parameter and do not specify a value for the COLUMN parameter, LINE is assumed. If you omit this parameter and specify values for both the LINE and COLUMN parameters, STREAM is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following example erases the first two lines of the file:

```
erase_text lines=1..2
```

- The following example erases columns 1 through 3 on all lines of the file:

```
erat l=all c=1..3 b=box
```

EXCHANGE_POSITION EDIF Subcommand

Purpose Saves the current position in the file you are editing and returns you to a previously saved position.

Format

EXCHANGE_POSITION or
EXCP
STATUS=status variable

Remarks

- You must save a position with the SAVE_POSITION subcommand before executing an EXCHANGE_POSITION subcommand.
- For more information, see the NOS/VE File Editor manual.

EXCHANGE_SCREEN_WIDTH EDIF Subcommand

Purpose Alternates between the 80- and 132-column screen displays, for those terminals that support them.

Format

EXCHANGE_SCREEN_WIDTH or
EXCSW
STATUS=status variable

Remarks

- To set your column width to a value other than 80 or 132, use the SET_SCREEN_OPTIONS subcommand.
- For more information, see the NOS/VE File Editor manual.

Examples If you are using an 80-column screen, entering

```
excsw
```

changes it to a 132-column screen.

FORMAT_PARAGRAPHS EDIF Subcommand

Purpose Adjusts words or sentences in a paragraph of text to bring line lengths as close as possible to preset margins.

Format

FORMAT_PARAGRAPHS or
FORMAT_PARAGRAPH or
FORP
NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
STATUS=status variable

Parameters *NUMBER* or *N*

Specifies the number of lines to format starting with current line and moving forward. If LINE is omitted and NUMBER is specified, the number of lines in the current paragraph specified by the NUMBER parameter are formatted. If both the NUMBER and LINE parameter are omitted, the current paragraph is assumed.

LINES or *LINE* or *L*

Specifies a range of lines to format. If omitted, the current paragraph is assumed.

Remarks

- Using this subcommand adds two blanks after periods, colons, exclamation marks, and question marks.
- A paragraph consists of any group of lines delimited by empty lines.
- Margins are set using the SET_PARAGRAPH_MARGINS subcommand.
- If you have not entered the SET_PARAGRAPH_MARGINS subcommand, the paragraph margins are set at 1 and 65. The first line is indented four characters.
- For more information, see the NOS/VE File Editor manual.

Examples The following example adjusts the current line and the 5 subsequent lines to conform to previously set margins.

```
format_paragraph number=6
```

$FUNCTION_ROW
EDIF Function

Purpose Returns an integer specifying the top row in which the menu of operations is displayed.

Format **$FUNCTION_ROW** or
$FR

Parameters None.

Remarks

- If you are in line mode, zero is returned.
- For more information, see the NOS/VE File Editor manual.

$FUNCTION_SIZE
EDIF Function

Purpose Returns an integer specifying the number of rows on the screen used by the menu of operations.

Format **$FUNCTION_SIZE** or
$FS

Parameters None.

Remarks

- If you are in line mode, zero is returned.
- For more information, see the NOS/VE File Editor manual.

Examples

- The following commands display the number of screen rows required to display a single menu row.

```
set_screen_options mr=1
display_value $function_size
2
```

- The following command, executed repeatedly, will display 0, 1, and 2 rows of the menu of operations.

```
setso mr=$mod($function_size/2+1,3)
```

$HOME_ROW
EDIF Function

Purpose Returns an integer specifying the row used for entering subcommands and responses to the editor.

Format **$HOME_ROW** or
$HR

Parameters None.

Remarks

- If you are in line mode, zero is returned.
- For more information, see the NOS/VE File Editor manual.

INDENT_TEXT EDIF Subcommand

Purpose Inserts blank characters or deletes characters in front of lines of text.

Format

INDENT_TEXT or
INDT
OFFSET=integer
NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
STATUS=status variable

Parameters

OFFSET or *O*

Specifies the number of columns to indent the specified block of text.

If positive, that number of blanks are added.

If negative, that number of characters are deleted.

If omitted, 1 is assumed.

NUMBER or *N*

Specifies the number of lines to be indented.

If you specify a range for the LINES parameter, the NUMBER parameter assumes a value of ALL.

If no range is specified for the LINES parameter, the NUMBER parameter assumes a value of 1.

LINES or *LINE* or *L*

Specifies a range of lines to be indented. Values can be an integer, line identifier, or one of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN.

If no range is specified, range is the current line to the last line.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following example deletes the first 7 characters from lines 25 through the last line:

```
indent_text o=-7 l=25..last
```

- The following example indents all lines five spaces:

```
indt o=5 l=all
```

INSERT_CHARACTERS EDIF Subcommand

Purpose Inserts a string of characters before a specified location.

Format

INSERT_CHARACTERS or
INSC or
INSERT_CHARACTER or
IC
NEW_TEXT=string
INSERTION_LOCATION=keyword or *range of: keyword* or *integer* or *line_identifier*
INSERTION_COLUMN=keyword or *integer*
NUMBER=keyword or *integer*
STATUS=status variable

Parameters

NEW_TEXT or *NT*

Specifies the text to be inserted.

If omitted, one blank character is inserted.

INSERTION_LOCATION or *INSERTION_LOCATIONS* or *L* or *LINE* or *IL*

Specifies the line or range of lines in which the text is inserted. Values can be an integer, line identifier, or one of the following keywords: ALL, MARK, SCREEN, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN.

If omitted, the value is the current line.

INSERTION_COLUMN or *C* or *COLUMN* or *IC*

Specifies the column before which the insertion is to occur. Values can be an integer from 1 through 256 or any of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MAXIMUM. Ranges are not allowed.

If omitted, the current column is assumed.

NUMBER or *N*

Specifies the number of lines to process. If omitted, 1 is assumed. If the INSERTION_LOCATION parameter provides an explicit range of lines, ALL is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following inserts the text *Short Comment* in front of the current column on the current line:

```
insert_characters 'Short Comment'
```

- Using an SCL string variable as the value for the NEW_TEXT parameter is an efficient way of inserting the same text numerous places in a file. For example, you could initialize a string variable as follows:

```
a = 'characters to be inserted'
```

When the cursor is positioned at a point where you want to insert the text, enter the following command:

```
insc a
```

The following example inserts a dollar sign before column 10 on each line in the marked region.

```
insc nt='$' c=10 l=mark
```

INSERT_EMPTY_LINES EDIF Subcommand

Purpose Enables you to insert empty lines.

Format

INSERT_EMPTY_LINES or
INSERT_EMPTY_LINE or
INSEL
NUMBER=integer
INSERTION_LOCATION=keyword or *integer* or *line_identifier*
PLACEMENT=keyword
STATUS=status variable

Parameters *NUMBER* or *N*

Specifies the number of empty lines to be inserted.

If omitted, 1 is assumed.

INSERTION_LOCATION or *IL*

Specifies the line before or after which the insertion is to occur. Values can be an integer, line identifier, or one of the keywords: CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. Ranges are not allowed.

If omitted, CURRENT is assumed.

PLACEMENT or *P*

Specifies whether the insertion is to occur BEFORE(B) or AFTER(A) the line specified by the INSERTION_LOCATION parameter.

If omitted, AFTER is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples The following inserts 2 empty lines after line 50:

```
insel number=2 insertion_location=50
```

INSERT_LINES
EDIF Subcommand

Purpose Inserts one or more lines of text.

Format

INSERT_LINES or
INSERT_LINE or
INSL or
I

NEW_TEXT=string
PLACEMENT=keyword
INSERTION_LOCATION=keyword or *integer* or *line_identifier*
UNTIL=string
STATUS=status variable

Parameters

NEW_TEXT or *NT*

Specifies the new line of text to be inserted.

If NEW_TEXT is omitted, the text to be inserted is taken from the command input file.

PLACEMENT or *P*

Indicates whether the insertion is to occur BEFORE (B) or AFTER (A) the location specified by the INSERTION_LOCATION parameter.

If omitted, AFTER is assumed.

INSERTION_LOCATION or *IL*

Specifies the line after which or before which the insertion is to occur. Values can be an integer, line identifier, or one of the keywords: CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. Ranges are not allowed.

If omitted, CURRENT is assumed.

UNTIL or *U*

In line mode, specifies a string that stops the insert.

If the NEW_TEXT parameter is omitted, you are prompted to enter input until the editor encounters the string specified by this parameter at the end of a line.

If the UNTIL parameter is omitted, ** is assumed.

Remarks

For more information, see the NOS/VE File Editor manual.

Examples

- The following inserts the line *NEW LINE* after the current line.

```
insert_lines 'NEW LINE'
```

- The following inserts the line *Insert* before the current line:

```
insl nt='Insert' p=b
```

- The following inserts the line *First line* before the first line of the file.

```
insl nt='First line' il=first p=b
```

- The following inserts lines from the command input file before line 45 until a # character is encountered as the last character in a line.

```
i il=45 p=b u='#'
```

INSERT_WORD
EDIF Subcommand

Purpose Inserts a string or 32 blank characters.

Format

INSERT_WORD or
INSW or
IW
NEW_TEXT=string
INSERTION_LOCATION=keyword or *integer* or *line_identifier*
INSERTION_COLUMN=keyword or *integer*
STATUS=status variable

Parameters

NEW_TEXT or *NT*

Specifies the string to be inserted. The default is 32 blanks.

INSERTION_LOCATION or *IL*

Specifies the line in which the string is to be inserted. Values can be an integer, line identifier, or one of the keywords: CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. Ranges are not allowed.

If omitted, value is the current line.

INSERTION_COLUMN or *IC*

Specifies the column before which the insertion is to occur. Values can be an integer from 1 through 256 or any of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MAXIMUM.

If omitted, CURRENT is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following inserts 32 blank characters before the current column of the current line:

```
insert_word
```

- The following inserts the word *LINE* in front of line 10:

```
insw nt='LINE' il=10 ic=1
```

JOIN_TEXT
EDIF Subcommand

Purpose Joins a line with the next line by appending the second to the first.

Format

JOIN_TEXT or
JOIT or
J

LINES=keyword or *integer* or *line_identifier*
COLUMN=keyword or *integer*
STATUS=status variable

Parameters

LINES or *LINE* or *L*

Specifies the first of two lines to be joined. The next line is joined to the specified line. Values can be an integer, line identifier, or one of the keywords: CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. Ranges are not allowed.

If omitted, CURRENT is assumed.

COLUMN or *C*

Specifies the starting column to which the second line is moved. The second line is always added after the end of the first line. The columns parameter determines how far after the first line the second line is added.

Values can be an integer from 1 through 256 or any of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MAXIMUM. Ranges are not allowed.

If the value you specify is less than or equal to the length of the first line, the line is added to the end of the first line. If the value you specify is greater than the length of the first line, the editor fills the columns in between with blank characters.

If COLUMN is omitted, CURRENT is assumed.

Remarks

- If the joined line is longer than 256 characters, the subcommand is not performed and the editor displays the following message:

```
Line length exceeded.
```

- For more information, see the NOS/VE File Editor manual.

$LAST_COMMAND EDIF Function

Purpose Returns a string containing the last successfully completed operation.

Format **$LAST_COMMAND** or
$LC

Parameters None.

Remarks An operation can be any of the following:

- A single subcommand.
- A sequence of subcommands entered on the home line in screen mode or at the prompt in line mode. Individual subcommands are separated by semicolons.
- A sequence of subcommands executed by pressing the key(s) identified in your menu of operations or a dedicated key.
- For more information, see the NOS/VE File Editor manual.

$LINES_ENABLED EDIF Function

Purpose Returns the current keyword value for the LINES parameter of the ENABLE_LINES subcommand. The keywords ALL or ACTIVE indicate which type of lines DELETE_LINES, INSERT_LINES, and REPLACE_LINES accept as values for the LINES or INSERTION_LOCATION parameters.

Format **$LINES_ENABLED** or
$LE

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

$LINE_IDENTIFIER EDIF Function

Purpose Returns the line identifier of the current line (for editing decks only). The value is of type line identifier unless the context calls for a string.

Format **$LINE_IDENTIFIER** or
$LI

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

$LINE_TEXT EDIF Function

Purpose Returns the text of the current line as a string.

Format **$LINE_TEXT** or
$LT

Parameters None.

Remarks

- One of the uses for this function is in procedures that operate on lines of a file. You can use the POSITION_CURSOR subcommand to move to a line, and then use the $LINE_TEXT function to return the value of the line.
- For more information, see the NOS/VE File Editor manual.

Examples

- The following adds the string 'append' to the end of the current line:

```
replace_line nt=$lt//'append'
```

- The following expression returns the length of the current line:

```
$size($line_text)
```

- The following executes the current line:

```
include_command command=$lt
```

LIST_BACKWARDS EDIF Subcommand

Purpose In line mode, displays a range of lines ending with the current line. In effect, it enables you to view a number of lines just before the current line and end up where you started.

Format

LIST_BACKWARDS or
LISB or
LIST_BACKWARD or
LB
NUMBER=keyword or *integer*
STATUS=status variable

Parameters *NUMBER* or *N*

Specifies the number of lines to list. Values can be integers or the keyword ALL. ALL lists all lines from the beginning of the file to the current line.

If NUMBER is omitted, a value of 1 is assumed.

Remarks

- This subcommand is typically only used in line mode.
- For more information, see the NOS/VE File Editor manual.

Examples The following example lists 15 lines ending with the current line.

```
list_backward n=15
```

LIST_FORWARDS
EDIF Subcommand

Purpose In line mode, displays a range of lines beginning with the current line.

Format

LIST_FORWARDS or
LISF or
LIST_FORWARD or
LF
NUMBER=keyword or *integer*
STATUS=status variable

Parameters *NUMBER* or *N*

Specifies the number of lines to list. Values can be integers or the keyword ALL. ALL lists all lines from the current line to the end of the file.

If NUMBER is omitted, a value of 1 is assumed.

Remarks

- This subcommand is typically used only in line mode.
- For more information, see the NOS/VE File Editor manual.

Examples The following example lists 15 lines beginning with the current line.

```
list_forward n=15
```

LIST_LINES
EDIF Subcommand

Purpose In line mode, lists a specified line or range of lines. In screen mode, the cursor is positioned at the specified line, or the last line in the range.

Format

LIST_LINES or
LISL or
LIST_LINE or
LL
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
STATUS=status variable

Parameters *LINES* or *LINE* or *L*

Specifies the line or range of lines to list. Values can be an integer, line identifier, or one of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN.

If LINE is omitted, CURRENT is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples The following example lists lines 25 through 40.

```
list_lines l=25..40
```

LOCATE_ALL
EDIF Subcommand

Purpose Searches the entire file to locate all occurrences of a specified string. In screen mode, all occurrences are then listed in a directory enabling you to either position the cursor at a specific line or enter the desired line number. In line mode, all occurrences are listed and you are positioned at the last occurrence of the string.

Format **LOCATE_ALL** or
LOCA or
LA
TEXT=range of string
UPPER_CASE=boolean
WORD=boolean
STATUS=status variable

Parameters *TEXT* or *T*

Specifies the text string you want to find. If omitted, the last text string specified is assumed, if any.

UPPER_CASE or *UC*

Determines the significance of capitalization in the search.

If you specify TRUE, the editor searches the file as if it were all uppercase.

If you specify FALSE, the editor searches for the text exactly as it was entered.

If you omit UPPER_CASE, FALSE is assumed. If omitted and no string is entered for the TEXT parameter, the last value specified is assumed.

WORD or *W*

Determines whether the editor searches for the specified text string as a word (the text you want to search for is surrounded by nonalphanumeric characters).

If you specify TRUE, the editor searches for the text as a word. If you specify FALSE, it doesn't.

If you omit WORD, FALSE is assumed. If omitted and no string is entered for the TEXT parameter, the last value specified is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples The following example locates all occurrences of the string, *find this text,* in the file and lists them.

```
locate_all text='find this text'
```

LOCATE_EMPTY_LINES EDIF Subcommand

Purpose Finds empty lines. An empty line is a line of all blank characters.

Format

LOCATE_EMPTY_LINES or
LOCATE_EMPTY_LINE or
LOCEL
NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
DIRECTION=keyword
VETO=boolean
STATUS=status variable

Parameters *NUMBER* or *N*

Specifies the number of empty lines to find. Values can be numbers or the keyword ALL.

If a LINE parameter is specified, NUMBER assumes a value of ALL.

If no LINE parameter is specified, NUMBER assumes a value of 1.

LINES or *LINE* or *L*

Specifies a range of lines to search.

Values can be an integer, line identifier, or one of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN. If you specify a value of only one line, the search is limited to that line.

If you omit LINE and specify BACKWARD for the DIRECTION parameter, CURRENT..FIRST is assumed.

If both LINE and DIRECTION are omitted, CURRENT..LAST is assumed.

DIRECTION or *D*

Specifies whether to search FORWARD (F) or BACKWARD (B) from the current line.

If no value is specified, FORWARD is assumed.

VETO or *V*

Instructs the editor to turn the veto option on or off.

If TRUE is specified, the editor displays a directory of located lines.

If VETO is omitted, FALSE is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following positions the cursor to the fifth empty line:

```
locate_empty_lines number=5
```

- The following positions the cursor to the last empty line:

```
locel l=20..40
```

- The following positions the cursor to the tenth empty line in the marked text:

```
locel n=10 l=mark
```

LOCATE_NEXT
EDIF Subcommand

Purpose Locates the next occurrence of a previously specified string. Previously specified values for the UPPER_CASE and WORD parameters also remain in effect. The search begins one column after the current column.

Format

LOCATE_NEXT or
LOCN or
LN
STATUS=status variable

Remarks For more information, see the NOS/VE File Editor manual.

LOCATE_STRING
EDIF Subcommand

Purpose Beginning at the current line and column, it searches for the specified string.

Format

LOCATE_STRING or
LOCS or
LS
TEXT=range of string
UPPER_CASE=boolean
WORD=boolean
STATUS=status variable

Parameters *TEXT* or *T*

Specifies strings of text in the first and last lines of a block of text to be located. If you enter only one string, the block of text to be located will contain only one line. If you enter two strings, the search for the second begins immediately after the first is found and the cursor is positioned at the beginning of the first string.

If omitted, the last string parameter specified, if any, is used.

UPPER_CASE or *UC*

Determines the significance of capitalization in the search.

If you specify TRUE, the editor searches the file as if all characters were uppercase.

If you specify FALSE, the editor searches for the text exactly as it was entered.

If you omit UPPER_CASE, FALSE is assumed. If omitted and no string is entered for the TEXT parameter, the last value specified is assumed.

WORD or *W*

Determines whether the editor searches for the specified text string as a word (the text you want to search for is surrounded by nonalphanumeric characters).

If you specify TRUE, the editor searches for the text as a word. If you specify FALSE, it doesn't.

If you omit WORD, FALSE is assumed. If omitted and no string is entered for the TEXT parameter, the last value specified is assumed.

Remarks

- This subcommand is typically only used in line mode.
- For more information, see the NOS/VE File Editor manual.

Examples

The following example locates the string *work now*:

```
locate_string 'work now'
```

LOCATE_TEXT EDIF Subcommand

Purpose

Locates blocks of text.

Format

LOCATE_TEXT or
LOCT or
L

TEXT=range of string
NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
COLUMNS=keyword or *range of: keyword* or *integer*
BOUNDARY=keyword
DIRECTION=keyword
UPPER_CASE=boolean
WORD=boolean
REPEAT_SEARCH=boolean
VETO=boolean
STATUS=status variable

Parameters *TEXT* or *T*

Specifies strings of text in the first and last lines of a block of text to be located. If you enter only one string, the block of text to be located will contain only one line. If you enter two strings, the search for the second begins immediately after the first is found and the cursor is positioned at the beginning of the first string.

If TEXT is omitted, the lines to be located will be determined by the NUMBER, LINE, and DIRECTION parameters.

NUMBER or *N*

Specifies the number of blocks of text to be found. Values for this parameter can be an integer or the keyword ALL(A).

In line mode, use the NUMBER parameter to display a range of lines.

If you specify a range of values for the LINE parameter, NUMBER assumes a value of ALL.

If no range is specified for the LINE parameter, NUMBER assumes a value of 1.

LINES or *LINE* or *L*

Specifies a range of lines to be searched.

Values can be an integer, line identifier, or any of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN. If one line is specified, the search is limited to that line.

If you specify MARK for this parameter, the values preserved in the mark are used for the COLUMNS and BOUNDARY parameters. Specifying MARK also ensures that the marked deck or file is made current.

In line mode, use the LINE parameter to specify which lines to print.

If you omit LINE and specify BACKWARD for the DIRECTION parameter, CURRENT..FIRST is assumed.

If LINE and DIRECTION are both omitted, CURRENT..LAST is assumed.

COLUMNS or *COLUMN* or *C*

Specifies the range of columns to search. Values can be an integer from 1 through 256 or any of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MARK, MAXIMUM.

If omitted, CURRENT is assumed. If omitted and you have specified LINE=MARK, the marked lines provide the column boundaries. If COLUMN is omitted and you have specified a LINE parameter other than MARK, all columns will be searched.

BOUNDARY or *B*

Specifies the type of boundary that limits the search. Values can be BOX, LINE, or STREAM.

If COLUMNS is specified and BOUNDARY is omitted, STREAM is assumed.

If both BOUNDARY and COLUMNS are omitted, LINE is assumed.

DIRECTION or *D*

Specifies whether to search FORWARD (F) or BACKWARD (B) from the current line.

If no value is specified, FORWARD is assumed.

UPPER_CASE or *UC*

Determines the significance of capitalization in the search.

If you specify TRUE, the editor searches the file as if it were all uppercase.

If you specify FALSE, the editor searches for the text exactly as it was entered.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for UPPER_CASE is used.

WORD or *W*

Determines whether the editor searches for the specified text string as a word (the text you want to search for is surrounded by nonalphanumeric characters).

If you specify TRUE, the editor searches for the text as a word. If you specify FALSE, it doesn't.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for WORD is used.

REPEAT_SEARCH or *RS*

Instructs the editor how to use the values entered for the last TEXT, UPPER_CASE, and WORD parameters.

If you specify TRUE, the editor uses the same TEXT, UPPER_CASE, and WORD parameters as the last time you entered them for any subcommand (unless you have specified values for this subcommand).

If you specify FALSE, the editor uses the parameters entered with the current subcommand.

If omitted, FALSE is assumed.

VETO or *V*

Instructs the editor to turn the veto option on or off.

If TRUE is specified, the editor displays a directory of located lines.

If VETO is omitted, FALSE is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following example locates the next occurrence of *PROCEND*:

```
locate_text 'PROCEND'
```

- The following example locates the previous occurrence of *TITLE*:

```
loct 'TITLE' d=b
```

- The following example positions the cursor on line 250 of the current file or deck:

```
loct l=250
```

- The following example locates the string you last specified as a value for the TEXT parameter:

```
loct rs=true
```

- The following example locates all occurrences of PARAMETER from the current position to the end of the file and displays the lines in a directory-type display:

```
l 'PARAMETER' n=all v=true
```

- The following example locates the next block of text beginning with *one* and ending with *twenty*:

```
l 'one'..'twenty'
```

- The following example prints the current line and four subsequent lines in line mode. In screen mode, the cursor is positioned four lines forward:

```
l n=5
```

- The following example displays a directory of each occurrence of the word *MISPELL* regardless of case:

```
l 'mispell' l=a uc=true v=true
```

- The following example locates all instances of the string *abc* between columns 20 and 30 on any line in the file.

```
l 'abc' l=all c=20..30 b=box
```

LOCATE_WIDE_LINES EDIF Subcommand

Purpose Locates lines that are wider than the margins set by the SET_LINE_WIDTH subcommand.

Format

LOCATE_WIDE_LINES or
LOCATE_WIDE_LINE or
LOCWL
NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
DIRECTION=keyword
VETO=boolean
STATUS=status variable

Parameters

NUMBER or *N*

Specifies the number of wide lines to be found. Values for this parameter can be an integer or the keyword ALL(A).

If a LINES parameter is specified, NUMBER assumes a value of ALL. Otherwise, the assumed value for NUMBER is 1.

LINES or *LINE* or *L*

Specifies a range of lines to be searched. Values can be an integer, line identifier, or one of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN. If you specify a value for only one line, the search is limited to that line.

If LINE and DIRECTION are omitted, CURRENT..LAST is assumed. If you omit LINE and specify BACKWARD for the DIRECTION parameter, CURRENT..FIRST is assumed.

DIRECTION or *D*

Specifies whether to search FORWARD (F) or BACKWARD (B) from the current line.

If omitted, FORWARD is assumed.

VETO or *V*

Instructs the editor to turn the veto option on or off.

If TRUE is specified, the editor displays a directory of located lines.

If VETO is omitted, FALSE is assumed.

Remarks

For more information, see the NOS/VE File Editor manual.

Examples

- The following locates the first wide line in the file:

```
locate_wide_lines number=1 lines=all
```

- The following locates the next wide line starting at the current line:

```
locwl
```

- The following locates and displays a directory of all wide lines between the top line of the current screen and the last line of the file:

```
locwl l=first_screen..last v=true
```

MARK_BOXES
EDIF Subcommand

Purpose Marks a rectangular area of text.

Format

MARK_BOXES or
MARB or
MARK_BOX or
MB
LINES=keyword or *integer* or *line_identifier* or *range of: keyword* or *integer* or *line_identifier*
COLUMNS=keyword or *integer* or *range of: keyword* or *integer*
STATUS=status variable

Parameters

LINES or *LINE* or *L*

Specifies the lines in which the corners of the box reside. Values can be an integer, line identifier, or one of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN.

If omitted, CURRENT is assumed.

COLUMNS or *COLUMN* or *C*

Specifies the columns in which the corners of the box reside. Values can be any integer from 1 through 256 or any of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MARK, MAXIMUM.

If omitted, CURRENT is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following example marks a box 5 lines by 1 column; the marked area will cover lines 4 through 8 at column 12:

```
mark_box lines=4..8 column=12
```

- To mark a box with dimensions 5 lines by 3 columns, enter:

```
mb l=2..6 c=3..5
```

The marked area covers lines 2, 3, 4, 5, and 6 at columns 3, 4, and 5 in each line as illustrated below:

```
  1 2 3 4 5 6 7 8

1 x x x x x x x x
2 x x x x x x x x
3 x x x x x x x x
4 x x x x x x x x
5 x x x x x x x x
6 x x x x x x x x
7 x x x x x x x x
```

The same results can be achieved by positioning the cursor to the upper left corner of the intended box (line 2, column 3), entering the MARK_BOX subcommand, then positioning the cursor to the lower right corner of the intended box, (line 6, column 5) and entering the MARK_BOX subcommand.

MARK_CHARACTERS EDIF Subcommand

Purpose Marks specific characters. These marks specify the boundary for text that is to be processed later by subcommands that insert, delete, move, copy, and replace text.

Format

MARK_CHARACTERS or
MARC or
MARK_CHARACTER or
MC

LINES=keyword or *integer* or *line_identifier* or *range of: keyword* or *integer* or *line_identifier*
COLUMNS=keyword or *integer* or *range of: keyword* or *integer*
STATUS=status variable

Parameters

LINES or *LINE* or *L*

Specifies the lines in which the marked characters reside.

If LINES is omitted, CURRENT is assumed.

COLUMNS or *COLUMN* or *C*

Specifies the columns to be marked within the specified line(s). Values can be any integer from 1 through 256 or any of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MARK, MAXIMUM.

If COLUMN is omitted, CURRENT is assumed.

Remarks

- If a character is specified, only that character is marked. If a single character is specified and another single character is already marked, the characters between the two will become marked. If a range is specified, the entire range is marked and any other marks are unmarked.

- Even though you can mark a range of characters by entering two MARK_CHARACTER subcommands, if you mark one character and then reference the mark in another editing operation such as inserting or copying, the editor assumes the marking operation is complete. Then, when you mark a second character, the editor starts another marking operation; the first character is unmarked, and the second marked.

- The following functions can be used to determine the location and type of the marked region.

  $MARK_FIRST_COLUMN
  $MARK_FIRST_LINE
  $MARK_LAST_COLUMN
  $MARK_LAST_LINE
  $MARK_TYPE (returns a value of STREAM)

- For more information, see the NOS/VE File Editor manual.

Examples

- The following marks column 30 of line 40 through column 30 of line 50:

```
mark_character line=40..50 column=30
```

- The following marks columns 7 through 10 of the current line:

```
mc c=7..10
```

- To mark column 5 of line 2 through column 3 of line 5, enter:

```
mc l=2..5 c=5..3
```

The marked area covers column 5 of line 2 through column 3 of line 5 as illustrated below:

```
  1 2 3 4 5 6 7 8 ...

1 x x x x x x x x
2 x x x x x x x x
3 x x x x x x x x
4 x x x x x x x x
5 x x x x x x x x
6 x x x x x x x x
7 x x x x x x x x
```

The same results can be achieved by positioning the cursor to the first character to be marked (line 2, column 5), entering the MARK_CHARACTER subcommand, then positioning the cursor to the last character to be marked (line 5, column 3) and entering the MARK_CHARACTER subcommand.

$MARK_FIRST_COLUMN EDIF Function

Purpose Returns an integer specifying the column number of the first marked column.

Format **$MARK_FIRST_COLUMN** or **$MFC**

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

$MARK_FIRST_LINE EDIF Function

Purpose Returns an integer specifying the line number of the first marked line.

Format **$MARK_FIRST_LINE** or
$MFL

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

$MARK_LAST_COLUMN EDIF Function

Purpose Returns an integer specifying the column number of the last marked column.

Format **$MARK_LAST_COLUMN** or
$MLC

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

$MARK_LAST_LINE EDIF Function

Purpose Returns an integer specifying the line number of the last marked line.

Format **$MARK_LAST_LINE** or
$MLL

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

Examples The following statements write marked lines to file $LOCAL.SCR1. If no lines (or one line) are marked, all lines are written to the file.

```
lines_to_write = all
IF $mark_first_line<>$mark_last_line THEN
   lines_to_write = mark
IFEND
write_file f=$local.scr1 l=lines_to_write
```

MARK_LINES
EDIF Subcommand

Purpose Marks a line to be processed later.

Format

MARK_LINES or
MARK_LINE or
MARL or
ML
LINES=keyword or *integer* or *line_identifier* or *range of: keyword* or *integer* or *line_identifier*
STATUS=status variable

Parameters *LINES* or *LINE* or *L*

Specifies a line or range of lines to be marked. Marked text can be processed by subcommands that insert, delete, move, copy, and replace text. Values can be an integer, line identifier, or one of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN.

If LINE is omitted, the current line is assumed.

Remarks

- If a line is specified, only that line is marked. If a single line is specified and another single line is already marked, the lines between the two will become marked. If a range is specified, the entire range is marked and any other marks are unmarked.

- Even though you can mark a range of lines by entering two MARK_LINES subcommands, if you mark one line and then reference the mark in another editing operation such as inserting or copying, the

editor assumes the marking operation is complete. Then, when you mark a second line, the editor starts another marking operation; the first line is unmarked, and the second marked.

- The following functions can be used to determine the location and type of the marked region.

  $MARK_FIRST_LINE
  $MARK_LAST_LINE
  $MARK_TYPE (returns a value of LINES)

- For more information, see the NOS/VE File Editor manual.

$MARK_OBJECT EDIF Function

Purpose Returns a string specifying the name of the current file or deck containing the marked text.

Format **$MARK_OBJECT** or
$MO

Parameters None.

Remarks

- Use the $MARK_OBJECT_TYPE function to determine if the object is a file or a deck.
- For more information, see the NOS/VE File Editor manual.

$MARK_OBJECT_TYPE EDIF Function

Purpose Returns a string specifying if the marked text is in a file or a deck. Values returned can be FILE, DECK, or NULL.

Format **$MARK_OBJECT_TYPE** or
$MOT

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

$MARK_TYPE
EDIF Function

Purpose Returns a value indicating whether the marked region is bounded by lines, bounded by characters, or is a box. Values returned can be LINES (line boundary), STREAM (character boundary), or BOX (marked zone boundary).

Format **$MARK_TYPE** or
$MT

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

$MARKED_STRING
EDIF Function

Purpose Returns a list of strings containing marked characters.

Format **$MARKED_STRING** or
$MS

Parameters None.

Remarks

- If no strings are marked, an empty list is returned.
- For more information, see the NOS/VE File Editor manual.

$MESSAGE_ROW
EDIF Function

Purpose Returns an integer specifying the number of the row on the screen used to display messages.

Format **$MESSAGE_ROW** or
$MR

Parameters None.

Remarks

- If you are in line mode, zero is returned.
- This function is often used as the value for the ROW parameter on the PUT_ROW subcommand.
- For more information, see the NOS/VE File Editor manual.

$MOUSE_COLUMN EDIF Function

Purpose Returns the column number of the mouse position for the most recent mouse click. This is intended for CONNECT VIEW users.

Format **$MOUSE_COLUMN** or
$MC

Parameters None.

Remarks For more information about CONNECT VIEW, see the CONNECT VIEW for the IBM PC manual.

$MOUSE_ROW EDIF Function

Purpose Returns the row number of the mouse position for the most recent mouse click. This is intended for CONNECT VIEW users.

Format **$MOUSE_ROW**

Parameters None.

Remarks For more information about CONNECT VIEW, see the CONNECT VIEW for the IBM PC manual.

MOVE_TEXT
EDIF Subcommand

Purpose Moves a block of text from one place to another in the same file.

Format

MOVE_TEXT or
MOVT or
M

TEXT=range of string
NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
COLUMNS=keyword or *range of: keyword* or *integer*
INSERTION_LOCATION=keyword or *integer* or *line_identifier*
INSERTION_COLUMN=keyword or *integer*
PLACEMENT=keyword
BOUNDARY=keyword
UPPER_CASE=boolean
WORD=boolean
REPEAT_SEARCH=boolean
STATUS=status variable

Parameters

TEXT or *T*

Specifies string(s) of text in the first and last lines of a block of text to be moved. If you enter only one string, the block of text to be moved will contain only one line. If you enter two strings, the search for the second begins immediately after the first is found.

If TEXT is specified, the INSERTION_COLUMNS and BOUNDARY parameters are ignored and line boundaries are used.

If omitted, the lines to be copied will be determined by the NUMBER and LINES parameters or the REPEAT_SEARCH parameter.

NUMBER or *N*

Specifies the number of blocks of text to be moved. Values for this parameter can be numbers or the keyword ALL (A).

If omitted and a range is specified for the LINES parameter, this parameter assumes a value of ALL. Otherwise, the assumed value is 1.

LINES or *LINE* or *L*

Specifies the range of lines to be searched for the text to be moved. If a single value is specified, only that line is searched. Values can be an integer, line identifier, or one of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN.

If omitted, CURRENT..LAST is assumed.

COLUMNS or *COLUMN* or *C*

Specifies the range of columns to be searched for text to be moved. The integers can be from 1 through 256 or any of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MARK, MAXIMUM.

If omitted, CURRENT is assumed. If omitted and you have specified LINE=MARK, the marked lines provide the column boundaries. If COLUMN is omitted and you have specified a LINE parameter other than MARK, all columns will be searched.

INSERTION_LOCATION or *IL*

Specifies the line before or after which the text is to be moved (depending on the value of the PLACEMENT parameter). Values can be an integer, line identifier, or one of the LINE keywords: CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. Ranges are not allowed.

If omitted, CURRENT is assumed.

INSERTION_COLUMN or *IC*

Specifies the column before or after which the text is to be moved (depending on the value of the PLACEMENT parameter). Values can be an integer from 1 through 256, or any of the COLUMN keywords: CURRENT, FIRST_MARK, LAST_MARK, MAXIMUM. Ranges are not allowed.

If omitted, CURRENT is assumed.

If a value for TEXT is specified, INSERTION_COLUMN is ignored.

PLACEMENT or *P*

Specifies if the moved lines are to appear BEFORE (B) or AFTER (A) the location specified by the INSERTION_LOCATION parameter.

If omitted, AFTER is assumed.

BOUNDARY or *B*

Specifies the type of boundary that will limit the search. Values can be BOX, LINE, or STREAM.

If COLUMNS is specified and BOUNDARY is omitted, STREAM is assumed.

If both BOUNDARY and COLUMNS are omitted, LINE is assumed.

If a value for TEXT is specified, BOUNDARY is ignored; line boundaries are used.

UPPER_CASE or *UC*

Determines the significance of capitalization in the search.

If you specify TRUE, the editor searches the file as if it were all uppercase.

If you specify FALSE, the editor searches for the text exactly as it was entered.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for UPPER_CASE is used.

WORD or *W*

Determines whether the editor searches for the specified text string as a word (the text you want to search for is surrounded by nonalphanumeric characters).

If you specify TRUE, the editor searches for the text as a word. If you specify FALSE, it doesn't.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for WORD is used.

REPEAT_SEARCH or *RS*

Instructs the editor how to use the values entered for the last TEXT, UPPER_CASE, and WORD parameters.

If you specify TRUE, the editor uses the same TEXT, UPPER_CASE, and WORD parameters as the last time you entered them for any subcommand (unless you have specified values for this subcommand).

If you specify FALSE, the editor uses the parameters entered with the current subcommand.

If omitted, FALSE is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following moves lines 30 through 40 to immediately after the current line.

```
move_text line=30..40
```

- The following moves the next occurrence of a block of text beginning with the line containing *batch* and ending with the line containing *interactive* to immediately before line 71:

```
movt t='batch'..'interactive' il=71 p=b
```

- The following moves a box of characters from columns 4 through 6 in lines 2 through 4 to columns 5 through 7 of lines 5 through 7:

```
m l=2..4 c=4..6 il=5 ic=5 b=box
```

Before move:

```
abcdefghij
 abcdefghij
  abcdefghij
   abcdefghij
1234567890
1234567890
1234567890
1234567890
```

After move:

```
abcdefghij
 abfghij
  aefghij
   defghij
1234cde567890
1234bcd567890
1234abc567890
1234567890
```

$NEW_TEXT
EDIF Function

Purpose Returns the last string entered in a NEW_TEXT parameter.

Format **$NEW_TEXT** or
$NT

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

Examples If you enter:

```
replace_text text='good' new_text='the best'
```

you can then use:

```
r t='better' nt=$nt
```

to replace *better* with *the best.*

$NUMBER_OF_COLUMNS
EDIF Function

Purpose Returns an integer specifying the number of columns currently being used to display text on the screen.

Format **$NUMBER_OF_COLUMNS** or
$NUMBER_OF_COLUMN or
$NOC

Parameters None.

Remarks

- If you are in line mode, zero is returned.
- For more information, see the NOS/VE File Editor manual.

$NUMBER_OF_LINES EDIF Function

Purpose Returns an integer specifying the number of active lines in the current object.

Format **$NUMBER_OF_LINES** or
$NOL

Parameters None.

Remarks

- If no object is selected, zero is returned.
- For more information, see the NOS/VE File Editor manual.

$NUMBER_OF_MARKS EDIF Function

Purpose Returns an integer value indicating whether nothing is marked (0), one end of a region is marked (1), or both ends of a region are marked (2).

Format **$NUMBER_OF_MARKS** or
$NOM

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

$NUMBER_OF_ROWS EDIF Function

Purpose Returns an integer specifying the number of rows currently displayed on the screen, including the menu of operations, message line, home line, and file header.

Format **$NUMBER_OF_ROWS** or
$NUMBER_OF_ROW or
$NOR

Parameters None.

Remarks

- If you are in line mode, zero is returned.
- For more information, see the NOS/VE File Editor manual.

$NUMBER_OF_SPLITS EDIF Function

Purpose Returns an integer specifying the number of splits on the screen.

Format **$NUMBER_OF_SPLITS** or
$NUMBER_OF_SPLIT or
$NOS

Parameters None.

Remarks

- If you are in line mode, zero is returned.
- For more information, see the NOS/VE File Editor manual.

Examples The following alternates between 1 or 2 screens:

```
set_screen_option s=3-$number_of_splits
```

$OBJECT_MODIFIED EDIF Function

Purpose Returns a boolean value indicating whether the object is modified.

Format **$OBJECT_MODIFIED** or
$OM

Parameters None.

Remarks

- If no object is selected, FALSE is returned.
- For more information, see the NOS/VE File Editor manual.

$OFFSET
EDIF Function

Purpose Returns an integer identifying the number specified on the OFFSET parameter of the ALIGN_SCREEN subcommand.

Format
$OFFSET or
$O

Parameters None.

Remarks
- If you have not specified the OFFSET parameter, or are in line mode, zero is returned.
- For more information, see the NOS/VE File Editor manual.

OVERLAY_TEXT
EDIF Subcommand

Purpose Replaces text at the specified location with text determined by the LINES and COLUMN parameters.

Format
OVERLAY_TEXT or
OVET or
O

NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
COLUMNS=keyword or *range of: keyword* or *integer*
OVERLAY_LOCATION=keyword or *range of: keyword* or *integer* or *line_identifier*
OVERLAY_COLUMNS=keyword or *range of: keyword* or *integer*
BOUNDARY=keyword
STATUS=status variable

Parameters *NUMBER* or *N*

Specifies the maximum number of lines to be overlayed. Values can be numbers or the keyword ALL.

If you omit this parameter and specify a range for the LINE parameter, NUMBER assumes a value of ALL.

If you omit this parameter without specifying a range of lines, NUMBER assumes a value of 1.

LINES or *LINE* or *L*

Specifies the range of lines to be copied. Values can be an integer, line identifier, or one of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN.

If you enter a single value, only that line is copied.

If LINES and COLUMNS are omitted, CURRENT..LAST is assumed.

COLUMNS or *COLUMN* or *C*

Specifies the range of columns used to form the text address. The integers can be from 1 through 256 or any of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MARK, MAXIMUM.

OVERLAY_LOCATION or *OL*

Specifies the line(s) to be replaced. The default is CURRENT..LAST.

OVERLAY_COLUMNS or *OC*

Specifies the column address for the block of text to be replaced.

BOUNDARY or *B*

Specifies the type of boundary that will limit the erasure. Values can be LINE, STREAM, or BOX. If you omit this parameter and do not specify a value for the COLUMN parameter, LINE is assumed. If you omit this parameter and specify values for both the LINE and COLUMN parameters, STREAM is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following overlays the text on lines 2 and 3 over the text on lines 6 and 7:

```
overlay_text l=2..3 ol=6..7 b=line
```

Before overlay:

```
abcdefghijk
abcdefghijk
abcdefghijk
abcdefghijk
```

```
0987654321
0987654321
0987654321
0987654321
```

After overlay:

```
abcdefghijk
abcdefghijk
abcdefghijk
abcdefghijk
0987654321
abcdefghijk
abcdefghijk
0987654321
```

- The following overlays the first three letters of line 2 through 5 on the first three numbers of lines 9 through 12:

```
o l=2..5 c=1..3 ol=9..12 oc=1..3 b=box
Before overlay:


abcdefghijklmnopqrs
abcdefghijklmnopqrs
bcdefghijklmnopqrs
cdefghijklmnopqrs
abcdefghijklmnopqrs
abcdefghijklmnopqrs
abcdefghijklmnopqrs
1234567890
1234567890
1234567890
1234567890
1234567890
1234567890
```

After overlay:

```
abcdefghijklmnopqrs
abcdefghijklmnopqrs
bcdefghijklmnopqrs
cdefghijklmnopqrs
abcdefghijklmnopqrs
abcdefghijklmnopqrs
abcdefghijklmnopqrs
```

```
1234567890
abc4567890
bcd4567890
cde4567890
abc4567890
1234567890
```

$PARAGRAPH_MARGINS EDIF Function

Purpose Returns an integer specifying the current margin setting. The keyword specified determines the value returned.

Format

$PARAGRAPH_MARGINS or
$PARAGRAPH_MARGIN or
$PM
(PARAMETER_1: keyword)

Parameters

PARAMETER_1

Determines the current margin for which you want a value returned. Values can be LEFT (for the left margin setting), or RIGHT (for the right margin setting), or OFFSET (for the current margin offset).

This parameter is required.

Remarks For more information, see the NOS/VE File Editor manual.

Examples The following example saves and then restores the current margin settings:

```
left_margin = $pm(left)
right_margin = $pm(right)
offset = $pm(offset)
  :
  "temporarily change the values"
  :
set_paragraph_margins mc=left_margin..right_margin ..
      offset=offset
```

POSITION_BACKWARDS EDIF Subcommand

Purpose Moves your position in the file backward a specified number of lines.

Format

POSITION_BACKWARDS or
POSB or
POSITION_BACKWARD or
PB

NUMBER=keyword or *integer*
STATUS=status variable

Parameters *NUMBER* or *N*

Specifies the number of lines to move backward. If you specify ALL, the cursor will be positioned on the first line of the file.

If omitted, 1 is assumed.

Remarks

- This subcommand is typically only used in line mode.
- For more information, see the NOS/VE File Editor manual.

Examples The following example moves the cursor backward 25 lines from the current line.

```
position_backward  number=25
```

POSITION_CURSOR EDIF Subcommand

Purpose Locates text and positions the cursor at the specified line of text. Using this subcommand in screen mode, you can move the cursor to a nontext line.

Format

POSITION_CURSOR or
POSC or
P

TEXT=range of string
NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
COLUMNS=keyword or *range of: keyword* or *integer*
BOUNDARY=keyword

DIRECTION=keyword
UPPER_CASE=boolean
WORD=boolean
REPEAT_SEARCH=boolean
ROW=integer
STATUS=status variable

Parameters *TEXT* or *T*

Specifies a text string at which to position the cursor. If omitted, the new cursor position is determined by the LINE, COLUMNS, and BOUNDARY parameters.

NUMBER or *N*

Specifies the number of times the search is to be repeated. Values can be an integer or the keyword ALL (A).

If NUMBER is omitted, and you have specified a range for the LINES parameter, ALL is assumed.

If NUMBER is omitted and no range has been specified for LINES, 1 is assumed.

LINES or *LINE* or *L*

Specifies one of two things:

- When a single line number is specified, the cursor is positioned at that line.
- When a range of lines is specified, the editor searches for the specified text string within that range of lines.

Values can be an integer, line identifier, or any of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN.

If you specify MARK for this parameter, the values preserved in the mark are used for the COLUMNS and BOUNDARY parameters. Specifying MARK also ensures that the marked file or deck is made current.

If you omit LINE and specify BACKWARD for the DIRECTION parameter, CURRENT..FIRST is assumed.

If both LINE and DIRECTION are omitted, CURRENT..LAST is assumed.

COLUMNS or *COLUMN* or *C*

Specifies the range of columns to be searched to locate the specified text or word. Values can be an integer from 1 through 256 or any of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MARK, MAXIMUM.

When you supply a value, the BOUNDARY parameter assumes a value of STREAM.

If omitted, CURRENT is assumed. If omitted and you have specified LINE=MARK, the marked lines provide the column boundaries. If COLUMN is omitted and you have specified a LINE parameter other than MARK, all columns will be searched.

BOUNDARY or *B*

Specifies the type of boundary that limits the search. Values can be BOX, LINE, or STREAM.

If COLUMNS is specified and BOUNDARY is omitted, STREAM is assumed.

If both BOUNDARY and COLUMNS are omitted, LINE is assumed.

DIRECTION or *D*

Specifies whether to search FORWARD (F) or BACKWARD (B) from the current line.

If no value is specified, FORWARD is assumed.

UPPER_CASE or *UC*

Determines the significance of capitalization in the search.

If you specify TRUE, the editor searches the file as if it were all uppercase.

If you specify FALSE, the editor searches for the text exactly as it was entered.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for UPPER_CASE is used.

WORD or *W*

Determines whether the editor searches for the specified text string as a word (the text you want to search for is surrounded by nonalphanumeric characters).

If you specify TRUE, the editor searches for the text as a word. If you specify FALSE, it doesn't.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for WORD is used.

REPEAT_SEARCH or *RS*

Instructs the editor how to use the values entered for the last TEXT, UPPER_CASE, and WORD parameters.

If you specify TRUE, the editor uses the same TEXT, UPPER_CASE, and WORD parameters as the last time you entered them for any subcommand (unless you have specified values for this subcommand).

If you specify FALSE, the editor uses the parameters entered with the current subcommand.

If omitted, FALSE is assumed.

ROW or *R*

Enables you to move the cursor in relation to the screen instead of in relation to the file text.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following positions the cursor at line 500 of the file:

```
position_cursor l=500
```

- The following moves the current position backward three lines from the current line:

```
posc n=3 d=b
```

- The following moves the cursor to the first column of the next line:

```
posc l=current..last n=2 c=1
```

- The following moves the cursor to the second line of the current screen:

```
posc r=2
```

- The following moves the cursor to the first line in the file:

```
p l=first
```

- The following moves the cursor to the last occurrence of the string *abc* between columns 10 and 20 in lines 200 through the end of the file:

```
p t='abc' l=200..last c=10..20 b=box
```

POSITION_FORWARDS EDIF Subcommand

Purpose Moves your position in the file forward a specified number of lines.

Format

POSITION_FORWARDS or
POSF or
POSITION_FORWARD or
PF
NUMBER=keyword or *integer*
STATUS=status variable

Parameters *NUMBER* or *N*

Specifies the number of lines to move forward. If you specify ALL, the cursor will be positioned on the last line of the file.

If omitted, 1 is assumed.

Remarks

- Using this subcommand, you cannot position past the last line of the file.
- This subcommand is typically only used in line mode.
- For more information, see the NOS/VE File Editor manual.

Examples The following example moves the cursor forward 63 lines from the current line.

```
position_forward number=63
```

PUT_ROW
EDIF Subcommand

Purpose Displays text string on a specified row of the screen.

Format

PUT_ROW or
PUTR
TEXT=string
ROW=integer
STATUS=status variable

Parameters

TEXT or **T**

Specifies the text to be printed. This is a text string from 1 through 256 characters.

This parameter is required.

ROW or *ROWS* or *R*

Indicates the row in which the text will be written. Values can be any integer from 1 through the number of rows available on your screen.

If ROW is omitted, the current line number is assumed.

Remarks

- PUT_ROW is used with procedures to print text on any row on the screen. It enables you to display messages on different lines on the screen.
- You can use the $MESSAGE_ROW function as the value for the ROW parameter. This function causes your text to be placed on the message line of the screen.
- You can use the $HOME_ROW function as the value for the ROW parameter. This function causes your text to be placed on the home line of the screen. The text you place on the home line can be changed before it is executed. You can type over characters, insert characters, and delete characters. Execute the contents of the home line by pressing the return key.
- For more information, see the NOS/VE File Editor manual.

Examples

- In a procedure which defines an alternate set of function key definitions for the CDC721 terminal, you might want to write the message

```
New 721 keys are set.
```

in the message row. To do this, include the following subcommand in the procedure:

```
put_row text='New 721 keys are set.' row=$mr
```

- The following example defines the fifth key in your menu of operations to execute a PUT_ROW subcommand. The subcommand places the string you executed as the last operation on the home line. The last operation is returned by using the $LAST_COMMAND function. The label for the operation in the menu of operations becomes Repeat.

```
ef/set_function_key number=5 command_string=..
ef/'put_row text=$last_command row=$home_row' ..
ef/label='Repeat'
```

The PUT_ROW subcommand as shown in the preceding example does not become the last command. This allows you to repeat the last operation more than once.

READ_FILE EDIF Subcommand

Purpose Inserts the text of another file into the current file.

Format

READ_FILE or
REAF
FILE=file
INSERTION_LOCATION=keyword or *integer* or *line_identifier*
PLACEMENT=keyword
MULTI_PARTITION=boolean
STATUS=status variable

Parameters

FILE or F

Specifies the name of the file from which the text is to be inserted. The entire file will be inserted. This parameter is required.

INSERTION_LOCATION or *IL*

Specifies the line before or after which the text is to be inserted (depending on the value of the PLACEMENT parameter). Values can be an integer, line identifier, or one of the LINE keywords: CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN. Ranges are not allowed.

If omitted, CURRENT is assumed.

PLACEMENT or *P*

Specifies whether the insertion is to occur BEFORE (B) or AFTER (A) the line specified by the INSERTION_LOCATION parameter.

If omitted, AFTER is assumed.

MULTI_PARTITION or *MP*

Specifies whether the editor is to change end-of-partition delimiters to WEOP directives when an external file is copied to the current working file.

If you specify TRUE, the editor changes end-of-partition delimiters to WEOP directives and reads the entire file.

If you specify FALSE, no change takes place and the editor stops reading at the first partition.

If omitted, FALSE is assumed.

Remarks

- The READ_FILE subcommand reads the external copy of the specified file. If you have been editing a file within the editor and have not made the changes permanent using the WRITE_FILE subcommand and then specify that file on a READ_FILE subcommand, an external copy is inserted, not the changed working copy.

- For more information, see the NOS/VE File Editor manual.

Examples

- The following inserts the contents of file *ALPHA* into the current file immediately after line 320:

```
read_file file=alpha insertion_location=320
```

- The following inserts the contents of file *BETA* into the current file immediately before the last marked line:

```
reaf f=beta il=last_mark p=b
```

REPLACE_LINES EDIF Subcommand

Purpose Replaces lines of text by deleting the old text and replacing it with the text you specify.

Format

REPLACE_LINES or
REPLACE_LINE or
REPL
TEXT=range of string
NEW_TEXT=string
NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
UNTIL=string
UPPER_CASE=boolean
WORD=boolean
REPEAT_SEARCH=boolean
STATUS=status variable

Parameters

TEXT or *T*

Specifies the text you want to replace.

If a range of text is specified, the lines containing the entire range are replaced with the string supplied in the NEW_TEXT parameter.

If TEXT is omitted, the LINE and NUMBER parameters determine the lines to be replaced.

NEW_TEXT or *NT*

Specifies the new line of text that is to replace the specified line.

If this parameter is omitted, you are prompted to enter text line by line until the editor encounters the character(s) specified by the UNTIL parameter.

NUMBER or *N*

Specifies the number of lines to replace. Values can be a number or the keyword ALL.

If a range of text is specified, NUMBER indicates the number of blocks of text to replace.

If you omit this parameter and specify a range for the LINE parameter, the assumed value is ALL.

If you omit this parameter and do not specify a range for the LINE parameter, the assumed value is 1.

LINES or *LINE* or *L*

Specifies a range of lines in which the replacement is to occur. Values can be an integer, line identifier, or one of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN. If a single value is specified, only that line is replaced.

If omitted, CURRENT..LAST is assumed.

UNTIL or *U*

In line mode, specifies a string that stops the replacement text.

If NEW_TEXT is omitted, you are prompted to enter input until the editor encounters the character(s) you specify with the UNTIL parameter.

If the UNTIL parameter is omitted, ** is assumed.

UPPER_CASE or *UC*

Determines the significance of capitalization in the search.

If you specify TRUE, the editor searches the file as if it were all uppercase.

If you specify FALSE, the editor searches for the text exactly as it was entered.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for UPPER_CASE is used.

WORD or *W*

Determines whether the editor searches for the specified text string as a word (the text you want to search for is surrounded by nonalphanumeric characters).

If you specify TRUE, the editor searches for the text as a word. If you specify FALSE, it doesn't.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for WORD is used.

REPEAT_SEARCH or *RS*

Instructs the editor how to use the values entered for the last TEXT, UPPER_CASE, and WORD parameters.

If you specify TRUE, the editor uses the same TEXT, UPPER_CASE, and WORD parameters as the last time you entered them for any subcommand (unless you have specified values for this subcommand).

If you specify FALSE, the editor uses the parameters entered with the current subcommand.

If omitted, FALSE is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following replaces lines 30 to the end of the file with a line that says *text*:

```
replace_line new_text='text' line=30..last
```

- The following replaces the current line with text you are prompted to enter until the editor encounters ** at the end of one of the replacement lines.

```
repl
```

REPLACE_TEXT EDIF Subcommand

Purpose Replaces blocks of text.

Format

REPLACE_TEXT or
REPT or
R

TEXT=string
NEW_TEXT=string
NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
COLUMNS=keyword or *range of: keyword* or *integer*
BOUNDARY=keyword
UPPER_CASE=boolean
WORD=boolean
REPEAT_SEARCH=boolean
VETO=boolean
STATUS=status variable

Parameters

TEXT or *T*

Specifies the text string to replace in the specified block of text.

If omitted, you must specify TRUE for the REPEAT_SEARCH parameter.

NEW_TEXT or *NT*

Specifies the replacement text for the string specified in the TEXT parameter.

If omitted, the string specified in the TEXT parameter is deleted.

NUMBER or *N*

Specifies the number of times the original text is to be replaced within the block of text. Values can be any integer or the keyword ALL (A).

If you omit this parameter and specify a range of values for the LINE parameter, the assumed value is ALL.

If you omit this parameter and do not specify a range for the LINE parameter, the assumed value is 1.

LINES or *LINE* or *L*

Specifies the range of lines affected by the replacement. Values can be an integer, line identifier, or one of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN. If a single value is specified, only the number of occurrences of text are replaced in that line.

If you specify MARK for this parameter, the values preserved in the mark are used for the COLUMNS and BOUNDARY parameters. Specifying MARK also ensures that the marked deck or file is made current.

If omitted, CURRENT..LAST is assumed.

COLUMNS or *COLUMN* or *C*

Specifies the range of columns affected by the replacement. The integers can be from 1 through 256 or any of the keywords: CURRENT, FIRST_MARK, LAST_MARK, MARK, MAXIMUM.

With the COLUMN parameter you can specify a beginning and ending column for the replacement. When you specify a boundary of STREAM, the search for replacement starts at the beginning column on the beginning line, continues through all columns of the next lines, and stops at the end column of the ending line.

If COLUMN, BOUNDARY, and LINE are omitted and NUMBER=ALL, the replacement search starts at the current column of the current line and ends at the last column of the last line. If COLUMN is omitted and LINE is specified, the replacement search uses all columns of the lines specified. If COLUMN, BOUNDARY, LINE, and NUMBER are omitted, the current column is assumed.

BOUNDARY or *B*

Specifies the type of boundary that will limit the replacement. Values can be BOX, LINE, or STREAM.

If BOUNDARY and COLUMNS are both omitted, LINE is assumed.

If BOUNDARY is omitted but COLUMNS is specified, STREAM is assumed.

UPPER_CASE or *UC*

Determines the significance of capitalization in the search.

If you specify TRUE, the editor searches the file as if it were all uppercase.

If you specify FALSE, the editor searches for the text exactly as it was entered.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for UPPER_CASE is used.

WORD or *W*

Determines whether the editor searches for the specified text string as a word (the text you want to search for is surrounded by nonalphanumeric characters).

If you specify TRUE, the editor searches for the text as a word. If you specify FALSE, it doesn't.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for WORD is used.

REPEAT_SEARCH or *RS*

Instructs the editor how to use the values entered for the last TEXT, UPPER_CASE, and WORD parameters.

If you specify TRUE, the editor uses the same TEXT, NEW_TEXT, UPPER_CASE, and WORD parameters as the last time you entered them for any subcommand (unless you have specified values for this subcommand).

If you specify FALSE, the editor uses the parameters entered with the current subcommand.

If omitted, FALSE is assumed.

VETO or *V*

Enables you to display a directory of replaced lines allowing you to choose a line at which you want the cursor to be positioned. Allows you to veto any of the displayed lines affected by the subcommand.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following changes the first occurrence of *ALPHA* to *BETA* from the current line and column to the last line and column:

```
replace_text t='alpha' nt='beta'
```

- The following uses the same values for TEXT, NEW_TEXT, UPPER_CASE, and WORD parameters specified on a previous subcommand:

```
rept rs=true
```

- The following replaces all occurrences of *BETA* with *ALPHA* from line 50 to the end of the file:

```
r t='BETA' nt='ALPHA' l=50..last
```

- The following replaces the first occurrence of *$* with *#* from the current line and column to the last line and column:

```
rept t='$' nt='#'
```

- The following deletes the text *CHI* in all lines of the file:

```
r t='chi' l=a
```

- The following changes *r* to *t* starting at line 2 column 5, and ending at line 4 column 3.

```
rept 'r' 't' l=2..4 c=5..3
```

The following occurs:

Before replacement:	After replacement:
rrrrrrr	rrrrrrr
rrrrrrr	rrrrttt
rrrrrrr	ttttttt
rrrrrrr	tttrrrr

- The following changes *r* to *t* in columns 3 to 5 of lines 2 through 4.

```
r 'r' 't' l=2..4 c=3..5 b=box
```

The following occurs:

Before replacement:	After replacement:
rrrrrrr	rrrrrrr
rrrrrrr	rrtttrr
rrrrrrr	rrtttrr
rrrrrrr	rrtttrr

- Use the SET_MASK subcommand to leave characters unaltered between those that are being replaced. When the specified number of characters has been masked, any additional characters are displayed as blank.

  The following sets the mask character as '#' and then replaces *FUN* with *FUN & PROFIT*:

```
set_mask c='#'
rept t='FUN' nt='F###&#PROFIT'
```

RESET_FILE EDIF Subcommand

Purpose Cancels all the changes you have made to your current file since you last accessed the file using the EDIT_FILE command.

Format

RESET_FILE or
RESF
STATUS=status variable

Remarks

- The RESET_DECK subcommand discards changes to decks.
- For more information, see the NOS/VE File Editor manual.

RESTORE_POSITION
EDIF Subcommand

Purpose Enables you to return to the position saved by the SAVE_POSITION subcommand.

Format
RESTORE_POSITION or
RESP
STATUS=status variable

Remarks For more information, see the NOS/VE File Editor manual.

$ROW_TEXT
EDIF Function

Remarks Reserved for site personnel, Control Data, or future use.

SAVE_POSITION
EDIF Subcommand

Purpose Enables you to save the current column, line, and file name for reference later.

Format
SAVE_POSITION or
SAVP
STATUS=status variable

Remarks
- To return to this position later, use the RESTORE_POSITION subcommand.
- For more information, see the NOS/VE File Editor manual.

$SCREEN_ACTIVE
EDIF Function

Purpose Returns a boolean value. It is TRUE if screen mode is active, and FALSE if it is not.

Format
$SCREEN_ACTIVE or
$SA

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

$SCREEN_INPUT EDIF Function

Purpose Returns the text you enter on the subcommand line as the string.

Format

$SCREEN_INPUT or
$SI
(*PARAMETER_1: string*)

Parameters *PARAMETER_1*

The text you want displayed on the message row as a prompt for input. If omitted, ENTER TEXT is used as the prompt.

Remarks

- This function allows an SCL procedure to pause and request input.
- When the LOCATE_TEXT subcommand is executed, the user provides the text normally, without concern that it will become a string. The user does not put apostrophes around the text or use double apostrophes within the text.
- Trailing blanks are not truncated.
- For more information, see the NOS/VE File Editor manual.

Examples

- The following subcommand locates whatever text the user provides in response to $SI.

```
locate_text t=$si('What do you want to locate?')
```

- The following subcommand programs key 7 to insert whatever characters are specified:

```
set_function_key number=7 ..
  command_string='insc '//$quote($SI(..
  'characters to be inserted by key 7'))
```

$SEARCH_MARGINS
EDIF Function

Purpose Returns an integer specifying the column number of either the right or left margin. The keyword specified determines the value returned.

Format

$SEARCH_MARGINS or
$SEARCH_MARGIN or
$SM
(PARAMETER_1: keyword)

Parameters **PARAMETER_1**

Specifies the margin for which you want a value returned. Values can be LOW (for the left margin) or HIGH (for the right margin).

This parameter is required.

Remarks

- The function can be used to save the values for the current search margins so they can be temporarily altered.
- For more information, see the NOS/VE File Editor manual.

SET_EPILOG
EDIF Subcommand

Purpose Specifies a file containing editor subcommands you want executed each time you leave the editor.

Format

SET_EPILOG or
SETE
FILE=file
STATUS=status variable

Parameters *FILE* or *F*

Specifies the file to contain the editor subcommands. If omitted, $USER.SCU_EDITOR_EPILOG is assumed.

Remarks

- If you do not enter a SET_EPILOG subcommand within your editing session, no epilog file is executed.
- You can enter this command anytime within your editing session.
- If you want epilog file processing to occur automatically, put the SET_EPILOG subcommand into your prolog file.
- For more information, see the NOS/VE File Editor manual.

Examples

The following process always leaves your screen display at 132 columns after stopping the editor:

1. Place the following in file $USER.SCU_EDITOR_EPILOG.

```
if $screen_active then;setso c=132;ifend
```

2. Include the following subcommand in your prolog file.

```
set_epilog
```

SET_FUNCTION_KEY EDIF Subcommand

Purpose

Enables you to create your own set or sets of function keys.

Format

SET_FUNCTION_KEY or
SETFK
NUMBER=keyword or **integer**
COMMAND_STRING=string
SHIFT=boolean
LABEL=string
STATUS=status variable

Parameters

NUMBER or **N**

Specifies the number of the key to be defined. Values can be any integer from 1 through 16. These numbers correspond to the highlighted boxes in the menu of operations at the bottom of the screen. The numbers 1 through 8 correspond to the first row of boxes; 9 through 16 correspond to the second row of boxes.

You can also specify one of the following keywords: DOWN(D), EDIT(E), FWD(F), BKW(B), BACK, HELP(H) STOP(S), UNDO, UP(U).

The keywords relate to keys on some terminals. If your terminal has defined sequences that relate to these keywords, you can create your own function keys using these keywords.

This parameter is required.

COMMAND_STRING or CS

Specifies the subcommand(s) to be executed when the specified key is pressed. Values can be any editor or SCL command. When more than one subcommand is specified, separate them with semicolons.

This parameter is required.

SHIFT or S

For those terminals that have one key identifier next to each highlighted box in the menu of operations, the SHIFT parameter indicates whether the key to be used is shifted. Specify TRUE for the shifted key and FALSE for the nonshifted key.

For those terminals that have two key identifiers next to each highlighted box in the menu of operations, the SHIFT parameter indicates which key you use.

Specify TRUE to use the key corresponding to the top key identifier. Specify FALSE to use the key corresponding to the bottom key identifier.

If SHIFT is omitted, FALSE is assumed.

LABEL or L

Specifies a string as the label that is to appear in the menu of operations for the specified key.

If LABEL is omitted, current label remains the same.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following SET_FUNCTION_KEY subcommand defines the shifted F5 key to execute the HELP subcommand. The key has a screen label of help:

```
set_function_key n=5 cs='help' s=true l='help'
```

- The online Examples manual lists a number of useful function key definitions.

SET_LINE_WIDTH EDIF Subcommand

Purpose Specifies the maximum line length. When a line exceeds this limit, a warning message is displayed.

Format

SET_LINE_WIDTH or
SETLW
WIDTH=integer
STATUS=status variable

Parameters

WIDTH or W

Specifies the number of characters you can have on one line before the editor sends you a message. Values can be an integer from 0 through 256. Specifying 0 eliminates the message and adds no trailing blanks to lines. When you create a file, an initial width value of 0 is assumed. For decks the value is taken from the deck header information.

This parameter is required.

Remarks

- Each time you edit a file, you must enter the SET_LINE_WIDTH subcommand to be warned when lines exceed a given length.
- Once this command is entered, the editor adds trailing spaces to lines with a character count less than the limit when making string comparisons.
- You can locate long lines using the LOCATE_WIDE_LINES subcommand.
- For more information, see the NOS/VE File Editor manual.

Examples The following subcommand sets the line width limit at 80:

```
set_line_width width=80
```

SET_LIST_OPTIONS EDIF Subcommand

Purpose Provides you with the options in line mode of either displaying the line identifier on the same line as the text, on a separate line from the text, or not at all.

Format

SET_LIST_OPTIONS or
SET_LIST_OPTION or
SETLO
LINE_IDENTIFIER=keyword
STATE=boolean
STATUS=status variable

Parameters *LINE_IDENTIFIER* or *LI*

Specifies where or if the identifier is to be displayed. Values can be LEFT (L), SEPARATE (S), or NONE.

If LINE_IDENTIFIER is omitted, NONE is assumed.

STATE or *S*

Specifies whether the state of the modification associated with the line's introduction is to be displayed.

If TRUE, the state is displayed.

If omitted, FALSE is assumed.

Remarks

- This subcommand is usually entered when you are line editing decks and want to see the line identifiers.
- Modification states are described in the NOS/VE Source Code Management manual.
- For more information, see the NOS/VE File Editor manual.

SET_MASK
EDIF Subcommand

Purpose Defines a special character to match any other character. The character you define serves as a wild card character. You can use it when specifying a value for any TEXT parameter on file editor subcommands.

Format

SET_MASK or
SETM
CHARACTER=keyword or **string**
STATUS=status variable

Parameters

CHARACTER or **C**

Specifies the mask character. Values can be any alphanumeric character or the keyword NONE. If NONE is specified, the mask feature is turned off.

This parameter is required.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- This example uses a mask character to replace the following strings:

```
Ford Fred Food Find Fund
```

Specify the mask character and the REPLACE_TEXT subcommand as follows:

```
set_mask character='#'
rept t='F##d' nt='Feed' l=all
```

The following occurs:

Before replacement:	After replacement:
Ford	Feed
Fred	Feed
Food	Feed
Find	Feed
Fund	Feed

- This example uses a mask character to replace the following strings:

```
Ford Find Fund
```

Specify the mask character and the REPLACE_TEXT subcommand as follows:

```
set_mask character='#'
rept t='F##d' nt='P##t' l=all
```

The following occurs:

Before replacement:	After replacement:
Ford	Port
Find	Pint
Fund	Punt

- Depending on its position in the new text string, you can use the mask character to shorten or lengthen text strings, replace characters, or insert spaces in a text string. For example, if you specified # as the mask character, you can use it in the NEW_TEXT parameter with the following results:

```
rept t='fund' nt='f##'
```

Replaces *fund* with *fun.*

```
rept t='fund' nt='f##k'
```

Replaces *fund* with *funk.*

```
rept t='fund' nt='f###amental'
```

Replaces *fund* with *fundamental.*

- When you use the SET_MASK subcommand with REPLACE_TEXT, the system substitutes the mask characters in the new text until it reaches the end of the old text. Additional mask characters in the new text display as blanks. The following (in which the mask character is '#') replaces *FUN* with *FUN & PROFIT*:

```
rept t='FUN' nt='F###&#PROFIT'
```

SET_PARAGRAPH_MARGINS EDIF Subcommand

Purpose

Changes the paragraph margins. In any subsequent FORMAT_ PARAGRAPH or CENTER_LINE subcommands, the margins set with SET_PARAGRAPH_MARGINS are used.

Format

SET_PARAGRAPH_MARGINS or
SET_PARAGRAPH_MARGIN or
SETPM
MARGIN_COLUMNS=range of integer
OFFSET=integer
STATUS=status variable

Parameters

MARGIN_COLUMNS or *MARGIN_COLUMN* or *MC*

Specifies the left and right margins. If just one column number is specified, the left margin is set to that number.

If omitted and you have not specified this subcommand previously in your editing session, columns 1 and 65 are used. If you have specified the subcommand previously, any parameter not specified is not changed.

OFFSET or *O*

Specifies the number of columns the first line in the paragraph is to be offset from the rest of the lines in the paragraph. If the number specified is a positive number, the first line of the paragraph is indented the number of columns specified. If zero is specified, the first line is not indented. If a negative value is given, the first line begins to the left of the rest of the paragraph.

If omitted and you have specified this subcommand during this terminal session, the previous value is used. If you have not entered this subcommand previously and omit the OFFSET parameter, 4 is assumed.

Remarks

- You can use the $PARAGRAPH_MARGINS function to return paragraph margin values.
- For more information, see the NOS/VE File Editor manual.

Examples

- To set the paragraph margins to columns 7 and 72, with an offset of 4, enter:

```
set_paragraph_margins margin_columns=7..72
```

- To set the margins to 10 and 70 and also specify that you want the first line of the paragraph indented 5 columns, enter:

```
setpm mc=10..70 o=5
```

SET_SCREEN_OPTIONS EDIF Subcommand

Purpose Enables you to change the way the screen appears. Among other things, you can change the number of lines that are listed on your screen, the number of rows in the menu of operations that is displayed, the number of files displayed at one time, and the number of columns displayed.

Format

SET_SCREEN_OPTIONS or
SET_SCREEN_OPTION or
SETSO
MODEL=name
COLUMNS=integer
MENU_ROWS=integer
ROWS=integer
SPLITS=integer
SPLIT_SIZES=list of integer
STATUS=status variable

Parameters *MODEL* or *M*

Specifies the type of terminal you are using. Valid entries are:

Entry	Terminal
MAC_CONNECT_10	Apple Macintosh running version 1.0 or 1.0+ of Control Data CONNECT for the Macintosh
MAC_CONNECT_11	Apple Macintosh running version 1.1 of Control Data CONNECT for the Macintosh

MAC_CONNECT_20	Apple Macintosh running version 2.0 of Control Data CONNECT for the Macintosh
MAC_CONNECT_21	Apple Macintosh running version 2.1 of Control Data CONNECT for the Macintosh
MAC_CONNECT_22	Apple Macintosh running version 2.2 of Control Data CONNECT for the Macintosh
PC_CONNECT_10	IBM PC or equivalent running version 1.0 of Control Data CONNECT for the IBM PC
PC_CONNECT_11	IBM PC or equivalent running version 1.1 of Control Data CONNECT for the IBM PC
PC_CONNECT_12	IBM PC or equivalent running version 1.2 of Control Data CONNECT for the IBM PC
PC_CONNECT_13	IBM PC or equivalent running version 1.3 or 1.4 of Control Data CONNECT for the IBM PC
PC_CONNECT_20	IBM PC or equivalent running version 2.0 of Control Data CONNECT for the IBM PC
IBM_3270	IBM 3270 with 24 x 80 screen
IBM_3270_2	IBM 3270 with 24 x 80 screen
IBM_3270_3	IBM 3270 with 32 x 80 screen
IBM_3270_4	IBM 3270 with 43 x 80 screen
IBM_3270_5	IBM 3270 with 27 x 132 screen
DEC_VT100	Digital Equipment VT100 with 18 function keys
DEC_VT100_GOLD	Digital Equipment VT100 with 32 function keys

DEC_VT220	Digital Equipment VT220 for users logging in through CDCNET
TV_955	Televideo 955 with full editing capability
TV_955_PROTECTED	Televideo 955 with form entry access only

If the MODEL parameter has not been specified on an earlier subcommand of the editing session, or by a CHANGE_TERMINAL_ ATTRIBUTES TM=name command previous to the editing session, it is required.

This parameter is effective in identifying your terminal only when you enter the SET_SCREEN_OPTIONS subcommand in line mode. If you are editing in screen mode, enter the DEACTIVATE_SCREEN subcommand first and then enter the SET_SCREEN_OPTIONS subcommand. To resume screen editing, enter the ACTIVATE_SCREEN subcommand.

COLUMNS or *COLUMN* or *C*

Specifies the number of columns to be displayed. Values range from 1 to the maximum number allowed on your terminal. The number you enter is compared to the screen sizes set up in the terminal definition for your terminal. The number of columns displayed is the closest number as large or larger than the number you enter on the COLUMNS parameter.

Each time the editor is entered, a value of 80 columns is assumed.

If COLUMN is omitted, the number of columns displayed remains the same.

MENU_ROWS or *MENU_ROW* or *MR*

Specifies the number of rows of the menu of operations prompts to display. Values can be:

0	Does not display the menu of operations.
1	Displays 1 row of highlighted boxes from the menu of operations.
2	Displays 2 rows of the menu.

If MENU_ROW is omitted, the number of rows displayed remains the same. When starting the editor, 1 row is displayed.

ROWS or *ROW* or *R*

Specifies the number of rows to display for terminals that support multiple screen sizes. Values can be from 10 to the maximum number allowed for your terminal. The number you enter is compared to the screen sizes set up in the terminal definition for your terminal. The number of rows displayed is the closest number as large or larger than the number you enter on the ROWS parameter.

When you first enter the editor, it assumes a value of 32.

Not all terminals support multiple screen sizes.

SPLITS or *SPLIT* or *S*

Specifies the number of areas of text (splits) you want displayed on the screen when the screen is divided horizontally to show more than one file. This number determines how many files you can display at the same time. Values are 1 through 16.

Each time the editor is entered, a value of 1 is assumed.

If SPLIT is omitted, the number of splits remains the same.

SPLIT_SIZES or *SPLIT_SIZE* or *SS*

Specifies the number of lines you want displayed within a particular area of text (split). The value(s) you specify correspond positionally to the splits displayed; the first value you specify corresponds to the topmost split, the second value to the next lowest split and so on. Values are 2 through 255.

If SPLIT_SIZE is omitted, each split contains an equal number of lines.

Remarks

- For all omitted parameters, the editor assumes you want the same value used the last time you entered the SET_SCREEN_OPTIONS subcommand.
- For more information, see the NOS/VE File Editor manual.

Examples

- The following example displays an additional file onto a screen. The new screen contains two split areas with a different file in each area.

  1. Press **Home** and enter:

     ```
     set_screen_options split=2
     ```

  2. Move the cursor to the split in which you want the new file (ZETA) to appear.

  3. Press **Home** and enter:

     ```
     edif zeta
     ```

     File ZETA appears in the split area the cursor was last in.

- The following example displays all of your menu of operations:

  ```
  setso mr=2
  ```

SET_SEARCH_MARGINS EDIF Subcommand

Purpose Limits the number of columns to be searched in subsequent subcommands that use string searches.

Format

SET_SEARCH_MARGINS or
SET_SEARCH_MARGIN or
SETSM
MARGIN_COLUMNS=range of integer
STATUS=status variable

Parameters *MARGIN_COLUMNS* or *MC*

Specifies the column(s) in which to perform the search. Values can be any number or any of the COLUMN keywords: CURRENT, FIRST_MARK, LAST_MARK, MARK, MAXIMUM. If you specify two values, the search is done from the first column through the last column specified. If you specify a single integer, only that column is searched.

If MARGIN_COLUMN is omitted, columns 1 through 256 are assumed.

Remarks

- The $SEARCH_MARGINS function can be used to return the MARGIN_COLUMNS values.
- This subcommand can be used with the REPLACE_TEXT subcommand to change a string within a limited range of columns for many lines.
- For more information, see the NOS/VE File Editor manual.

Examples To set the search margins to columns 1 and 7, enter:

```
set_search_margins margin_columns=1..7
```

SET_TAB_OPTIONS EDIF Subcommand

Purpose Sets a tab character and the columns in which you want tabs set.

Format

SET_TAB_OPTIONS or
SET_TAB_OPTION or
SETTO
CHARACTER=string
TAB_COLUMN=list of integer
STATUS=status variable

Parameters

CHARACTER or *C*

Specifies the tab character. Values can be any character. The horizontal tab character, $char(9), works well as a value.

When you enter a tab character within text typed from your terminal, the tab character moves any text from the current position to the next tab setting.

If you enter a tab character after the last tab column, the tab character is included as part of the file text.

When you start editing a file, the tab character is set to the reverse slant. When you start editing a deck, the tab character is set as specified in the deck header (refer to the CREATE_DECK SCU subcommand in the NOS/VE Source Code Management manual).

If CHARACTER is omitted, the tab character is not changed.

TAB_COLUMN or *TAB_COLUMNS* or *TC*

Specifies tab columns to be added to those already selected.

A maximum of 256 columns can be specified as tab columns. Values can be any integer from 1 through 256 and must be enclosed in parentheses. When you start editing a file, the tabs are set at columns 1, 7, and 72.

When you start editing a deck, the tab columns selected are those specified in the deck header (refer to the CREATE_DECK SCU subcommand in the NOS/VE Source Code Management manual).

If TAB_COLUMN is omitted, the tab settings are not changed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following sets the tab character to] and adds columns 11, 18, 41 and 53 as tab columns:

```
set_tab_options character=']' ..
    tab_column=(11,18,41,53)
```

- The following sets the tab character to ! and adds column 3 as a tab column:

```
setto c='!' tc=(3)
```

- The following sets the tab character to \ and inserts the text 'line1' preceded by six blank spaces:

```
set_tab_options c='\' tc=(7 10)
insert_line
\line1**
```

SET_VERIFY_OPTION
EDIF Subcommand

Purpose Displays lines that have been changed using the REPLACE_TEXT subcommand and displays the first and last lines of a block of text located with the LOCATE_TEXT subcommand.

Format

SET_VERIFY_OPTION or
SETVO
ECHO=boolean
STATUS=status variable

Parameters

ECHO or **E**

Specifies whether you want the verify option on or off.

This parameter is required.

Remarks

- In screen mode the verify option is always off.
- Inside your editor prolog, the value for the ECHO parameter is FALSE. The verify option is off while your editor prolog is processed unless you specifically turn it on. When you turn the verify option on, the editor displays changed lines.

  Outside your editor prolog, the default value for the ECHO parameter is TRUE. The verify option is on in line mode unless you specifically turn it off. You can set the verify option in your editor prolog or while you are editing a file.
- The function $VERIFY_OPTION returns the current value of the verify option.
- For more information, see the NOS/VE File Editor manual.

SET_WORD_CHARACTERS EDIF Subcommand

Purpose

Enables you to add or delete allowable characters (within words) for use with the WORD parameter.

Format

SET_WORD_CHARACTERS or
SET_WORD_CHARACTER or
SETWC
ADD=list of string
DELETE=list of string
STATUS=status variable

Parameters

ADD or *A*

Specifies the characters to add as allowable characters. Values can be any printable character. The space character cannot be specified as an allowable character.

Enclose each character in quotes and all inside parentheses.

If ADD is omitted, no characters are added.

DELETE or *D*

Specifies the characters to delete as allowable characters in a word. In other words, characters specified by this parameter will be treated as punctuation marks. Values can be any printable character. The space character is not allowed.

Enclose each character in quotes and all inside parentheses.

If DELETE is omitted, no characters are deleted.

Remarks

- The initial word characters consist of the alphanumerics plus the underscore (_), dollar-sign ($), number-sign (#), and at-sign (@).
- If you specify more than one character, separate them with commas or spaces.
- For more information, see the NOS/VE File Editor manual.

Examples

- The following adds % as an allowable word character and deletes x as an allowable word character:

```
set_word_characters add='%' delete='x'
```

- The following changes the characters allowed in words to those used in the NOS/VE COBOL compiler:

```
setwc a='-' d='$' '#' '_' '@'
```

$SPLIT_SIZE
EDIF Function

Purpose Returns an integer specifying the number of available text lines for the specified split of the screen.

Format
$SPLIT_SIZE or
$SS
(*PARAMETER_1: integer*)

Parameters *PARAMETER_1*

Specifies the split of the screen for which you want a value returned. If omitted, the current split is assumed.

Remarks
- If you are in line mode, zero is returned.
- For more information, see the NOS/VE File Editor manual.

$TEXT
EDIF Function

Purpose Returns a string specifying the last text you specified for a TEXT parameter.

Format
$TEXT or
$T

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

$TITLE_ROW
EDIF Function

Purpose Returns an integer specifying the row number of the title row (file header) used for the specified split of the screen.

Format
$TITLE_ROW or
$TR
(*PARAMETER_1: integer*)

Parameters *PARAMETER_1*

Specifies the split of the screen for which you want a value returned. If omitted, the current split is assumed.

Remarks

- If you are in line mode, zero is returned.
- For more information, see the NOS/VE File Editor manual.

UNDO
EDIF Subcommand

Purpose Cancels changes in reverse chronological order. Entire transactions are undone until one that included a change is reached.

Format

UNDO or
UND
STATUS=status variable

Remarks

- A transaction consists of all changes made between two presses of the return key.
- The following terminals include an automatic return when you press keys that perform editing operations:

  IBM PC
  Apple Macintosh

  At these terminals, pressing keys that perform editing operations marks the end of a transaction. At other terminals, you press return to end transactions that include editing operations.
- Use the UNMARK subcommand to cancel marks.
- For each UNDO subcommand, all changes made since the last time you pressed the return key are canceled.
- You can undo only changes made to the current file. You can, however, make any file that was edited during this session the current file if it has not been closed with END_FILE, END_DECK, or a SELECT_DECK subcommand. You can do this by entering the EDIT_FILE or EDIT_DECK subcommand, or, if your screen is split, by positioning the cursor in the file you want to be the current file.

- To undo all changes you have made since opening the current file, use the RESET_FILE subcommand.
- For more information, see the NOS/VE File Editor manual.

Examples The following changes were made to a file in the order given:

1. Five lines in the file were deleted using one DELETE_LINES subcommand.
2. The next three lines are displayed using the LOCATE_TEXT subcommand.
3. A new line is entered using the INSERT_LINES subcommand.

Each time UNDO is entered, the following changes are undone:

1. The first time UNDO is entered, the new line inserted is deleted.
2. The second time, the five lines deleted are returned.

UNMARK
EDIF Subcommand

Purpose Explicitly cancels the marks on any lines or characters you previously marked.

Format

UNMARK or
UNM
STATUS=status variable

Remarks

- You implicitly unmark text by marking a new region of text, by deleting marked text, or by entering the undo operation (or UNDO subcommand), which undoes the most recent change as well as undoing any current marks.
- When you enter the END_FILE subcommand you can close a file containing the marked text.
- For more information, see the NOS/VE File Editor manual.

$UPPER_CASE
EDIF Function

Purpose Returns a boolean value specifying the most recent value supplied for an UPPER_CASE parameter.

Format **$UPPER_CASE** or
$UC

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

$VERIFY_OPTION
EDIF Function

Purpose Returns a boolean value indicating whether the VERIFY option has been activated (TRUE) or not (FALSE).

Format **$VERIFY_OPTION** or
$VO

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

$WORD
EDIF Function

Purpose Returns a boolean value indicating whether the word search feature is active (TRUE) or not (FALSE).

Format **$WORD** or
$W

Parameters None.

Remarks For more information, see the NOS/VE File Editor manual.

WRITE_FILE
EDIF Subcommand

Purpose Copies text from the current working file to the external copy of a file.

Format

WRITE_FILE or
WRIF
TEXT=range of string
NUMBER=keyword or *integer*
LINES=keyword or *range of: keyword* or *integer* or *line_identifier*
FILE=file
UPPER_CASE=boolean
WORD=boolean
REPEAT_SEARCH=boolean
MULTI_PARTITION=boolean
STATUS=status variable

Parameters *TEXT* or *T*

Specifies string(s) of text in the first and last lines of a block of text to be written.

If you enter only one string, the block of text to be written will contain only one line. If you enter two strings, the search for the second begins immediately after the first is found and the cursor is positioned at the beginning of the first string.

If omitted, the lines to be written are determined by the NUMBER, LINE, and DIRECTION parameters or by the REPEAT_SEARCH parameter.

NUMBER or *N*

Specifies the number of blocks of text to be copied. Values for this parameter can be an integer or the keyword ALL (A).

If NUMBER is omitted, ALL is assumed.

LINES or *LINE* or *L*

Specifies a range of lines to be searched to locate the text to be copied. Values can be an integer, line identifier, or one of the keywords: ALL, CURRENT, FIRST, FIRST_MARK, FIRST_SCREEN, LAST, LAST_MARK, LAST_SCREEN, MARK, SCREEN.

If a single value is specified, only that line is searched.

If LINE is omitted, ALL is assumed.

FILE or *F*

Specifies the file to which the text is to be copied.

If the object you are editing is a file and FILE is omitted, the editor writes the file to the external file from which the working file was made.

If the object you are editing is a deck, this parameter is required.

UPPER_CASE or *UC*

Determines the significance of capitalization in the search.

If you specify TRUE, the editor searches the file as if it were all uppercase.

If you specify FALSE, the editor searches for the text exactly as it was entered.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for UPPER_CASE is used.

WORD or *W*

Determines whether the editor searches for the specified text string as a word (the text you want to search for is surrounded by nonalphanumeric characters).

If you specify TRUE, the editor searches for the text as a word. If you specify FALSE, it does not.

If omitted, FALSE is assumed unless you specify TRUE for REPEAT_SEARCH. In this case, your last value for WORD is used.

REPEAT_SEARCH or *RS*

Instructs the editor how to use the values entered for the last TEXT, UPPER_CASE, and WORD parameters.

If you specify TRUE, the editor uses the same TEXT, UPPER_CASE, and WORD parameters as the last time you entered them for any subcommand (unless you have specified values for this subcommand).

If you specify FALSE, the editor uses the parameters entered with the current subcommand. If omitted, FALSE is assumed.

MULTI_PARTITION or *MP*

Specifies whether the editor is to change WEOP directives to end-of-partition delimiters when the current working file is copied to an external file.

If TRUE, the editor changes WEOP directives to end-of partition delimiters. If FALSE, no substitution takes place. If omitted, FALSE is assumed.

Remarks For more information, see the NOS/VE File Editor manual.

Examples

- The following copies 3 blocks of text beginning with the line containing *even* and ending with the line containing *odd* to the file BOTH:

```
write_file text='even'..'odd' number=3 file=both
```

- The following copies all lines from the current file to the external copy of file ALPHA:

```
wrif l=all f=alpha
```

- The following copies all of the current working file to the end of file BETA:

```
wrif f=BETA.$EOI
```

- The following copies the working copy of the current file to the external copy. In other words, it makes your changes permanent without closing the current file and leaving the editor:

```
wrif
```

File Manager 19

ENTER_FILE_MANAGER Command

Purpose Starts a File Manager utility session.

Format

ENTER_FILE_MANAGER or
ENTFM
CATALOG=file
PROLOG=file
STATUS=status variable

Parameters

CATALOG or *C* or *FILE* or *F*

Specifies the catalog to be displayed as the main display in File Manager. The default is $WORKING_CATALOG.

PROLOG or *P*

Specifies the file to be used as your prolog for File Manager. This file includes commands you want executed every time you enter File Manager. If omitted, $USER.$FILE_MANAGER.PROLOG is used.

You can establish a different default prolog file. Using the CREATE_DEFAULT_VARIABLE command, define the EUD$ENTFM_PROLOG variable to point to the File Manager prolog you want to use. For example:

```
create_default_variable name=eud$entfm_prolog ..
  default='$user.$file_manager.prolog_filename'
```

In any prolog file you define, include the system commands and File Manager subcommands you want executed.

Remarks The catalog under which you entered File Manager is restored as your working catalog when you quit the session even if you switch to a different working catalog during the session. You can change this through the KEEP_CURRENT_WORKING_CATALOG parameter of the QUIT subcommand.

ACTIVATE_SCREEN
File Manager Subcommand

Purpose Switches from line mode to screen mode.

Format **ACTIVATE_SCREEN** or
ACTS

Parameters None.

Remarks By default, the catalog displayed when you enter ACTIVATE_SCREEN becomes your working catalog. To prevent this, set the UPDATE_WORKING_CATALOG parameter of the SET_SCREEN_OPTIONS subcommand to FALSE.

ALIGN_SCREEN
File Manager Subcommand

Purpose Scrolls the contents of the display.

Format **ALIGN_SCREEN** or
ALIS
FUNCTION=keyword

Parameters **FUNCTION** or **F**

Specifies the scrolling action. This parameter is required. Keyword options are described next.

NOTE

The BKW, FWD, FIRST, LAST, UP, and DOWN keywords perform the same action as the BKW, FWD, FIRST, LAST, UP, and DOWN function keys.

BKW

Moves backward to the previous screen of the display.

FWD

Moves forward to the next screen of the display.

FIRST

Positions the first line of the catalog to the top of the screen.

LAST

Positions the last line of the catalog to the bottom of the screen.

UP

Positions the line at the cursor position to the top of the screen.

DOWN

Positions the line at the cursor position to the bottom of the screen.

BEEP_TERMINAL_BELL
File Manager Subcommand

Purpose Causes the bell on the terminal to beep.

Format **BEEP_TERMINAL_BELL** or
BEETB
STATUS=status variable

Remarks The terminal bell acknowledgement sequence sent to the terminal is defined in the terminal definition. For details, see the NOS/VE Terminal Definition manual.

$CURRENT_CATALOG
File Manager Function

Purpose Returns the catalog path of the currently displayed catalog.

Format **$CURRENT_CATALOG** or
$CC

Parameters None.

Remarks

- The $CURRENT_CATALOG function is only available in the File Manager task. It can be used in SCL procedures but not in other tasks or commands that initiate other tasks. An alternative to $CURRENT_CATALOG is the File Manager variable VCC, which you can use anywhere to specify the current catalog.

In the following example, you cannot use $CURRENT_CATALOG:

```
release_mass_storage;exclude_catalog ..
  catalog=$current_catalog;quit
```

You can, however, use the VCC variable:

```
release_mass_storage;exclude_catalog ..
  catalog=vcc;quit
```

NOTE

VCC is the default variable name; you can change it using the SET_VARIABLES subcommand.

- For further information about functions, see the NOS/VE System Usage manual.

$CURRENT_FILE
File Manager Function

Purpose Returns a file reference naming the file or subcatalog to which the cursor is currently positioned.

Format $CURRENT_FILE or
$CF

Parameters None.

Remarks

- The $CURRENT_FILE function is only available in the File Manager task. It can be used in SCL procedures but not in other tasks or commands that invoke other tasks. An alternative to $CURRENT_FILE is the File Manager variable VCF, which you can use anywhere to specify the current file.

  In the following example, you cannot use $CURRENT_FILE in the following command:

```
create_object_library;add_module ..
  library=$current_file;quit
```

  You can, however, use the VCF variable:

```
create_object_library;add_module ..
  library=vcf;quit
```

NOTE

VCF is the default variable name; you can change it using the SET_VARIABLES subcommand.

- For further information about functions, see the NOS/VE System Usage manual.

DEACTIVATE_SCREEN
File Manager Subcommand

Purpose Switches from screen mode to line mode.

Format **DEACTIVATE_SCREEN** or
DEAS

Parameters None.

Remarks

- The default prompt in line mode is your login family name. You can change this using the SET_LINE_MODE_PROMPT subcommand.
- Return to screen mode by entering ACTIVATE_SCREEN.

EDIT_PATH
File Manager Subcommand

Purpose Displays the current catalog path in a window. While viewing the window, you can change the entry to switch to a different path.

Format EDIT_PATH or
EDIP

Parameters None.

Remarks

- If you specify a nonexisting catalog, then the catalog last displayed is selected, and the cursor is positioned to the first file or catalog that is at the same level or a level above what you specify.
- By default, your working catalog is changed to the catalog you specify. You can prevent this by setting the UPDATE_WORKING_CATALOG parameter of the SET_SCREEN_OPTIONS subcommand to FALSE.

EXTEND_CATALOG_ENTRY_MARK
File Manager Subcommand

Purpose Marks catalog entries from a preceding mark to the current cursor position.

Format **EXTEND_CATALOG_ENTRY_MARK** or **EXTCEM**
STATUS=status variable

Remarks Use the MARK_CATALOG_ENTRY subcommand to mark the initial entry.

Examples To mark rows 3 through 15 of a catalog display: Position the cursor to row 3 and enter MARK_CATALOG_ENTRY; then, move the cursor to row 15 and enter EXTEND_CATALOG_ENTRY_MARK.

$FUNCTION_KEYS
File Manager Function

Purpose Returns a list of records containing the current function key settings.

Format **$FUNCTION_KEYS**

Parameters None.

Remarks

- The list of function key records is returned in the following format:

```
list of record
  number: integer 1..16
  shift: boolean
  command_string: string
  label: string 1..6 = $optional
recend
```

- For further information about functions, see the NOS/VE System Usage manual.

$FUNCTION_SIZE
File Manager Function

Purpose Returns an integer specifying the number of rows on the screen used by the function key menu.

Format **$FUNCTION_SIZE**

Parameters None.

Remarks

- You can use the values returned for $NUMBER_OF_ROWS and $FUNCTION_SIZE to determine the number of screen rows available for displaying windows.
- One row of highlighted boxes in the function key menu uses 2 rows on the screen. Two function key menu rows use 5 rows on the screen.
- For further information about functions, see the NOS/VE System Usage manual.

HELP
File Manager Subcommand

Purpose Displays help information on File Manager.

Format **HELP**
STATUS=status variable

Remarks

- This subcommand is provided for terminals that do not have a predefined Help key.
- In screen mode, entering HELP displays a brief description of the currently displayed screen and associated function keys. The help window indicates whether to enter HELP again for more detailed information.

  In line mode, entering HELP takes you to the online NOS/VE Commands and Functions manual, which includes descriptions of the File Manager commands and functions.

HOME
File Manager Subcommand

Purpose Positions the cursor to the home line, allowing you to enter commands and utility subcommands.

Format

HOME
STATUS=status variable

Remarks

- This subcommand is provided for those terminals that do not have a predefined Home key.
- The home line is inactive when a window is displayed on the screen.

$MAIN_TITLE
File Manager Function

Purpose Returns the File Manager main display title line template.

Format **$MAIN_TITLE**

Parameters None.

Remarks

- The default File Manager title line template is:

```
CATALOG:$C:31$B:23$T Page $S:2of $L:2
```

  See the SET_MAIN_TITLE subcommand for information on interpreting and changing the content of the title line template.

- For further information about functions, see the NOS/VE System Usage manual.

MARK_CATALOG_ENTRY File Manager Subcommand

Purpose Marks or unmarks one or more entries in the currently displayed catalog.

Format

MARK_CATALOG_ENTRY or
MARCE
ENTRY=string or *keyword*
STATUS=status variable

Parameters *ENTRY* or *E*

Specifies the entry or entries to be marked or unmarked. Use a keyword or string to specify the entries to be marked. By default, the entry at which the cursor was last positioned is marked. If the specified entry is already marked, it is unmarked.

Keyword options are:

CURRENT or C

Marks or unmarks the entry at which the cursor was last positioned. This is the default.

ALL

Marks or unmarks all entries in the catalog.

MARK_ALL

Marks all entries in the catalog.

UNMARK_ALL

Unmarks all entries in the catalog.

If you use a string instead of a keyword, you can specify a search string to mark entries that match it. Include any of the following wild card characters in the search string:

?

Matches any single character. For example, a?a matches aza but not aa.

*

Matches zero or more characters. For example, az*az matches azaz and azXXXaz but not aziz.

'c'

Matches the characters within the apostrophes. These can be any characters except the apostrophe. This entry is typically used to specify the asterisk, question mark, right bracket, or left bracket characters. For example, a'['* matches a[z and a[22.

[group]

Matches any single character within the brackets. Specify the characters as a list [aei], a range [a-z], or a combination [a-zA-Z$1-9]. To represent the apostrophe character in the group use ''. Example, [AEI]* matches entries beginning with A, E, or I.

[^]

Identifies a group of single characters to exclude from matching. The group consists of the characters following the caret. Specify the characters as a list [^aei], a range [^a-f], or a combination [^a-fA-F$1-9]. To represent the apostrophe character in the group use ''. Example, [^AEI]* matches all entries that do not begin with A, E or I.

{patterns}

Matches any one of several patterns within the braces. Separate individual patterns from one another using a vertical line (|). For example, {[ac]*|*report} matches entries beginning with the letters a or c and entries ending with the word report.

Remarks Use the EXTEND_CATALOG_ENTRY_MARK subcommand to mark a range of entries from a preceding mark to the current cursor position.

$MARKED_CATALOG_ENTRIES
File Manager Function

Purpose Returns the marked catalog entries.

Format **$MARKED_CATALOG_ENTRIES** or **$MCE**

Parameters None.

Remarks

- The marked catalog entries (catalogs and/or files) are returned as a list of file values.
- For further information about functions, see the NOS/VE System Usage manual.

$MARKED_CATALOG_ENTRIES_SIZE
File Manager Function

Purpose Returns an integer specifying the number of marked catalog entries.

Format **$MARKED_CATALOG_ENTRIES_SIZE**

Parameters None.

Remarks

- Using the $MARKED_CATALOG_ENTRIES_SIZE function is a faster alternative to $SIZE($MARKED_CATALOG_ENTRIES).
- For further information about functions, see the NOS/VE System Usage manual.

$NUMBER_OF_ROWS
File Manager Function

Purpose Returns the number of rows available on the screen.

Format **$NUMBER_OF_ROWS**

Parameters None.

Remarks

- Typically, the value returned by $NUMBER_OF_ROWS is 24 or 30.
- For more information about functions, see the NOS/VE System Usage manual.

POSITION_CURSOR
File Manager Subcommand

Purpose Displays the catalog specified by name or cursor position as the File Manager main display.

Format

POSITION_CURSOR or
POSC or
VIEW or
VIE
CATALOG=file
STATUS=status variable

Parameters *CATALOG* or *C*

Specifies the catalog to be displayed as the main display. If you position the cursor to a catalog before you use POSITION_CURSOR, that catalog is used as the main display. The default is the currently displayed catalog.

Remarks By default, your working catalog is changed to the new catalog. You can prevent this by setting the UPDATE_WORKING_CATALOG parameter of the SET_SCREEN_OPTIONS subcommand to FALSE.

QUIT
File Manager Subcommand

Purpose Ends a File Manager session.

Format

QUIT or
QUI
KEEP_CURRENT_WORKING_CATALOG=boolean
STATUS=status variable

Parameters *KEEP_CURRENT_WORKING_CATALOG* or *KCWC*

Tells File Manager whether you want the working catalog to be the current catalog or or restored to the catalog under which you entered File Manager.

Specify YES to make your current catalog the working catalog. The default is NO, which means your working catalog is restored to the catalog under which you entered File Manager.

REFRESH_SCREEN
File Manager Subcommand

Purpose Clears and repaints the terminal screen.

Format **REFRESH_SCREEN** or
REFS
STATUS=status variable

Remarks REFRESH_SCREEN can be used to clear any extraneous characters from the screen.

$SCREEN_OUTPUT
File Manager Function

Purpose Returns the path of the file reserved for output generated within the utility session.

Output directed to this file by commands executed either on the home line or by function keys is displayed in a window after command completion.

Format **$SCREEN_OUTPUT** or
$SO

Parameters None.

Remarks

- Only one file is used for an entire utility session. Before completing the execution of a command, the file is emptied then new output is added.
- The file is checked each time a command is executed while in screen mode within the utility session. If the file contains command output, it is displayed in a window. Otherwise, the file remains empty.

- The contents of the file is displayed in a window with the same characteristics as the window generated by the SHOW_FILE command.

SET_FUNCTION_KEY
File Manager Subcommand

Purpose Defines one or more function keys.

Format

SET_FUNCTION_KEY or
SETFK
NUMBER=integer
COMMAND_STRING=string
SHIFT=boolean
LABEL=string
KEY_DEFINITIONS=list of record
STATUS=status variable

Parameters

NUMBER or *N*

Specifies the number of the function key to be defined. Enter an integer from 1 to 16. These numbers correspond to the numbers in the function key menu displayed on the screen. Numbers 1 to 8 correspond to the top row in the menu; numbers 9 through 16 correspond to the bottom row.

If you omit this parameter, the subcommand is ignored.

COMMAND_STRING or *CS*

Specifies the statement(s) to be executed when the specified key is pressed. The string can contain any available, executable statement. Separate multiple statements with semicolons.

If you omit this parameter, a single blank character is used.

SHIFT or *S*

Parameter Attributes: BY_NAME

For those terminals that have one key identifier next to each highlighted box in the function key menu, the SHIFT parameter indicates whether the key to be used is shifted. Specify TRUE for the shifted key and FALSE for the nonshifted key.

For those terminals that have two key identifiers next to each highlighted box in the function key menu, the SHIFT parameter indicates which key you use. Specify TRUE to use the key corresponding to the top key identifier. Specify FALSE to use the key corresponding to the bottom key identifier.

If you omit this parameter, FALSE is used.

LABEL or *L*

Parameter Attributes: BY_NAME

Specifies the function key label displayed in the function key menu on the screen. Enter the label as a string of 1 to 6 characters.

Use leading blanks to position the label within the 6-character label field. For example, enter ' Help' to center the Help label in its field.

If you omit this parameter, a string of 6 blank characters is used.

KEY_DEFINITIONS or *KD* or *KEY_DEFINITION*

Parameter Attributes: BY_NAME

Specifies a list of one or more records containing function key definitions. The value you specify must have the following data structure:

```
list of record
  number: integer 1..16
  shift: boolean
  command_string: string
  label: string 1..6 = $optional
recend
```

You can use this parameter rather than specifying individual subcommands for each key definition.

See the NUMBER, SHIFT, COMMAND_STRING, and LABEL parameter descriptions for information on the values you can specify.

Remarks

- You can define function keys one at a time through the NUMBER, SHIFT, COMMAND_STRING, and LABEL parameters or by entering a list of definitions through the KEY_DEFINITIONS parameter. If you have more than one key definition to specify, you will probably save time using the KEY_DEFINITIONS parameter.

- By default in File Manager, 2- and 4-character labels are centered; 3- and 5-character labels are indented 1 character.

Examples

- The following example uses the KEY_DEFINITIONS parameter to specify more than one function key definition:

```
set_function_key key_definitions=( ..
  (1 yes 'help' ' Help') (2 no 'email' 'Email'))
```

- You can use the $FUNCTION_KEYS function to store the current function key settings in a variable for later use. The value returned by this function has the same structure as the value required by the KEY_DEFINITIONS parameter.

  The following example creates an environment variable named KEYS that is accessible throughout a utility. The current function key settings are stored in the variable.

```
VAR
  keys : (UTILITY) list of record
       number : integer 1..16
       shift : boolean
       command_string : string
       label : string
    recend
VAREND

keys = $function_keys
```

  If you want to restore these settings after making changes to the current settings, use the variable KEYS in conjunction with the SET_FUNCTION_KEY subcommand:

```
set_function_key key_definitions = keys
```

SET_LINE_MODE_PROMPT
File Manager Subcommand

Purpose Defines the prompt that precedes the slash when you are using File Manager in line mode.

Format

SET_LINE_MODE_PROMPT or
SETLMP
PROMPT_STRING=string
STATUS=status variable

Parameters

PROMPT_STRING or **PS**

Specifies the content of the prompt. Enter a string of 1 to 27 characters. This parameter is required. The default line mode prompt is the login family name.

Examples The following example sets the prompt string to your login user name:

```
set_line_mode_prompt ..
  prompt_string=$string($job(login_user))
```

SET_MAIN_TITLE
File Manager Subcommand

Purpose Defines the title line of the File Manager main catalog display.

Format

SET_MAIN_TITLE or
SETMT
MAIN_TITLE=string
STATUS=status variable

Parameters

MAIN_TITLE or **MT**

Specifies the content of the title line. Enter a string of 1 to 256 characters. This parameter is required.

The line you specify can include special character strings, called metastrings, to represent variable items such as the date and time.

Metastring descriptions:

The following metastrings are available for use in the title line:

Metastring	Description
$B	For users of CONNECT VIEW. The button functions UpCtlg, PgDn, and PgUp are positioned in this space, right-justified.
$C	Name of the currently displayed catalog.
$D	Current date in the site-defined default format.
$F	Login family name.
$L	Last screen of the File Manager main display.
$P	The family name of the current catalog.
$S	The current screen of the File Manager main display.
$T	Time in H24:MM format.
$U	Login user name.
$V	Value of a specified SCL string variable. Format: `$Vvariable_name:field_width` For details on specifying field width for metastring values, see the field width section following the list of metastrings.
$:	Colon.
$$	Dollar sign.
$0 ⋮ $9	Integers 0 to 9.

Specifying field width in the title line:

You can specify a field width for the values that replace the $B, $C, $D, $F, $L, $P, $S, $T, and $U metastrings. If you do not specify a field width, the values replacing the metastrings begin where you position them in relation

to other elements in the title line. For the metastrings $:, $$, and $0 through $9, the field width is 1. For $V, you must specify a field width.

To specify the field width, enter the metastring, a colon (:), and then an integer specifying the number of characters to reserve for the field. For example, $F:10 reserves 10 characters in the title line for the login family name. The substituted value is always left-justified within the field. You can, however, use spaces to position the metastrings and other characters within the template. For example, the template:

```
My user name is   $u:20  $t   Page $s:2of $l:2
```

produces the title line:

```
My user name is   JIMBO  09:04   Page 1 of 2
```

NOTE

For CONNECT VIEW users:

The labels for the mouse button functions UpCtlg, PgDn, and PgUp require 23 columns. The default File Manager title line template is:

```
CATALOG:$C:31$B:23$T Page $S:2of $L:2
```

The labels UpCtlg, PgDn, and PgUp are positioned in the $B field to be sensitive to mouse clicks.

Remarks

- Use the $MAIN_TITLE function to display the template for the current title line.
- For more information about CONNECT VIEW, see the CONNECT VIEW for the IBM PC manual.

Examples The following example uses deferred evaluation of variables to have a variable updated by SCL. The $V metastring is used to display the task's job mode CPU time in the title line:

```
VAR
  cpu_time:(job defer) string = ..
    $substring($integer_string( ..
      $substring($integer_string( ..
        $cpu_time(tjm)) 1 10)
VAREND

set_main_title ..
  main_title='CPU time is $vcpu_time:10'
```

SET_SCREEN_OPTIONS
File Manager Subcommand

Purpose Specifies the number of function key menu rows displayed. It also specifies whether the working catalog is updated when the display is changed to view a different catalog.

Format

SET_SCREEN_OPTIONS or
SETSO
MENU_ROW=integer
UPDATE_WORKING_CATALOG=boolean
STATUS=status variable

Parameters *MENU_ROW* or *MENU_ROWS* or *MR*

Specifies the number of function key menu rows. The value can be 0, 1, or 2. The default is 1.

UPDATE_WORKING_CATALOG or *UWC*

Specifies whether the working catalog is changed when the display is changed to view a different catalog. TRUE specifies that the working catalog will be updated. FALSE specifies that the working catalog will not be updated. The default is TRUE.

Remarks

- Use the $FUNCTION_SIZE function to list the number of screen rows used by the function key menu rows. There can be 0, 1, or 2 menu rows corresponding to 0, 2, or 5 screen rows.
- Use the value returned by $FUNCTION_SIZE and that returned by $NUMBER_OF_ROWS to determine the number of screen rows available for displaying windows.

SET_VARIABLES
File Manager Subcommand

Purpose Specifies the SCL variables used to store the current catalog and current file.

Format

SET_VARIABLES or
SETV
CURRENT_CATALOG_VARIABLE=data_name
CURRENT_FILE_VARIABLE=data_name
STATUS=status variable

Parameters

CURRENT_CATALOG_VARIABLE or *CCV*

Specifies the variable used to store the current catalog, which is the catalog last displayed. The default variable name is VCC.

CURRENT_FILE_VARIABLE or *CFV*

Specifies the variable used to store the current file, which is the file to which the cursor was last positioned. The default variable name is VCF.

UP_CATALOG
File Manager Subcommand

Purpose Displays the previous catalog in the catalog hierarchy.

Format

UP_CATALOG or
UPC
STATUS=status variable

Remarks

- This subcommand is ignored if you are in your master catalog ($USER) or in the $LOCAL catalog.
- By default, your working catalog is changed to the catalog displayed when you enter UP_CATALOG. You can prevent this by setting the UPDATE_WORKING_CATALOG parameter of the SET_SCREEN_OPTIONS subcommand to FALSE.

ENTER_FTAM_UTILITY Command

Purpose Initiates FTAM/VE. This command establishes a NOS/VE FTAM initiator utility that allows you to perform the file transfer and file management functions. When you initiate FTAM/VE you control execution by specifying parameters. Most of the options you specify on parameters can be changed with subcommands.

Format

ENTER_FTAM_UTILITY or
ENTFU or
FTAM
HOST=application
USER_VALIDATION=application
PASSWORD=application
ACCOUNT=application
BELL_MODE=boolean
TRACE_MODE=boolean
VERBOSE_MODE=boolean
EXPRESSION_EVALUATION=boolean
REALSTORE=keyword
INPUT=file
OUTPUT=file
PROLOG=file
TRACE_FILE=file
STATUS=status variable

Parameters *HOST* or *H*

Specifies the remote host with which the FTAM/VE initiator attempts to establish a connection.

If the HOST parameter is specified, the USER_VALIDATION and the PASSWORD parameters are required. If you omit the USER_VALIDATION or PASSWORD parameter when you are running interactively, the system prompts you for the information.

If this parameter is omitted, you can establish a connection with a remote host later using the CREATE_HOST_CONNECTION subcommand.

USER_VALIDATION or *USER* or *UV* or *U*

Specifies the identity of the calling user.

If the remote host is a NOS/VE host, this parameter must follow the rules of the FTAM/VE responder. For information on the rules used by the responder for processing of this parameter, refer to chapter 1 in the FTAM/VE manual.

If the remote host is a foreign host, this parameter must follow the rules and conventions dictated by the responding system implementation.

To indicate that you wish anonymous access to the responding system, specify an empty string.

This parameter is required when you specify the HOST parameter.

PASSWORD or *PW*

Parameter Attributes: SECURE

Specifies a password, which the FTAM/VE responder uses to validate you. This parameter may be used to authenticate the USER_VALIDATION parameter. It is required when the HOST parameter is specified.

ACCOUNT or *A*

Parameter Attributes: SECURE

Specifies the account to be charged for costs incurred during FTAM operations. This release of the FTAM/VE initiator and responder does not provide accounting.

This parameter may be essential for connecting to some foreign hosts.

BELL_MODE or *BM*

Specifies whether or not an ASCII BEL character is sent to interactive terminals with an input prompt.

This parameter can be used for long file transfers when you want to attend to other matters and be notified when the transfer is complete.

If this parameter is omitted, the default is OFF.

TRACE_MODE or *TM*

Specifies whether or not FTAM protocol data units (PDUs) are written to a trace file.

If this parameter is omitted, the default is OFF.

NOTE

This feature is provided to aid experienced FTAM network system analysts in resolving FTAM/VE problems.

VERBOSE_MODE or *VM*

Specifies whether or not information indicating the amount of data transferred and informative diagnostic messages are written to an output file.

If this parameter is omitted, the default is OFF.

NOTE

Diagnostic messages indicating an error or warning are always supplied.

EXPRESSION_EVALUATION or *EE*

Specifies the mode of evaluation for parameters of type application. Parameters are evaluated as type application if the EXPRESSION_EVALUATION option is OFF and as type string if the EXPRESSION_EVALUATION option is ON. Both types preserve case, which is significant for certain operating systems, and allow most remote host file names, as well as other values to be specified without quotation marks.

If EXPRESSION_EVALUATION is OFF, function expressions within an SCL procedure used as parameters to an FTAM/VE subcommand are not evaluated.

If this parameter is omitted, the default is OFF.

REALSTORE or *R*

Specifies the remote host type. The keywords are:

NOSVE (N)

Identifies the remote host as a NOS/VE system. This option may be useful for file transfers between NOS/VE systems. See subcommands APPEND_FILE, REPLACE_FILE, and GET_FILE for additional

information. Exercise care when using this parameter. Setting the REALSTORE option to NOSVE when the remote system is not a NOS/VE system running FTAM/VE may cause the connection attempt to fail.

UNKNOWN (U)

Indicates that the remote host type is not NOS/VE.

The default is UNKNOWN (U).

INPUT or *I*

Specifies a file from which FTAM/VE subcommands are read.

If FTAM/VE is initiated from within an SCL procedure, FTAM/VE subcommands are not read from the default file $INPUT; they are read from the $COMMAND file.

If this parameter is omitted, the default is $INPUT.

OUTPUT or *O*

Specifies a file to which responses to some FTAM/VE subcommands and all verbose information is written.

If this parameter is omitted, the default is $OUTPUT.

PROLOG or *P*

Specifies a prolog file to be processed when FTAM/VE is invoked. The prolog file is useful for establishing FTAM/VE options different from the default. FTAM/VE options set in the prolog file take precedence over those specified with the ENTER_FTAM_UTILITY command.

If this parameter is omitted, the default is $USER.FTAM_PROLOG.

TRACE_FILE or *TF*

Specifies a file on the local host to which trace information is to be written. The option TRACE_MODE controls whether tracing information is written to this file.

If this parameter is omitted, the default is $USER.FTAM_BINARY_TRACE.

This option is allowed to be specified only with the ENTER_FTAM_UTILITY command. It cannot be changed after the FTAM/VE utility is initiated.

Remarks

- The FTAM/VE user prompt for user input is ftam/.
- The FTAM/VE user prompt for additional input is ftam../.
- FTAM/VE ignores a pause break condition entered during an FTAM/VE subcommand.
- FTAM/VE processes a terminal break condition ending the current operation, if possible.

Examples The following command initiates the FTAM/VE utility.

```
/enter_ftam_utility
ftam/
```

The following command creates a connection upon entering the utility.

```
/enter_ftam_utility host=hosta uv=evol pw=evolpw
ftam/
```

The following command sets options at startup for the session.

```
/enter_ftam_utility tm=on ee=on
ftam/
```

APPEND_FILE FTAM/VE Subcommand

Purpose Appends data from the local host file to the remote host file. If the remote host file exists, the local file is appended to the end of the remote host file. If the remote host file does not exist, it is created by the remote host.

If the VERBOSE_MODE option is currently ON, information about the type of file transfer and the amount of data transferred is written to the local output file.

Format

APPEND_FILE or
APPF or
APPEND
LOCAL_FILE=file
REMOTE_FILE=application
ACCESS_PASSWORDS=list of record

CONCURRENCY_CONTROL=list of record
CREATE_PASSWORD=application
LOCAL_FILE_PASSWORD=name
PERMITTED_ACTIONS=list of keyword
STATUS=status variable

Parameters

LOCAL_FILE or LF

Specifies the name of the local host file.

REMOTE_FILE or *RF*

Specifies the name of the remote host file. The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information, refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

If the REMOTE_FILE parameter is omitted, the APPEND_FILE subcommand uses the file name portion of the value supplied on the LOCAL_FILE parameter as the remote host file name, stripping off the family name, master catalog, and any subcatalogs.

ACCESS_PASSWORDS or *AP*

Parameter Attributes: SECURE

Specifies the passwords associated with the actions you are requesting to be performed. These values may be used by the responding system to verify you are allowed the requested access.

PASSWORD_TYPE

Specifies the access that the password corresponds with. The allowed values

EXTEND (EX)

INSERT (IN)

READ_ATTRIBUTE (RA)

REPLACE (RP)

See File Attributes in chapter 4 of the FTAM/VE Usage manual for information on how these values map into NOS/VE values.

PASSWORD_VALUE

Indicates the password that corresponds to the password type. The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

CONCURRENCY_CONTROL or *CC*

Specifies the locks that are required for the requested access. The locks define the access available to you and to any other user.

REQUESTED_ACCESS

Specifies the access that the lock corresponds with. The access values are:

READ (RE)

INSERT (IN)

REPLACE (RP)

ERASE (ER)

EXTEND (EX)

READ_ATTRIBUTE (RA)

CHANGE_ATTRIBUTE (CA)

DELETE_FILE (DF)

See Concurrency Control in chapter 4 of the FTAM/VE Usage manual for information on how these values map to NOS/VE values.

LOCK

NOT REQUIRED (NR)

Indicates that you will not perform the operation, but others may.

SHARED (S)

Indicates that you may perform the operation and so may others.

EXCLUSIVE (E)

Indicates that you may perform the operation, but others may not.

NO_ACCESS (NA)

Indicates that no one may perform the operation.

CREATE_PASSWORD or *CP*

Parameter Attributes: SECURE

Specifies the password that may be required by the responding system to verify that you have permission to create files in the remote filestore.

The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information, refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

LOCAL_FILE_PASSWORD or *LFP*

Parameter Attributes: SECURE

Specifies the file password. It must match the file password stored with the catalog entry. If the password does not match, an abnormal status is returned.

PERMITTED_ACTIONS or *PERMITTED_ACTION* or *PA*

Specifies the FTAM actions that will be allowed on the file:

READ (RE)

INSERT (IN)

REPLACE (RP)

ERASE (ER)

EXTEND (EX)

READ_ATTRIBUTE (CA)

DELETE_FILE (DF)

TRAVERSAL (T)

REVERSE_TRAVERSAL (RT)

RANDOM ORDER (RO)

See File Attributes in chapter 4 of the FTAM/VE Usage manual for information on how FTAM values map to NOS/VE attributes.

If this parameter is not specified and the file is transferred as an FTAM1 or FTAM3 file, the permitted actions specified by the initiator are: READ, INSERT, REPLACE, EXTEND, ERASE, READ_ATTRIBUTE, CHANGE_ATTRIBUTE, and DELETE_FILE.

If this parameter is not specified and the file is transferred as an FTAM2 or FTAM4 file, the permitted actions specified by the initiator are: READ, INSERT, REPLACE, EXTEND, ERASE, READ_ATTRIBUTE, CHANGE_ATTRIBUTE, DELETE_FILE, and TRAVERSAL.

If the remote file does not exist prior to performing an APPEND_FILE operation, the permitted actions are assigned to the file.

Remarks

- The reverse operation may be accomplished by doing a GET_FILE and specifying the $EOI file position on the local file.
- If the remote system is a NOS/VE host, and the value of the REALSTORE option is NOS/VE, and a remote file is created, the remote file inherits the attributes of the local file.

Examples

The following command appends a local host file to the remote host file.

```
ftam/append_file local_file=$user.text1 ..
ftam../remote_file=a:\dir1\remote_file
```

The following command appends a local host file to the remote host file, specifying passwords for the remote file access.

```
ftam/append_file local_file=$user.text1 ..
ftam../remote_file=a:\dir1\remote_file ap=((extend p-
w1) (replace pw1))
```

CHANGE_FTAM_ATTRIBUTES FTAM/VE Subcommand

Purpose Changes one or more of the FTAM/VE attributes for the specified remote file, provided the change operation is allowed by the remote system.

Format

CHANGE_FTAM_ATTRIBUTES or
CHANGE_FTAM_ATTRIBUTE or
CHAFA
REMOTE_FILE=application
NEW_FILE_NAME=application
STORAGE_ACCOUNT=application
FILE_AVAILABILITY=keyword
FUTURE_FILE_SIZE=integer
ACCESS_PASSWORDS=record
CONCURRENCY_CONTROL=list of record
STATUS=status variable

Parameters

REMOTE_FILE or **RF**

Specifies the name of the remote host file whose attributes are to be changed. The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

NEW_FILE_NAME or *NFN*

Specifies the new name for the remote host file. The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information, refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

STORAGE_ACCOUNT or *SA*

Specifies the storage account attribute value. This parameter may be implemented optionally by FTAM responder implementations. The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information, refer to the

EXPRESSION_EVALUATION parameter description for the ENTER _FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

See File Attributes in chapter 4 of the FTAM/VE Usage manual for information on FTAM/VE responder implementation.

FILE_AVAILABILITY or *FA*

Specifies the availability of the remote file.

IMMEDIATE (I)

Indicates that the file is stored on a fixed media device and no significant delay is encountered when accessing this file.

DEFERRED (D)

Indicates that the file may be stored on a removable media device.

If the value for a file is changed from DEFERRED to IMMEDIATE, this may indicate to a responder to move the file to an immediately available device. However, the actual use of this attribute is implementation- dependent.

See File Attributes in chapter 4 of the FTAM/VE Usage manual for information on FTAM/VE implementation.

FUTURE_FILE_SIZE or *FFS*

Specifies the largest size that the file may grow to (in octets) as a result of file operations. The maximum value allowed for this attribute depends on each responder implementation.

ACCESS_PASSWORDS or *AP*

Parameter Attributes: SECURE

Specifies the passwords associated with the actions you are requesting to be performed. These values may be used by the responding system to verify you are allowed the requested access.

PASSWORD_TYPE

Specifies the access that the password corresponds with. The allowed value is CHANGE_ATTRIBUTE (CA).

PASSWORD_VALUE

Indicates the password that corresponds with the password type. The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information, refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

20

CONCURRENCY_CONTROL or *CC*

Specifies the locks that are required for the requested access. The locks define the access available to you and to any other user.

REQUESTED_ACCESS

Specifies the access that the lock corresponds with. The allowed values are:

READ (RE)

INSERT (IN)

REPLACE (RP)

ERASE (ER)

EXTEND (EX)

READ_ATTRIBUTE (RA)

CHANGE_ATTRIBUTE (CA)

DELETE_FILE (DF)

See Concurrency Control in chapter 4 of the FTAM/VE Usage manual for information on how these values map to NOS/VE values.

LOCK

NOT REQUIRED (NR)

Indicates that you will not perform the operation, but others may.

SHARED (S)

Indicates that you may perform the operation and so may others.

EXCLUSIVE (E)

Indicates that you may perform the operation, but others may not.

NO_ACCESS (NA)

Indicates that no one may perform the operation.

Examples The following command changes two FTAM/VE attributes.

```
ftam/change_ftam_attributes rf=:rust.user1.filea ffs-
=10000 ..
ftam../fa=immediate
```


CHANGE_FTAM_OPTIONS FTAM/VE Subcommand

Purpose Changes one or more of the default FTAM/VE options to the specified value(s).

Format

CHANGE_FTAM_OPTIONS or
CHANGE_FTAM_OPTION or
CHAFO
BELL_MODE=boolean
DEBUG_MODE=boolean
EXPRESSION_EVALUATION=boolean
OUTPUT=file
TRACE_MODE=boolean
VERBOSE_MODE=boolean
STATUS=status variable

Parameters *BELL_MODE* or *BM*

Specifies whether or not an ASCII BEL character is sent to interactive terminals with the prompt. This option is useful for long file transfers when you may want to attend to other matters and be notified when the transfer is complete.

DEBUG_MODE or *DM*

Reserved.

EXPRESSION_EVALUATION or *EE*

Specifies the mode of evaluation for parameters of type application. Parameters are evaluated as type application if the EXPRESSION_EVALUATION option is OFF, and

as type string if the EXPRESSION EVALUATION option is ON. Both types preserve case, which is significant for certain operating systems, and allow most remote host file names as well as other values to be specified without quotation marks.

If EXPRESSION_EVALUATION is OFF, expressions within an SCL application type procedure are NOT evaluated when used as a parameter to an FTAM/VE subcommand.

OUTPUT or *O*

Specifies the file to which information generated by several of the FTAM/VE subcommands is written, as well as the file to which all verbose information is written.

TRACE_MODE or *TM*

Specifies whether or not FTAM PDUs are written to the trace file.

NOTE

This feature is provided to aid experienced FTAM network system analysts in resolving FTAM/VE problems.

VERBOSE_MODE or *VM*

Specifies whether or not information indicating the amount of data transferred and informative diagnostic messages are written to the output file.

NOTE

Diagnostic messages indicating an error or warning are always supplied.

Examples

The following command changes an FTAM/VE option setting.

```
ftam/change_ftam_option trace_mode=on
```

CHANGE_USER_VALIDATION FTAM/VE Subcommand

Purpose Changes the user validation information for the specified remote host. This information is used if you request that a connection be established with that remote host.

Format

CHANGE_USER_VALIDATION or
CHAUV
HOST = application
USER_VALIDATION = application
PASSWORD = application
ACCOUNT = application
REALSTORE = keyword
STATUS = status variable

Parameters

HOST or **H**

Specifies the remote host with which the FTAM/VE initiator attempts to establish a connection.

USER_VALIDATION or **USER** or **UV** or **U**

Specifies the identity of the calling user.

If the remote host is a NOS/VE host, this parameter must follow the rules of the FTAM/VE responder. For information on the rules used by the FTAM/VE responder for processing this parameter refer to chapter 1 of the FTAM/VE manual.

If the remote host is a foreign host, this parameter must follow the rules and conventions dictated by the responding system implementation.

To indicate that you wish anonymous access to the responding system, you may specify an empty string.

PASSWORD or **PW**

Parameter Attributes: SECURE

Specifies a password which the FTAM/VE responder uses to validate you. This parameter may be used to authenticate the USER_VALIDATION parameter.

ACCOUNT or *A*

Parameter Attributes: SECURE

Specifies the account to be charged for costs incurred during FTAM operations. This release of the FTAM/VE responder does not provide accounting.

This parameter may be essential for connecting to a foreign host.

REALSTORE or *R*

Specifies the remote host type.

NOSVE (N)

Indicates that the remote host is a NOS/VE system. This option is useful for file transfers between NOS/VE systems. See subcommands APPEND_FILE, REPLACE_FILE, and GET_FILE for additional information. Exercise care when using this parameter. Setting the REALSTORE option to NOSVE when the remote system is not a NOS/VE system running FTAM/VE may cause the connection to fail.

UNKNOWN (U)

Indicates that the remote host type is not NOS/VE.

Remarks

- The actual value of the ACCOUNT, HOST, USER_VALIDATION, or PASSWORD parameter depends on the value of the EXPRESSION_EVALUATION parameter. For additional information, refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

Examples

The following subcommand changes the user validation for remote host FTAM_VE2.

```
ftam/change_user_validation host=ftam_ve2 ..
ftam../user='login un=bjj fn=nosve' pw=bjjpw
```

CLEAR_TRACE_FILE FTAM/VE Subcommand

Purpose Deletes the current contents of the FTAM/VE trace file.

Format

CLEAR_TRACE_FILE or
CLETF or
CLEAR
STATUS=status variable

Remarks

- A subsequent DISPLAY_TRACE_FILE would not contain any protocol trace information.

Examples The following command deletes the current contents of the FTAM/VE trace file.

```
ftam/clear_trace_file
```

CREATE_HOST_CONNECTION FTAM/VE Subcommand

Purpose Establishes an association with a remote host. Any existing remote host connection must be deleted using the DELETE_HOST_CONNECTION subcommand prior to establishing another host connection.

Format

CREATE_HOST_CONNECTION or
CREHC or
OPEN
HOST=application
USER_VALIDATION=application
PASSWORD=application
ACCOUNT=application
REALSTORE=keyword
STATUS=status variable

Parameters **HOST or H**

Specifies the remote host with which the FTAM/VE initiator attempts to establish a connection.

USER_VALIDATION or *USER* or *UV* or *U*

Parameter Attributes: SECURE

Specifies the identity of the calling user.

If the remote host is a NOS/VE host, this parameter must follow the rules of the FTAM/VE responder. For information on the rules used by the FTAM/VE responder for processing this parameter, refer to chapter 1 in the FTAM/VE manual.

If the remote host is a foreign host, this parameter must follow the rules and conventions dictated by the responding system implementation.

To indicate that you wish anonymous access to the responding system, you may specify an empty string.

PASSWORD or *PW*

Parameter Attributes: SECURE

Specifies a password which the FTAM/VE responder uses to validate you. The remote responder may use this parameter to authenticate the USER_VALIDATION parameter.

ACCOUNT or *A*

Parameter Attributes: SECURE

Specifies the account to be charged for costs incurred during FTAM operations. This release of FTAM/VE does not provide accounting.

REALSTORE or *R*

Specifies the remote host type.

NOSVE (N)

Indicates that the remote host is a NOS/VE system. This option may be useful for file transfers between NOS/VE systems. See subcommands APPEND_FILE, REPLACE_FILE, and GET_FILE for additional information. Exercise care when using this parameter.

Setting the REALSTORE option to NOSVE when the remote system is not a NOS/VE system running FTAM/VE may cause the connection attempt to fail.

UNKNOWN (U)

Indicates that the remote host type is not NOS/VE.

Remarks

- Parameter values specified on CREATE_HOST_CONNECTION take precedence over values established by a previous CHANGE_USER_VALIDATION for the same host.

- If the HOST parameter is specified and there is no user validation defined for the specified host, then the USER_VALIDATION and the PASSWORD parameters are required. Entering an empty string for the USER_VALIDATION parameter specifies an anonymous user. PASSWORD parameters are required. If the USER_VALIDATION parameter or the PASSWORD parameter have been omitted and you are running interactively, you are prompted for the information.

- The actual value of the ACCOUNT, HOST, USER_VALIDATION, or PASSWORD parameter depends on the value of the EXPRESSION_EVALUATION parameter. For additional information, refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

Examples

The following subcommand establishes a connection with the remote host FTAM_HOST_1, with no existing user validation information. The example shows subsystem prompts when the USER_VALIDATION and PASSWORD parameters are omitted.

```
ftam/create_host_connection host=ftam_host_1
User_Validation (Remote_Host: ftam_host_1)
? user1, family2
Password (Remote_Host: ftam_host_1)
? user1pw
```

The following subcommands establish a connection with the remote host FTAM_HOST_1, after setting up the user validation.

```
ftam/change_user_validation host=ftam_host_1 user='u-
ser1, family2' ..
ftam../password=user1pw
ftam/create_host_connection host=ftam_host_1
```


DELETE_FILE FTAM/VE Subcommand

Purpose Deletes a file on the remote host.

When the FTAM/VE responder receives a DELETE_FILE request, it first selects the file, using the highest cycle of the file, unless the user requests a specific cycle number in the DELETE_FILE subcommand. If no cycle is specified, the highest cycle of the file is deleted. Therefore, if more than one cycle of the file exists, you should specify a cycle of $low or a specific cycle number.

NOTE

To delete a file on the local host, use the NOS/VE command $SYSTEM.DELETE_FILE.

Format

DELETE_FILE or
DELF or
DELETE
REMOTE_FILE=application
ACCESS_PASSWORDS=record
CONCURRENCY_CONTROL=record
STATUS=status variable

Parameters

REMOTE_FILE or RF

Specifies the name of the remote host file. The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information, refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

ACCESS_PASSWORDS or *AP*

Parameter Attributes: SECURE

Specifies the passwords associated with the actions you are requesting to be performed. These values may be used by the responding system to verify you are allowed the requested access.

PASSWORD_TYPE

Specifies the access that the password corresponds with. The allowed value is DELETE_FILE (DF).

PASSWORD_VALUE

Indicates the password that corresponds to the password type. The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information, refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

CONCURRENCY_CONTROL or *CC*

Specifies the access locks that are required for the requested access. The locks define the access available to you and to any other user.

REQUESTED_ACCESS

Specifies the access that the lock corresponds with. The allowed values are:

READ (RE)

INSERT (IN)

REPLACE (RP)

ERASE (ER)

EXTEND (EX)

READ_ATTRIBUTE (RA)

CHANGE_ATTRIBUTE (CA)

DELETE_FILE (DF)

See Concurrency Control in chapter 4 of the FTAM/VE Usage manual for information on how these values map to NOS/VE values.

LOCK

NOT REQUIRED (NR)

Indicates that you will not perform the operation, but others may.

SHARED (S)

Indicates that you may perform the operation and so may others.

EXCLUSIVE (E)

Indicates that you may perform the operation, but others may not.

NO_ACCESS (NA)

Indicates that no one may perform the operation.

Examples The following subcommand deletes a file on the remote host.

```
ftam/delete_file remote_file=a:\main\filea
```

DELETE_HOST_CONNECTION FTAM/VE Subcommand

Purpose Terminates an existing connection to a remote host without terminating FTAM/VE.

Format

DELETE_HOST_CONNECTION or
DELHC or
CLOSE
STATUS=status variable

Examples The following subcommand terminates the connection with a remote host.

```
ftam/delete_host_connection
```

DISPLAY_FTAM_ATTRIBUTES FTAM/VE Subcommand

Purpose Writes the current FTAM attributes for the specified remote file to the output file. The amount of information available depends on the responding implementation.

Format

DISPLAY_FTAM_ATTRIBUTES or
DISFA or
DISPLAY_FTAM_ATTRIBUTE or
STATUS
REMOTE_FILE = application
DISPLAY_OPTION = keyword or *list of keyword*
ACCESS_PASSWORDS = list of record
CONCURRENCY_CONTROL = list of record
OUTPUT = file
STATUS = status variable

Parameters

REMOTE_FILE or **RF**

Specifies the name of the remote host file whose attributes are to be displayed. The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Specifies the FTAM remote file attributes that are to be written to the output file.

ALL

Display all FTAM attributes available.

PERMITTED_ACTIONS (PA)

Displays the permitted actions attribute: read, insert, replace, extend, erase, read_attribute, change_attribute, and delete_file.

CONTENTS_TYPE (CT)

Displays the contents type of the remote file; generally a document type name: FTAM1, FTAM2, FTAM3, FTAM4, and so on.

STORAGE_ACCOUNT (SA)

Displays the storage account attribute.

DATE_TIME_OF_CREATION (DTOC)

Displays the date and time the remote file was created.

DATE_TIME_OF_MODIFICATION (DTOM)

Displays the date and time the remote file was last modified.

DATE_TIME_OF_READ (DTOR)

Displays the date and time the remote file was last read.

DATE_TIME_OF_ATTR_MODIFICATION (DTOAM)

Displays the date and time an attribute of the remote file was last modified.

IDENTITY_OF_CREATOR (IOC)

Displays the identity of the user who created the remote file.

IDENTITY OF MODIFIER (IOM)

Displays the identity of the user who last modified the remote file.

IDENTITY_OF_READER (IOR)

Displays the identity of the user who last read the remote file.

IDENTITY_OF_ATTR_MODIFIER (IOAM)

Displays the identity of the user who last modified an attribute of the remote file.

FILE_AVAILABILITY (FA)

Displays the availability of the remote file: DEFERRED or IMMEDIATE.

FILE_SIZE (FS)

Displays the file size of the remote file in octets.

FUTURE_FILE_SIZE (FFS)

Displays the largest size to which the remote file may grow, in octets.

If this parameter is omitted, the default is ALL.

ACCESS_PASSWORDS or *AP*

Parameter Attributes: SECURE

Specifies the passwords associated with the actions you are requesting to be performed. These values may be used by the responding system to verify you are allowed the requested access.

PASSWORD_TYPE

Specifies the access that the password corresponds with. The allowed value is READ_ATTRIBUTE (RA).

PASSWORD_VALUE

Indicates the password that corresponds with the password type. The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information, refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

CONCURRENCY_CONTROL or *CC*

Specifies the access locks that are required for the requested access. The locks define the access available to you and to any other user.

REQUESTED_ACCESS

Specifies the access that the lock corresponds with. The allowed values are:

READ (RE)

INSERT (IN)

REPLACE (RP)

ERASE (ER)

EXTEND (EX)

READ_ATTRIBUTE (RA)

CHANGE_ATTRIBUTE (CA)

DELETE_FILE (DF)

See chapter 4 of the FTAM/VE Usage manual for information on how these values map to NOS/VE values.

LOCK

NOT REQUIRED (NR)

Indicates that you will not perform the operation, but others may.

SHARED (S)

Indicates that you may perform the operation and so may others.

EXCLUSIVE (E)

Indicates that you may perform the operation, but others may not.

NO_ACCESS (NA)

Indicates that no one may perform the operation.

OUTPUT or *O*

Specifies the file to which the attributes of the remote file are written. If this parameter is omitted, the output file specified on the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand is used.

Examples

The following example illustrates the display of FTAM attributes for a remote file.

```
ftam/display_ftam_attributes prolog

Remote_File                        : prolog
Contents_Type                      : FTAM2
  Universal_Class                  : IA5 string
  Maximum_String_Length            : 134
  String_Significance              : not significant
File_Availability                  : immediate
Permitted_Actions                  : (read, insert, replace, extend, eras-
e,
                                     read_attribute, change_attribute, de-
lete)
File_Size                          : 1415
Future_File_Size                   : 100000000
Date_Time_of_Creation              : 1988-02-17 14:53:03:360
Date_Time_of_Modification          : 1989-08-21 15:28:54:400
Date_Time_of_Read                  : 1989-08-22 15:31:53:090
Date_Time_of_Attr_Modification     : no value available
```

```
Identity_of_Creator                : no value available
Identity_of_Modifier               : no value available
Identity_of_Reader                 : no value available
Identity_of_Attribute_Modifier     : no value available
Storage_Account                    : no value available
```

DISPLAY_FTAM_OPTIONS
FTAM/VE Subcommand

Purpose Writes current information about the FTAM/VE connection to the output file. This information includes the option settings that determine the behavior of FTAM/VE and connection status information, if a connection is currently active. The connection information includes the service class, functional units, attribute groups, and document types that have been negotiated during connection establishment.

Format

DISPLAY_FTAM_OPTIONS or
DISFO or
DISPLAY_FTAM_OPTION or
STATUS
DISPLAY_OPTION=keyword or *list of keyword*
OUTPUT=file
STATUS=status variable

Parameters *DISPLAY_OPTION* or *DISPLAY_OPTIONS* or *DO*

Specifies the FTAM/VE option values that are to be written to the output file.

ALL

Displays all FTAM/VE options.

BELL_MODE (BM)

Displays the bell mode, ON or OFF.

EXPRESSION_EVALUATION (EE)

Displays the expression evaluation mode, ON or OFF.

OUTPUT (O)

Displays the output file name.

TRACE_MODE (TM)

Displays the trace mode, ON or OFF.

VERBOSE_MODE (VM)

Displays the verbose mode, ON or OFF.

CONNECTION_STATUS (CS)

Displays information about the present connection, if one exists. This information includes the service class, functional units, attribute groups, and document types that may be used for this association. If no connection with a remote host is currently established, you are informed that no connection exists.

If this parameter is omitted, the default is ALL.

OUTPUT or *O*

Specifies the file to which the FTAM/VE options are written. If the parameter is omitted, the file specified on the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS command is used.

Examples

The following example illustrates the display of FTAM/VE options when no connection exists.

```
ftam/display_ftam_options
 Bell_Mode              : off
 Expression_Evaluation  : off
 Output                 : $LOCAL.$OUTPUT.1
 Trace_Mode             : off
 Verbose_Mode           : off
 Connection_Status      : closed
```

The following example illustrates the display of FTAM/VE options when a connection exists.

```
ftam/display_ftam_options
 Bell_Mode              : off
 Expression_Evaluation  : off
 Output                 : $LOCAL.$OUTPUT.1
 Trace_Mode             : off
 Verbose_Mode           : off
 Connection_Status      : open
 Attribute_Groups       : (kernel, storage)
 Functional_Units       : (read, write, lim. file mg-
mt, enhanced
                        : file mgmt, grouping)
 Service_Class          : transfer & mgmt
```

DISPLAY_TRACE_FILE FTAM/VE Subcommand

Purpose Writes a legible version of the FTAM/VE trace file to the output file. This information includes the FTAM service primitives (requests, indications, responses, confirmations), and the protocol data units (PDUs) containing the FTAM primitives.

NOTE

This feature is provided to aid experienced FTAM network system analysts in resolving FTAM/VE problems.

Format

DISPLAY_TRACE_FILE or
DISTF
DISPLAY_OPTION=keyword or *integer*
OUTPUT=file
STATUS=status variable

Parameters

DISPLAY_OPTION or *DO*

Specifies the number of lines to be written to the output file.

Selecting ALL displays all lines in the trace log. Selecting an integer value displays that number of lines from the end of the trace log.

If this parameter is omitted, the default is ALL.

OUTPUT or *O*

Specifies the file to which the legible trace is written. If omitted, the output file specified on the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand is used.

Examples The following example illustrates the display of an FTAM/VE trace file.

```
ftam/open brown_osi3 ftama ftamax
ftam/display_trace_file
  Cyber_Osiam[9913] Xfer EVN=2100, CHN=1, Sap=15
  Cyber_Osiam[9909] EVN=2100, CHN=1, CTX=0, Layer=70
  [Intcb] F-INITIALIZE-Req, Buffer1 is Titled,titling,ctn,ioi,accnt,fpas
  [Intcb] Sap=0000000f, Chkpt window=00000001, Remote addr=0115, ..=0116
  [Intcb] ..=0205e6, ..=150039840f01bb7b00000000000605080025c4c25d01
  [Intcb] Service Class=70, FU=3700, Att Grp=80, FTAM qos=00, Comm qos=00
  [Intcb] Buf glf=010101010001
  [Buf10S_,13] 4f312d332d393939392d312d37
  [Buf10S_,13] 4f312d332d393939392d312d37
  [Buf10S_,12] 000001000002000003000004
  [Buf10S_,5] 6674616d61
  [Buf10__,7] 016674616d6178
  Cyber_Osiam[9913] Xfer EVN=2100, CHN=1, Sap=16
  Cyber_Osiam[9909] EVN=2100, CHN=1, CTX=0, Layer=50
  [Intcb] S-CONNECT-Request, Buffer1 is Session Id, Buffer2 is User Data
  [Intcb] Sap=00000010, Mode=00000000, Init Serial number=00000001
  [Intcb] Init tk pos=00, Service=80, Functional Units=c002, SSAP Addr=0116
  [Intcb] ..=0205e6, ..=150039840f01bb7b00000000000605080025c4c25d01
  [Intcb] Capability=00
  [Buf20_T,238] 3181eba003800101a281e380020780810115
  [Buf21_T,238] 820115
  [Buf22_T,238] a459301002010106052 8c27b0201
  [Buf23_T,238] 300406025101
  [Buf24_T,238] 3010020102060528c27b0203
  [Buf25_T,238] 300406025101
  [Buf26_T,238] 3010020105060528c27b0204
  [Buf27_T,238] 300406025101
  [Buf28_T,238] 3010020107060528c27b0202
  [Buf29_T,238] 300406025101
  [Buf30_T,238] 300f020109060452010001
  [Buf31_t,238] 300406025101
  [Buf32_T,238] 617c307a020109a075607380020780a1
  Cyber_Osiam[9913] Xfer EVN=1201, CHN=1, Sap=16
  Cyber_Osiam[9909] EVN=1201, CHN=1, CTX=1, Layer=70
  [Intcb] S-CONNECT-Confirmation, Buffer1 is Session Id, Buffer2 is User D-
ata
  [Intcb] Size=00002ff9, Init Serial number=00000000, Init tk pos=00, Serv-
ice=00
  [Intcb] Please tk=00, Functional Units=0002, Capability=0100
  [Buf20_T,183] 3818b4a003800101a281ac80020780830115a52d30078001008102510 1-
300780
  [Buf20_T,183] 0100810251013007800100810251013007800100810251013007800100-
810251
  [Buf20_T,183] 0161743072020109a06d616b80020780a107060528c27b0101a2030201-
00a3056
  [Buf20_T,183] a103020100a40706052bce0f0107be472845020101a040a13e55010045-
010080
  Cyber_Osiam[9913] Xfer EVN=1201, CHN=1, Sap=15
  Cyber_Osiam[9909] EVN=1201, CHN=1, CTX=1, Layer=80
  [Intcb] F-INITIALIZE-Cnf (Positive), Buffer1 is Ctn, Buffer2 is Diag
  [Intcb] Chkpt window=000000001, Action Result=00, Service Class=10, FU=3-
700
  [Buf10__,12] 000001000002000003000004
  int confirm - positive
```

DISPLAY_USER_VALIDATIONS FTAM/VE Subcommand

Purpose Displays the current validations defined for the various hosts.

Format

DISPLAY_USER_VALIDATIONS or
DISPLAY_USER_VALIDATION or
DISUV

DISPLAY_OPTION=keyword or *list of name*
OUTPUT=file
STATUS=status variable

Parameters *DISPLAY_OPTION* or *DISPLAY_OPTIONS* or *DO*

Specifies the host validations to be written to the output file. Specifying ALL causes all host validations you have defined to be displayed.

If this parameter is omitted, the default is ALL.

OUTPUT or *O*

Specifies the file to which the host validations are written. If omitted, the output file specified on the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand is used.

Examples The following command displays the host validations.

```
ftam/display_user_validations

 Host                    : FTAM_HOST1
 User_Validation         : 'u=user1 fn=family1'
 Account                 : A127873
 Realstore               : nosve
 Host                    : FTAM_HOST2
 User_Validation         : 'u=user2 fn=family2'
 Account                 : A127878
 Realstore               : nosve
```

GET_FILE
FTAM/VE Subcommand

Purpose Retrieves a file from the remote host. If the VERBOSE_MODE option is ON, information about the type of file transfer and the amount of data transferred is written to the output file.

Format

GET_FILE or
GET or
GETF or
RECV or
RETR
REMOTE_FILE=application
LOCAL_FILE=file
ACCESS_PASSWORDS=record
CONCURRENCY_CONTROL=list of record
LOCAL_FILE_PASSWORD=name
STATUS=status variable

Parameters

REMOTE_FILE or RF

Specified the name of the remote host file. The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

LOCAL_FILE or *LF*

Specifies the name of the local host file.

If the LOCAL_FILE parameter is omitted, the subcommand uses the value supplied in the REMOTE_FILE parameter as the LOCAL_HOST file name. If it is unable to do this, the GET_FILE subcommand aborts with an error.

ACCESS_PASSWORDS or *AP*

Parameter Attributes: SECURE

Specifies the passwords associated with the actions you are requesting to be performed. These values may be used by the responding system to verify you are allowed the requested access.

PASSWORD_TYPE

Specifies the access that the password corresponds with. The allowed value is: READ (RE).

PASSWORD_VALUE

Indicates the password that corresponds with the password type. The actual parameter value depends on the value of the EXPRESSION_EVALUATION

parameter. For additional information, refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

CONCURRENCY_CONTROL or *CC*

Specifies the access locks that are required for the requested access. The access locks define the access available to you and to any other user.

REQUESTED_ACCESS

Specifies the access that the lock corresponds with. The allowed value is: READ (RE).

See Concurrency Control in chapter 4 of the FTAM/VE Usage manual for information on how these values map to NOS/VE values.

LOCK

NOT REQUIRED (NR)

Indicates that you will not perform the operation, but others may.

SHARED (S)

Indicates that you may perform the operation and so may others.

EXCLUSIVE (E)

Indicates that you may perform the operation, but others may not.

NO_ACCESS (NA)

Indicates that no one may perform the operation.

LOCAL_FILE_PASSWORD or *LFP*

Parameter Attributes: SECURE

Specifies the file password. The password must match the file password stored with the catalog entry. If the password does not match, an abnormal status is returned.

Remarks

- If the remote system is a NOS/VE system and the REALSTORE option is set to NOS/VE, the file attributes of the source file are preserved.

Examples The following subcommand gets a file from the remote host.

```
ftam/get_file remote_file=:family1.user1.filea local-
_file=fileb
```

QUIT
FTAM/VE Subcommand

Purpose Terminates execution of the FTAM/VE utility.

Format

QUIT or
Q or
QUI or
BYE
ABORT=boolean
STATUS=status variable

Parameters *ABORT* or *A*

Specifies whether to abort the current connection before terminating the FTAM/VE utility.

Setting the ABORT parameter to ON, TRUE, or YES causes FTAM/VE to abort the current connection before terminating; setting the ABORT parameter to OFF, FALSE, or NO causes FTAM/VE to close any existing connection before terminating.

If this parameter is omitted, FTAM/VE tries to close any existing connection before terminating.

Examples The following subcommand terminates the execution of FTAM/VE and closes any existing connection in an orderly fashion.

```
ftam/quit
/
```

REPLACE_FILE
FTAM/VE Subcommand

Purpose Sends data from a file on the local host to a file on the remote host. If the remote host file exists, the remote host file is replaced with the local file. If the remote host file does not exist, it is created and the data from the local file is copied into it.

If the VERBOSE_MODE option is ON, information about the type of file transfer and the amount of data transferred is written to the output file.

Format

REPLACE_FILE or
PUT or
REPF or
SEND or
STOR
LOCAL_FILE=file
REMOTE_FILE=application
OVERRIDE=keyword
TRANSPARENT=boolean
ACCESS_PASSWORDS=list of record
CONCURRENCY_CONTROL=list of record
CREATE_PASSWORD=application
LOCAL_FILE_PASSWORD=name
PERMITTED_ACTIONS=list of keyword
STATUS=status variable

Parameters

LOCAL_FILE or **LF**

Specifies the name of the local host file.

REMOTE_FILE or *RF*

Specifies the name of the remote host file. The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information, refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

If the REMOTE_FILE parameter is omitted, the subcommand uses the file name portion of the value supplied on the LOCAL_FILE parameter as the REMOTE_FILE parameter, stripping off the family name, master catalog, and any subcatalogs.

OVERRIDE or *O*

Specifies the action to be taken by the responding system if the remote file exists.

WRITE (W)

Replaces the contents of the remote file with the contents of the local file, retaining the file attributes of the remote file. This is the default value, which should be specified when performing a normal REPLACE_FILE operation.

DELETE (D)

Deletes the remote file and creates a new file with new attributes.

SELECT (S)

Uses the current file as is. Do not use this option for a normal REPLACE_FILE operation.

FAIL (F)

Causes the action to fail if the file already exists. You should not specify this option when performing a normal REPLACE_FILE operation.

TRANSPARENT or *T*

Specifies whether or not the file is to be transmitted transparently. If this parameter is TRUE, FTAM/VE attempts a binary transfer of the file.

See Initiator File Selection in chapter 4 of the FTAM/VE Usage manual for additional information on the rules for the FTAM/VE initiator.

ACCESS_PASSWORDS or *AP*

Parameter Attributes: SECURE

Specifies the passwords associated with the actions you are requesting to be performed. These values may be used by the responding system to verify you are allowed the requested access.

PASSWORD_TYPE

Specifies the access that the password corresponds with. The allowed values are:

EXTEND (EX)

INSERT (IN)

REPLACE (RP)

PASSWORD_VALUE

Indicates the password that corresponds with the password type. The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information, refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

CONCURRENCY_CONTROL or *CC*

Specifies the access locks that are required for the requested access. The access locks define the access available to you and to any other user.

REQUESTED_ACCESS

Specifies the access that the lock corresponds with. The allowed values are:

READ (RE)

INSERT (IN)

REPLACE (RP)

ERASE (ER)

EXTEND (EX)

READ_ATTRIBUTE (RA)

CHANGE_ATTRIBUTE (CA)

DELETE_FILE (DF)

See Concurrency Control in chapter 4 of the FTAM/VE Usage manual for information on how these values map to NOS/VE values.

LOCK

NOT REQUIRED (NR)

Indicates that you will not perform the operation, but others may.

SHARED (S)

Indicates that you may perform the operation and so may others.

EXCLUSIVE (E)

Indicates that you may perform the operation, but others may not.

NO_ACCESS (NA)

Indicates that no one may perform the operation.

CREATE_PASSWORD or *CP*

Parameter Attributes: SECURE

Specifies the password that the responding system might require to verify your permission to create files in the remote filestore.

The actual parameter value depends on the value of the EXPRESSION_EVALUATION parameter. For additional information, refer to the EXPRESSION_EVALUATION parameter description for the ENTER_FTAM_UTILITY command or the CHANGE_FTAM_OPTIONS subcommand.

LOCAL_FILE_PASSWORD or *LFP*

Parameter Attributes: SECURE

Specifies the file password. The password must match the file password stored with the catalog entry. If the password does not match, an abnormal status is returned.

PERMITTED_ACTIONS or *PERMITTED_ACTION* or *PA*

Specifies the FTAM actions to be allowed on the file:

READ (RE)

INSERT (IN)

REPLACE (RP)

ERASE (ER)

EXTEND (EX)

READ_ATTRIBUTE (RA)

DELETE_FILE (DF)

TRAVERSAL (T)

REVERSE_TRAVERSAL (RT)

RANDOM_ORDER (RO)

See File Attributes in chapter 4 of the FTAM/VE Usage manual for information on how VTAM values map to FTAM/VE attributes.

If this parameter is not specified and the file is transferred as an FTAM1 or FTAM3 file, the permitted actions specified by the initiator are: READ, INSERT, REPLACE, EXTEND, ERASE, READ_ATTRIBUTE, CHANGE_ATTRIBUTE, and DELETE_FILE.

If this parameter is not specified and the file is transferred as an FTAM2 or FTAM4 file, the permitted actions specified by the initiator are: READ, INSERT, REPLACE, EXTEND, ERASE, READ_ATTRIBUTE, CHANGE_ATTRIBUTE, DELETE_FILE, and TRAVERSAL.

If the remote file does not exist prior to performing the REPLACE_FILE operation, the permitted actions are assigned to the file.

If the remote file exists prior to performing the REPLACE_FILE operation, the actions are assigned to the file only when the OVERRIDE parameter is set to DELETE.

Remarks

- If the remote system is a NOS/VE system and the value of the REALSTORE option is set to NOS/VE, and a file is created, the file attributes of the source file are preserved.

Examples The following subcommand sends a local host file to a remote host file.

```
ftam/replace_file local_file=.user1.cat1.filea remot-
e_file=filexx
```

Generic Utility Interface 21

ACTIVATE_SCREEN
Generic Utility Interface Subcommand

Purpose Reestablishes screen mode interaction if line mode interaction has been initiated with the DEACTIVATE_SCREEN subcommand.

Format
ACTIVATE_SCREEN or
ACTS

Parameters None.

Remarks For more information, see the NOS/VE System Usage manual.

ALIGN_SCREEN
Generic Utility Interface Subcommand

Purpose Scrolls the contents of the display.

Format
ALIGN_SCREEN or
ALIS
FUNCTION=keyword

Parameters **FUNCTION** or **F**

Specifies the scrolling action. This parameter is required. Keyword options are described next.

NOTE

The BKW, FWD, FIRST, LAST, UP, and DOWN keywords perform the same action as the BKW, FWD, FIRST, LAST, UP, and DOWN function keys.

BKW

Moves backward to the previous screen of the display.

FWD

Moves forward to the next screen of the display.

FIRST

Positions the first line of the display to the top of the screen.

LAST

Positions the last line of the display to the bottom of the screen.

UP

Positions the line at the cursor position to the top of the screen.

DOWN

Positions the line at the cursor position to the bottom of the screen.

21

BEEP_TERMINAL_BELL
Generic Utility Interface Subcommand

Purpose Causes the bell on the terminal to beep.

Format

BEEP_TERMINAL_BELL or
BEETB
STATUS=status variable

Remarks The terminal bell acknowledgement sequence sent to the terminal is defined in the terminal definition. For details, see the NOS/VE Terminal Definition manual.

$CURRENT_LINE
Generic Utility Interface Function

Purpose Returns the line number of the line on which the cursor is positioned.

Format

$CURRENT_LINE or
$CL

Parameters None.

Remarks For further information about functions, see the NOS/VE System Usage manual.

$CURRENT_NAME
Generic Utility Interface Function

Purpose Returns the word on which the cursor is positioned. The value is returned as a name.

Format $CURRENT_NAME or
$CN

Parameters None.

Remarks For further information about functions, see the NOS/VE System Usage manual.

Examples The following example shows a portion of a procedure. The example sets the screen content to a list of file names and sets a function key to display the file on which the cursor is positioned using the SHOW_FILE command.

```
  :
VAR
  path : file = $working_catalog
VAREND

set_screen_content ..
  c=$catalog_contents(path,include_files,names)

set_function_key 4 ..
  cs='show_file f=path//$current_name' 1=' View '
  :
```

$CURRENT_WORD
Generic Utility Interface Function

Purpose Returns the word on which the cursor is positioned. The value is returned as a string.

Format $CURRENT_WORD or
$CW

Parameters None.

Remarks For further information about functions, see the NOS/VE System Usage manual.

Examples The following example shows a portion of a procedure. The example sets the screen content to a list of utility subcommands and sets a function key to execute the subcommand on which the cursor is positioned.

```
  :
set_screen_content ..
  c=$command_list_entry($utility(name))

set_function_key 4 ..
  cs='include_command $current_word' l=' Run '
  :
```

DEACTIVATE_SCREEN
Generic Utility Interface Subcommand

Purpose Switches from screen mode to line mode.

Format **DEACTIVATE_SCREEN** or
DEAS

Parameters None.

Remarks Return to screen mode by entering ACTIVATE_SCREEN.

$FUNCTION_KEYS
Generic Utility Interface Function

Purpose Returns a list of records containing the current function key settings.

Format **$FUNCTION_KEYS**

Parameters None.

Remarks

- The returned value has the following data structure:

```
list of record
  number: integer 1..16
  shift: boolean
  command_string: string
  label: string 1..6
recend
```

- The returned value includes only those function key settings defined using the SET_FUNCTION_KEY subcommand.
- For further information about functions, see the NOS/VE System Usage manual.

Examples

- The following example creates an environment variable named KEYS that is accessible throughout a utility. The current function key settings are stored in the variable.

```
VAR
  keys : (UTILITY) list of record
       number : integer 1..16
       shift : boolean
       command_string : string
       label : string
    recend
VAREND

keys = $function_keys
```

- If you want to restore these settings after making changes to the current settings, use the variable KEYS in conjunction with the SET_FUNCTION_KEY subcommand:

```
set_function_key key_definitions = keys
```

The following example assigns the current setting for the fifth function key to the variable F5:

```
all_keys = $function_keys
  f5 = $select(all_keys,x.number=5)
```


$FUNCTION_SIZE
Generic Utility Interface Function

Purpose Returns an integer specifying the number of rows on the screen used by the function key menu.

Format $FUNCTION_SIZE

Parameters None.

Remarks

- One row of highlighted boxes in the function key menu uses 2 rows on the screen. Two function key menu rows use 5 rows on the screen.
- For further information about functions, see the NOS/VE System Usage manual.

HELP
Generic Utility Interface Subcommand

Purpose Displays help for the current utility or operation.

Format

HELP
STATUS=status variable

Remarks

- This command is provided for those terminals that do not provide a predefined Help key.
- The help text is stored in the help module EUM$SH_GENSU. Procedure writers can provide more help via the HELP_MODULE parameter of the SET_SCREEN_OPTIONS subcommand.

HOME
Generic Utility Interface Subcommand

Purpose Positions the cursor on the home line, allowing you to enter commands and utility subcommands.

Format

HOME
STATUS=status variable

Remarks This command is provided for those terminals that do not provide a predefined Home key.

$LINE_MARKING
Generic Utility Interface Function

Purpose Returns a boolean value or a list of boolean values indicating whether the specified line or lines have been marked.

Format

$LINE_MARKING
(*LINES: keyword* or *list of integer*)

Parameters *LINES*

Specifies the lines to check for marking. You can specify a list of integers or the keyword ALL. If you omit this parameter, the value represented by the $CURRENT_LINE function is used.

Remarks

- If you specify a single line, the function returns a single boolean value. If you specify multiple lines, the function returns a list of boolean values.
- For further information about functions, see the NOS/VE System Usage manual.

$LINE_TEXT
Generic Utility Interface Function

Purpose Returns the text of the specified line or lines as a string or a list of strings.

Format

$LINE_TEXT or
$LT
(*LINES: keyword* or *list of integer*)

Parameters *LINES*

Specifies the lines to return. You can specify a list of integers or the keyword ALL. If you omit this parameter, the value represented by the $CURRENT_LINE function is used.

Remarks

- If you specify a single line, the function returns a single string. If you specify multiple lines, the function returns a list of strings.
- For further information about functions, see the NOS/VE System Usage manual.

Examples The following example assigns all lines in a display to a variable as a list of strings:

```
display_contents = $line_text(all)
```

$MARKED_LINES
Generic Utility Interface Function

Purpose Returns the text of marked lines as a list of strings.

Format **$MARKED_LINES** or
$ML

Parameters None.

Remarks

- If no lines are marked, an empty list is returned.
- For further information about functions, see the NOS/VE System Usage manual.

Examples The following example changes the contents of the screen display to those lines selected from the current display:

```
set_screen_contents c=$marked_lines
```


MARK_SCREEN
Generic Utility Interface Subcommand

Purpose Marks or unmarks lines in the display.

Format **MARK_SCREEN** or
MARS
LINES=keyword or *list of integer* or *list of range of integer*
MARK=boolean
STATUS=status variable

Parameters *LINES* or *LINE* or *L*

Specifies the line or lines to mark or unmark. You can specify an integer, a list of integers, a range of integers or the keyword ALL. If you omit this parameter, the current line is used.

MARK or *M*

Specifies whether the lines are to be marked (TRUE) or unmarked (FALSE). If you omit this parameter, TRUE is used.

Remarks

- You can specify multiple, noncontiguous lines.
- You can specify lines not currently displayed.
- For more information, see the NOS/VE System Usage manual.

Examples The following example defines two function keys: the first function key marks or unmarks the current line based on its current status, and the second function key extends the marking (or unmarking) from the last altered line to the current line:

```
set_function_key 4 ..
  cs='mark_screen m=$not($line_marking)' ..
  l=' Mark '

set_function_key 4 cs='mark_screen '//..
  'l=$most_recently_marked_line..$current_line '//..
  'm=$line_marking($most_recently_marked_line)' ..
     l='ExtMrk' s=true
```


$MOST_RECENTLY_MARKED_LINE
Generic Utility Interface Function

Purpose Returns the line number of the most recently marked line.

Format **$MOST_RECENTLY_MARKED_LINE**

Parameters None.

Remarks

- If no lines are marked, the line number of the current line is returned.
- For further information about functions, see the NOS/VE System Usage manual.

$NUMBER_OF_ROWS
Generic Utility Interface Function

Purpose Returns the number of rows on the screen.

Format **$NUMBER_OF_ROWS**

Parameters None.

Remarks

- Typically, the value returned by $NUMBER_OF_ROWS is 24 or 30.
- For further information about functions, see the NOS/VE System Usage manual.

POSITION_CURSOR
Generic Utility Interface Subcommand

Purpose Positions the cursor on a specified line in the display.

Format

POSITION_CURSOR or
POSC
LINE=keyword or **integer**
STATUS=status variable

Parameters

LINE or **L**

Specifies the line on which to position the cursor. You can specify an integer or one of the following keywords:

FIRST

Positions the cursor on the first line of the display.

LAST

Positions the cursor on the last line of the display.

This parameter is required.

Remarks

- If you specify a line number outside the bounds of the display content, an error occurs.
- For more information, see the NOS/VE System Usage manual.

REFRESH_SCREEN
Generic Utility Interface Subcommand

Purpose Clears and repaints the terminal screen.

Format **REFRESH_SCREEN** or
REFS
STATUS=status variable

Remarks REFRESH_SCREEN can be used to clear any extraneous characters from the screen.

$SCREEN_OUTPUT
Generic Utility Interface Function

Purpose Returns the path of the file reserved for output generated within the utility session.

Output directed to this file by commands executed either on the home line or by function keys is displayed in a window after command completion.

Format **$SCREEN_OUTPUT** or
$SO

Parameters None.

Remarks

- Only one file is used for an entire utility session. Before completing the execution of a command, the file is emptied then new output is added.
- The file is checked each time a command is executed while in screen mode within the utility session. If the file contains command output, it is displayed in a window. Otherwise, the file remains empty.
- The contents of the file is displayed in a window with the same characteristics as the window generated by the SHOW_FILE command.

Examples The following example defines a function key to display command information about a subcommand. The subcommand used is determined by the cursor position. The example assumes the content of the display is a list of utility subcommands.

Output written to the file represented by $SCREEN_OUTPUT is displayed in a window.

```
set_function_key n=4 cs='include_command '//..
  '''display_command_information ''//..
  $line_text//'' o=$screen_output''' l='DisCI'
```

$SELECT_LINES
Generic Utility Interface Function

Purpose Returns, as a list, the line numbers of the lines where text matches the specified text pattern.

Format

$SELECT_LINES
(PATTERN: string)

Parameters

PATTERN

Specifies the text pattern to match. The value you specify must be a string and can contain wild card characters.

Remarks For more information about wild card characters or functions, see the NOS/VE System Usage manual.

Examples

- The following example uses $SELECT_LINES in conjunction with the MARK_SCREEN subcommand to mark lines in the display that match the specified text pattern:

```
mark_screen l=$select_lines('MOD4*')
```

- The following example assigns the result of the $SELECT_LINES function to the variable LINES. The POSITION_CURSOR subcommand then positions the cursor to the first of the selected lines:

```
lines = $select_lines('*11:10.*')
  position_cursor l=$first(lines)
```

- The following example returns the text of the lines that match the specified pattern:

```
text_lines = $line_text($select_lines('MOD[345]*'))
```

SET_FUNCTION_KEY
Generic Utility Interface Subcommand

Purpose Defines one or more function key settings.

Format

SET_FUNCTION_KEY or
SET_FUNCTION_KEYS or
SETFK
NUMBER=integer
COMMAND_STRING=string
SHIFT=boolean
LABEL=string
KEY_DEFINITIONS=list of record
STATUS=status variable

Parameters

NUMBER or *N*

Specifies the number of the function key to be defined. Enter an integer from 1 to 16. These numbers correspond to the numbers in the function key menu displayed on the screen. Numbers 1 to 8 correspond to the top row in the menu; numbers 9 through 16 correspond to the bottom row.

If you omit this parameter, the command is ignored.

COMMAND_STRING or *CS*

Specifies the statement(s) to be executed when the specified key is pressed. The string can contain any available, executable statement. Separate multiple statements with semicolons.

If you omit this parameter, a single blank character is used.

SHIFT or *S*

Parameter Attributes: BY_NAME

For those terminals that have one key identifier next to each highlighted box in the function key menu, the SHIFT parameter indicates whether the key to be used is shifted. Specify TRUE for the shifted key and FALSE for the nonshifted key.

For those terminals that have two key identifiers next to each highlighted box in the function key menu, the SHIFT parameter indicates which key you use. Specify

TRUE to use the key corresponding to the top key identifier. Specify FALSE to use the key corresponding to the bottom key identifier.

If you omit this parameter, FALSE is used.

LABEL or *L*

Parameter Attributes: BY_NAME

Specifies the function key label displayed in the function key menu on the screen. Enter the label as a string of 1 to 6 characters.

Use leading blanks to position the label within the 6-character label field. For example, enter ' Help' to center the Help label in its field.

If you omit this parameter, a string of 6 blank characters is used.

KEY_DEFINITIONS or *KD* or *KEY_DEFINITION*

Parameter Attributes: BY_NAME

Specifies a list of one or more records containing function key definitions. The value you specify must have the following data structure:

```
list of record
  number: integer 1..16
  shift: boolean
  command_string: string
  label: string 1..6
recend
```

You can use this parameter rather than specifying individual subcommands for each key definition.

See the NUMBER, SHIFT, COMMAND_STRING, and LABEL parameter descriptions for information on the values you can specify.

Remarks For more information, see the NOS/VE System Usage manual.

Examples

- You can use the $FUNCTION_KEYS function to store the current function key settings in a variable for later use. The value returned by this function has the same structure as the value required by the KEY_DEFINITIONS parameter.

The following example creates an environment variable named KEYS that is accessible throughout a utility. The current function key settings are stored in the variable.

```
VAR
  keys : (UTILITY) list of record
       number : integer 1..16
       shift : boolean
       command_string : string
       label : string
    recend
VAREND

keys = $function_keys
```

If you want to restore these settings after making changes to the current settings, use the variable KEYS in conjunction with the SET_FUNCTION_KEY subcommand:

```
set_function_key key_definitions = keys
```


SET_SCREEN_CONTENTS
Generic Utility Interface Subcommand

Purpose Sets or changes the contents and title of the display.

Format

SET_SCREEN_CONTENTS or
SET_SCREEN_CONTENT or
SETSC
CONTENT=any
FILE=file
NEW_DISPLAY=boolean
REPRESENTATION=keyword
TITLE=string
STATUS=status variable

Parameters *CONTENT* or *C*

Specifies the value to display. You can specify a value of any type. The presentation of structured values, such as lists or arrays, is determined by the REPRESENTATION parameter.

To specify the contents of a file rather than a value, use the FILE parameter.

If you specify values for both the CONTENT and FILE parameters, the value specified for CONTENT is used. If you omit both the CONTENT and FILE parameters, the current or default display is maintained.

FILE or *F*

Specifies a file to display. The contents of the file are displayed as they appear in the file.

To specify a value rather than the contents of a file, use the CONTENT parameter.

If you specify values for both the FILE and CONTENT parameters, the value specified for CONTENT is used. If you omit both the FILE and CONTENT parameters, the current or default display is maintained.

NEW_DISPLAY or *ND*

Parameter Attributes: BY_NAME

Specifies whether the display is a new display (TRUE) or a reiteration of the existing display (FALSE).

If the display is new, the cursor is positioned at the top of the display. If the display is a refreshed version of the existing display, the current cursor position is maintained.

If you omit this parameter, TRUE is used.

REPRESENTATION or *R*

Parameter Attributes: BY_NAME

Specifies how a structured value specified for the CONTENTS parameter is presented in the display. You can specify one of the following keywords:

COMPRESSED_LABELED_ELEMENTS, CLE

Each element in the value is prefixed by its field name or subscript, if applicable, and is separated from the label by a colon. The value is displayed in a condensed format. Alphabetic characters in elements of type FILE or NAME are displayed as lowercase characters.

ELEMENTS, ELEMENT, E

Each element in the value is displayed on a separate line. Status values can require more than one line.

LABELED_ELEMENTS, LE

Each element in the value is displayed on a separate line and is prefixed by its field name or subscript, if applicable. A colon separates the label from the element. Alphabetic characters in elements of type FILE or NAME are displayed as lowercase characters.

All options display values of the following structured types as a single value on a single line:

COMMAND_REFERENCE
DATE_TIME
ENTRY_POINT_REFERENCE
LINE_IDENTIFIER
TIME_INCREMENT
TIME_ZONE

If you omit this parameter, ELEMENTS is used.

TITLE or *T*

Parameter Attributes: BY_NAME

Specifies the text that appears on the title line of the display. If the string you specify exceeds the available space on the title line, excess characters are truncated.

If you omit this parameter, the title is not altered. If you have not previously set the display title using this parameter, the name of the utility is used.

Remarks For more information, see the NOS/VE System Usage manual.

Examples The following example sets the display to include a list of the files in the current working catalog and an appropriate title:

```
catalog = $working_catalog
set_screen_content ..
  c=$catalog_contents(catalog,include_files) ..
  t='Contents of '//catalog
```

SET_SCREEN_OPTIONS
Generic Utility Interface Subcommand

Purpose Sets or changes certain characteristics of the display.

Format

SET_SCREEN_OPTIONS or
SET_SCREEN_OPTION or
SETSO
HELP_MODULE=name
MENU_ROW=integer
MOUSE_COMMAND=string
REFRESH_RATE=keyword or *time_increment*
UPDATE_COMMAND=string
STATUS=status variable

Parameters

HELP_MODULE or *HM*

Parameter Attributes: BY_NAME

Specifies the name of the help module that contains brief and full help text. Help text is displayed when you enter the HELP subcommand or press the Help function key.

If you omit this parameter, no help is available.

MENU_ROW or *MENU_ROWS* or *MR*

Parameter Attributes: BY_NAME

Specifies the number of function key menu rows. The value can be 0, 1, or 2. The default is 1.

MOUSE_COMMAND or *MC*

Parameter Attributes: BY_NAME

Specifies the statement(s) to be executed when a CONNECT VIEW user presses the left mouse button while positioned on a line in the display.

The string can be between 1 and 127 characters in length and can contain any available, executable statement. Separate multiple statements with semicolons.

If you omit this parameter, the mouse command is not altered. If you have not previously set the mouse command using this parameter, the mouse command is the QUIT subcommand.

REFRESH_RATE or *RR*

Parameter Attributes: BY_NAME

Specifies how often the statement(s) on the UPDATE_COMMAND parameter are called. You can specify a time increment or the keyword STATIC.

If you specify STATIC, the statement(s) specified on the UPDATE_COMMAND parameter are executed each time statement(s) are executed on the home line or from a function key.

If you specify a value of type TIME_INCREMENT, the statement(s) specified on the UPDATE_COMMAND parameter are executed under the following circumstances:

- Each time statement(s) are executed on the home line or from a function key.
- Each time the specified interval elapses following the last statement executed.

If you omit this parameter, the refresh rate is not altered. If you have not previously set the refresh rate using this parameter, the static refresh rate is used.

UPDATE_COMMAND or *UC*

Parameter Attributes: BY_NAME

Specifies the statement(s) to be executed each time the screen is refreshed. The string can be from 1 to 127 characters in length and can contain any available, executable statement. Separate multiple statements with semicolons.

If you omit this parameter, the update command is not altered. If you have not previously set the update command using this parameter, the update command is undefined.

Remarks

- For more information about CONNECT VIEW, see the CONNECT VIEW for the IBM PC manual.
- For more information, see the NOS/VE System Usage manual.

Examples The following example uses the UPDATE_COMMAND parameter to set the content of the display within a SOURCE_CODE_UTILITY session to show the decks on the current library:

```
set_screen_options ..
  uc='set_screen_content $deck_name_list nd=false'
```

Using the SET_SCREEN_CONTENT command in this way ensures that the contents of the display are current. In addition, the contents of the display cannot be overwritten except by reassigning the value of the UPDATE_COMMAND parameter.

EMAIL Command

Purpose

Begins a Mail/VE session in either screen or line mode depending on the interaction style in effect at the time the command is entered.

In screen mode, the screen you specify on the OPERATION parameter of the EMAIL command is displayed.

In line mode, the system executes the operation you specify on the OPERATION parameter. By default, the system displays the following prompt:

```
Mail/
```

Format

EMAIL or
EMA
OPERATION=keyword
OUTPUT=file
ERROR=file
PROLOG=file
STATUS=status variable

Parameters

OPERATION or *OP*

Specifies the operation initiated by Mail/VE after it executes any commands specified on the PROLOG parameter. Specify one of the following keywords:

CHECK or C

Checks mailboxes you own for mail.

Specify CHECK in screen mode to display the Mail List screen, from which you can select a letter to read.

Specify CHECK in line mode and Mail/VE displays a list of all your mailboxes that have mail, followed by the `Mail/` prompt. You can then enter Mail/VE subcommands.

READ or R

Specify READ in screen mode to display the Mail List screen, from which you can select a letter to read.

Specify READ in line mode to display the letter list from your default mailbox along with a one-line description of each letter. If there is no mail to display, an informative message is displayed.

WRITE or W

Specify WRITE in screen mode to display the Write Letter screen, where you can compose and send a letter.

Specify WRITE in line mode to enter the WRITE_LETTER subutility, where you can compose and send a letter. The prompt displayed is `Writ/`.

If the OPERATION parameter is omitted in screen mode, the system displays the Mail List screen. If omitted in line mode, the system displays the `Mail/` prompt.

OUTPUT or *O*

Specifies the file to which the mail session output is written. If omitted, output is displayed on your terminal screen.

ERROR or *E*

Specifies the file to which errors are written. If omitted, errors are written to $ERRORS.

PROLOG or *P*

Specifies the file containing Mail/VE subcommands or SCL commands. When you enter EMAIL to begin a mail session, the commands in this file are executed before processing what you specify on the OPERATION parameter.

If omitted, $NULL is used.

You can override the $NULL default by creating an SCL default variable called MVD$MAIL_PROLOG and setting its value to the name of a prolog file. Specify the name value as a string.

Remarks For more information, see the Mail/VE Version 2 Usage manual.

ACTIVATE_AUTO_FORWARDING CHAM and CREM Subcommand

Purpose Enables automatic forwarding of mail from the mailbox being created or changed.

Format **ACTIVATE_AUTO_FORWARDING** or **ACTAF**

PERSONAL_NAME=string or *record*
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
PRIVATE_DOMAIN=string
ADMINISTRATIVE_DOMAIN=string
COUNTRY=string
UA_IDENTIFIER=string
TERMINAL_IDENTIFIER=string
X121_ADDRESS=string
DOMAIN_DEFINED_ATTRIBUTE=list of record
POSTAL_ADDRESS=list of string
POSTAL_ADDRESS_COUNTRY_NAME=string
POSTAL_ADDRESS_CODE=string
STATUS=status variable

Parameters *PERSONAL_NAME* or *PN*

Specifies the name of the mailbox to which mail will be automatically forwarded. Enter the name as a 1- to 256-character string or a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need more information on specifying a mailbox name.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the mailbox to which mail will be automatically forwarded. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the mailbox to which mail will be automatically forwarded. Enter a 1- to 64-character string.

PRIVATE_DOMAIN or *PD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the private domain associated with the mailbox to which mail will be automatically forwarded. Enter a 1- to 16-character string.

ADMINISTRATIVE_DOMAIN or *AD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the administrative domain associated with the mailbox to which mail will be automatically forwarded. Enter a 1- to 16-character string.

COUNTRY or *C*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country associated with the mailbox to which mail will be automatically forwarded. Enter a 1- to 3-character string.

UA_IDENTIFIER or *UI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the user agent (UA) address identifier relative to an administrative domain to which mail will be automatically forwarded. Enter the identifier as a 1- to 32-character string.

TERMINAL_IDENTIFIER or *TI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the terminal address identifier of the telematic terminal to which mail will be automatically forwarded for reading. Enter the identifier as a 1- to 24-character string.

X121_ADDRESS or *XA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies an X.121 standard network address to which mail will be automatically forwarded. For example, an X.121 address might reference a fax machine or telex location. Enter the address as a 1- to 15-character string of integers 0 to 9.

DOMAIN_DEFINED_ATTRIBUTE or *DDA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the non-X.400 address attributes defined for a mail system to which mail will be automatically forwarded. Enter the attributes as a list of one to four records in the format:

```
record
  name: string 1..8
  value: string 1..128
recend
```

POSTAL_ADDRESS or *PA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the physical postal service address to which mail will be automatically forwarded. Enter the address as a list of two to six strings of 1 to 30 characters each.

POSTAL_ADDRESS_COUNTRY_NAME or *PACN*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country name for the physical postal service address to which mail will be automatically forwarded. Enter the country name as a 1- to 30-character string.

POSTAL_ADDRESS_CODE or *PAC*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the national code designation (for example, the zip code in the U.S.) of the physical postal service address to which mail will be automatically forwarded. Enter the code as a 1- to 128-character string.

Remarks

- Nondelivery notices as a result of the forwarding are returned to the mailbox to which mail is sent originally, not the mailbox of the sender.
- If automatic forwarding is enabled for a mailbox, the system forwards letters from that mailbox to the mailbox you specify. Letters that were previously autoforwarded from another mailbox and receipts and notices are not autoforwarded. To disable autoforwarding, use the DEACTIVATE_AUTO_FORWARDING subcommand.
- In processing the ACTIVATE_AUTO_FORWARDING subcommand, Mail/VE validates your permission to send mail to the specified address if either of the following is true.
  - The specified address is on the same host as your address.
  - Your site supports a domain-wide directory, and the address is on a remote host in the same domain as your Mail/VE mail service. If this is not true, Mail/VE accepts the address and issues a warning message.
- If you activate automatic forwarding to a BITNet or Internet address, use the PERSONAL_NAME parameter to specify the address with a commercial at sign (@):

  ```
  personal_name='jdsmith@arhops.cdc.com'
  ```

  The commercial at sign distinguishes a BITNet or Internet address from a Mail/VE address.
- Mail is delivered to the specified mailbox only if the address to which you are forwarding mail is unique and is defined at the time mail is forwarded.
- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.

- The POSTAL_ADDRESS, POSTAL_ADDRESS_COUNTRY_NAME, and POSTAL_ADDRESS_CODE parameters do not define a mailbox address in the mail directory. They are used to send mail via physical postal delivery.

- For more information, see the Mail/VE Version 2 Usage manual.

Examples In the following example, automatic forwarding is enabled for the mailbox named on the CHANGE_MAILBOX subcommand to the mailbox uniquely identified by the personal name and organization unit address attributes:

```
Mail/change_mailbox
Cham/personal_name='Sticky_Feedback'
Cham/activate_auto_forwarding ..
Cham../personal_name='Thomas Jensen' ..
Cham../organization_unit=( ..
Cham../'Research and Development' ..
Cham../'Adhesive Products' ..
Cham../'Chemicals Division')
```

ACTIVATE_SCREEN MAIL Subcommand

Purpose Activates screen mode of Mail/VE.

Format **ACTIVATE_SCREEN** or
ACTS
STATUS=status variable

Remarks

- Use the ACTIVATE_SCREEN subcommand and its counterpart the DEACTIVATE_SCREEN subcommand to move between screen and line mode in the mail system.

- If you enter ACTIVATE_SCREEN from within a mail subutility, such as WRITE_LETTER, you lose any entries you made within that subutility. Entering ACTIVATE_SCREEN within a subutility is equivalent to quitting the subutility.

- ACTIVATE_SCREEN applies only to the current operation; it does not permanently change your interaction style to screen mode. For example, if you are working in line mode and enter ACTIVATE_SCREEN, Mail/VE switches you to screen mode. When you select another function by pressing `Write`, for example, Mail/VE will return you to line mode. Your mode of interaction with Mail/VE is determined by whether the STYLE parameter on the NOS/VE CHANGE_INTERACTION_STYLE command is set to LINE or SCREEN.

For more information, see the Mail/VE Version 2 Usage manual.

ADD_ADDRESS
CHADL and CREDL Subcommand

Purpose Adds an address to a distribution list.

Format

ADD_ADDRESS or
ADDA
PERSONAL_NAME=string or *record*
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
PRIVATE_DOMAIN=string
ADMINISTRATIVE_DOMAIN=string
COUNTRY=string
UA_IDENTIFIER=string
TERMINAL_IDENTIFIER=string
X121_ADDRESS=string
DOMAIN_DEFINED_ATTRIBUTE=list of record
POSTAL_ADDRESS=list of string
POSTAL_ADDRESS_COUNTRY_NAME=string
POSTAL_ADDRESS_CODE=string
STATUS=status variable

Parameters

PERSONAL_NAME or *DISTRIBUTION_LIST_NAME* or *PN* or *DLN*

Specifies the mailbox or distribution list name of the address you want to add to the distribution list. Enter the name as a string of 1 to 256 characters, or a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need details on specifying a mailbox name or distribution list name.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the unit(s) within the organization of the address to be added to the distribution list. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies an organization name associated with the address to be added to the distribution list. Enter a 1- to 64-character string.

PRIVATE_DOMAIN or *PD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the private domain associated with the address to be added to the distribution list. Enter a 1- to 16-character string.

ADMINISTRATIVE_DOMAIN or *AD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the administrative domain associated with the address to be added to the distribution list. Enter a 1- to 16-character string.

COUNTRY or *C*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country associated with the address to be added to the distribution list. Enter a 1- to 3-character string.

UA_IDENTIFIER or *UI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the user agent (UA) address identifier relative to an administrative domain to be added to the distribution list. Enter the identifier as a 1- to 32-character string.

TERMINAL_IDENTIFIER or *TI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the terminal address identifier of the telematic terminal from which mail will be read to be added to the distribution list. Enter the identifier as a 1- to 24-character string.

X121_ADDRESS or *XA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies an X.121 standard network address to be added to the distribution list. For example, an X.121 address might reference a fax machine or telex location. Enter the address as a 1- to 15-character string of integers 0 to 9.

DOMAIN_DEFINED_ATTRIBUTE or *DDA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the non-X.400 address attributes defined for a mail system to be added to the distribution list. Enter the attributes as a list of one to four records in the format:

```
record
  name: string 1..8
  value: string 1..128
recend
```

POSTAL_ADDRESS or *PA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the physical postal service address to be added to the distribution list. Enter the address as a list of two to six strings of 1 to 30 characters each.

POSTAL_ADDRESS_COUNTRY_NAME or *PACN*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country name for the physical postal service address to be added to the distribution list. Enter the country name as a 1- to 30-character string.

POSTAL_ADDRESS_CODE or *PAC*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the national code designation (for example, the zip code in the U.S.) of the physical postal service address to be added to the distribution list. Enter the code as a 1- to 128-character string.

Remarks

- Addresses are added to a distribution list exactly as they are entered. No verification or validation of any form occurs.
- If you add a BITNet or Internet address to a distribution list, use the PERSONAL_NAME parameter with a commercial at sign (@):

```
personal_name='jdsmith@arhops.cdc.com'
```

  The commercial at sign distinguishes a BITNet or Internet address from a Mail/VE address.
- You cannot use the wildcard character * in addresses you specify.
- The POSTAL_ADDRESS, POSTAL_ADDRESS_COUNTRY_NAME, and POSTAL_ADDRESS_CODE parameters do not define a mailbox address in the mail directory. They are used to send mail via physical postal delivery.
- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example adds two addresses to an existing distribution list:

```
Mail/change_distribution_list..
Mail../'Technical_Review_Board'
Chadl/add_address 'Ellen Gray'
Chadl/add_address 'Malcolm Boswell'
Chadl/end_change_distribution_list
```

ADD_BLIND_COURTESY_COPY FORL, WRIL, and WRIR Subcommand

Purpose Adds an address to the blind courtesy copy address list.

Format

ADD_BLIND_COURTESY_COPY or
ADDBCC or
BCC
PERSONAL_NAME=string or *record*
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
PRIVATE_DOMAIN=string
ADMINISTRATIVE_DOMAIN=string
COUNTRY=string
UA_IDENTIFIER=string
TERMINAL_IDENTIFIER=string
X121_ADDRESS=string
DOMAIN_DEFINED_ATTRIBUTE=list of record
POSTAL_ADDRESS=list of string
POSTAL_ADDRESS_COUNTRY_NAME=string
POSTAL_ADDRESS_CODE=string
RECIPIENT_OPTIONS=keyword or *list of keyword*
STATUS=status variable

Parameters *PERSONAL_NAME* or *DISTRIBUTION_LIST_NAME* or *PN* or *DLN*

Specifies the name associated with the address to be added to the blind courtesy copy list. Enter the name as a string of 1 to 256 characters or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need help on specifying names.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the address to be added to the address list. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the address to be added to the address list. Enter a 1- to 64-character string.

PRIVATE_DOMAIN or *PD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the private domain associated with the address to be added to the blind courtesy copy address list. Enter a 1- to 16-character string.

ADMINISTRATIVE_DOMAIN or *AD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the administrative domain associated with the address to be added to the blind courtesy copy address list. Enter a 1- to 16-character string.

COUNTRY or *C*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country associated with the address to be added to the blind courtesy copy address list. Enter a 1- to 3-character string.

UA_IDENTIFIER or *UI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the user agent (UA) address identifier relative to an administrative domain to be added to the address list. Enter the identifier as a 1- to 32-character string.

TERMINAL_IDENTIFIER or *TI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the terminal address identifier of the telematic terminal from which mail is read to be added to the address list. Enter the identifier as a 1- to 24-character string.

X121_ADDRESS or *XA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies an X.121 standard network address to be added to the address list. For example, an X.121 address might reference a fax machine or telex location. Enter the address as a 1- to 15-character string of integers 0 to 9.

DOMAIN_DEFINED_ATTRIBUTE or *DDA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the non-X.400 address attributes defined for a mail system to be added to the address list. Enter the attributes as a list of one to four records in the format:

```
record
  name: string 1..8
  value: string 1..128
recend
```

POSTAL_ADDRESS or *PA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the physical postal service address to be added. Enter the address as a list of two to six strings of 1 to 30 characters each.

POSTAL_ADDRESS_COUNTRY_NAME or *PACN*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country name for the physical postal service address to be added. Enter the country name as a 1- to 30-character string.

POSTAL_ADDRESS_CODE or *PAC*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the national code designation (for example, the zip code in the U.S.) of the physical postal service address to be added. Enter the code as a 1- to 128-character string.

RECIPIENT_OPTIONS or *RECIPIENT_OPTION* or *RO*

Parameter Attributes: BY_NAME

Specifies the message attributes for the address you are adding to the blind courtesy copy list. This parameter does not apply to distribution lists that cannot be expanded.

Specify ALL, NONE, or one or more of the following keywords. ALL means the letter is certified for receipt or nonreceipt notification and indicates in the header information that a reply is requested. NONE means no letter attributes are set for the specified address.

CERTIFY_RECEIPT or CR

The originator receives a message when the recipient has seen the letter. The letter is considered seen if the recipient displays the letter or copies the letter to a file.

CERTIFY_NON_RECEIPT or CNR

The originator receives a message if the letter is deleted without having been seen or is autoforwarded. The mail system generates the nonreceipt notice if the letter is unseen at the time of a deletion or expiration.

REPLY_REQUESTED or RR

The letter indicates in the header information that the originator is expecting a reply. The system does not require the recipient to reply.

If omitted, NONE is used.

Remarks

- Recipients of blind courtesy copies are not displayed to the courtesy copy and primary recipients of the letter.
- When you enter the ADD_BLIND_COURTESY_COPY subcommand, the mail system performs the following address verification:
  - If you enter an ADD_BLIND_COURTESY_COPY subcommand without any address parameters, the subcommand terminates with an error message.
  - If you enter an address that is in the directory, the system accepts the address.
  - If you enter an ambiguous address, the system displays a message to let you know and does not add the address to the recipient list.
  - If you enter an address that is not in the directory and if the organization attributes you give for that address specify the local host, the system displays an informative message.
  - If you enter an address that is not in the directory and if the organization attributes you give for that address specify a remote host, the system accepts the address. However, the address is not verified until the remote host attempts to deliver it.
  - If you enter an address that is not in the directory and if the organization attributes you give for that address have not been specified by the Mail/VE administrator, the subcommand terminates with a warning message.
- There are three methods of adding a BITNet or Internet address to a blind courtesy copy recipient list. How your Mail/VE administrator configured your mail network determines which methods work. Select the one your Mail/VE administrator recommends.

- Enter the PERSONAL_NAME parameter with a commercial at sign:

```
personal_name='jdsmith@arhops.cdc.com'
```

  You can use this method any time. Because the commercial at sign indicates a BITNet or Internet address, the system automatically sends it to the gateway for routing.

- Enter the address the same way you do a Mail/VE address. For example, enter `jdsmith@arhops.cdc.com` as follows:

```
personal_name='jdsmith'
organization_unit='arhops'
organization_name='cdc'
```

  This method works only if your Mail/VE administrator has configured the Mail/VE gateway to map the organization name and/or organization units to a BITNet or Internet address of `arhops.cdc.com`.

- Enter the DOMAIN_DEFINED_ATTRIBUTES parameter to specify the address. The characters (a) replace the commercial at sign in this format.

```
domain_defined_attributes=( ..
('RFC-822','jdsmith(a)arhops.cdc.com'))
```

  If you use this method, you must also enter the organization name and/or organization units, private domain, administrative domain, and country of the Mail/VE gateway.

- If you enter a BITNet or Internet address, the CERTIFY_RECEIPT and CERTIFY_NON_RECEIPT options on the RECIPIENT_OPTIONS parameter do not apply.

- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.

- You cannot use the wildcard * character in the addresses you specify.

- The POSTAL_ADDRESS, POSTAL_ADDRESS_COUNTRY_NAME, and POSTAL_ADDRESS_CODE parameters do not define a mailbox address in the mail directory. They are used to send mail via physical postal delivery.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples In the following example a blind courtesy copy list is added to a forwarded letter:

```
Mail/forward_letter current
Forl/add_letter_part $local.note
Forl/set_subject 'Jury Analysis'
Forl/addbcc 'Harold Spooner'
Forl/addbcc 'Joseph Cheswick'
Forl/addbcc 'Karen Garnet'
Forl/end_forward_letter
```

ADD_COURTESY_COPY FORL, WRIL, and WRIR Subcommand

Purpose Adds addresses to the courtesy copy address list.

Format

ADD_COURTESY_COPY or
ADDCC or
CC
PERSONAL_NAME=string or *record*
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
PRIVATE_DOMAIN=string
ADMINISTRATIVE_DOMAIN=string
COUNTRY=string
UA_IDENTIFIER=string
TERMINAL_IDENTIFIER=string
X121_ADDRESS=string
DOMAIN_DEFINED_ATTRIBUTE=list of record
POSTAL_ADDRESS=list of string
POSTAL_ADDRESS_COUNTRY_NAME=string
POSTAL_ADDRESS_CODE=string
RECIPIENT_OPTIONS=keyword or *list of keyword*
STATUS=status variable

Parameters *PERSONAL_NAME* or *DISTRIBUTION_LIST_NAME* or *PN* or *DLN*

Specifies the mailbox or distribution list name to be added to the courtesy copy list. Enter the name as a string of 1 to 256 characters or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need more information on specifying names.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the address to be added to the courtesy copy list. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the address to be added to the courtesy copy list. Enter a 1- to 64-character string.

PRIVATE_DOMAIN or *PD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the private domain associated with the address to be added to the courtesy copy address list. Enter a 1- to 16-character string.

ADMINISTRATIVE_DOMAIN or *AD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the administrative domain associated with the address to be added to the courtesy copy address list. Enter a 1- to 16-character string.

COUNTRY or *C*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country associated with the address to be added to the courtesy copy address list. Enter a 1- to 3-character string.

UA_IDENTIFIER or *UI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the user agent (UA) address identifier relative to an administrative domain to be added to the address list. Enter the identifier as a 1- to 32-character string.

TERMINAL_IDENTIFIER or *TI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the terminal address identifier of the telematic terminal from which mail is read to be added to the address list. Enter the identifier as a 1- to 24-character string.

X121_ADDRESS or *XA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies an X.121 standard network address to be added to the list. For example, an X.121 address might reference a fax machine or telex location. Enter the address as a 1- to 15-character string of integers 0 to 9.

DOMAIN_DEFINED_ATTRIBUTE or *DDA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the non-X.400 address attributes defined for a mail system to be added to the address list. Enter the attributes as a list of one to four records in the format:

```
record
  name: string 1..8
  value: string 1..128
recend
```

POSTAL_ADDRESS or *PA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the physical postal service address to be added. Enter the address as a list of two to six strings of 1 to 30 characters each.

POSTAL_ADDRESS_COUNTRY_NAME or *PACN*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country name for the physical postal service address to be added. Enter the country name as a 1- to 30-character string.

POSTAL_ADDRESS_CODE or *PAC*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the national code designation (for example, the zip code in the U.S.) of the physical postal service address to be added. Enter the code as a 1- to 128-character string.

RECIPIENT_OPTIONS or *RECIPIENT_OPTION* or *RO*

Parameter Attributes: BY_NAME

Specifies the message attributes for the address you are adding to the courtesy copy list. This parameter does not apply to distribution lists that cannot be expanded.

Specify ALL, NONE, or one or more of the following keywords. ALL means the letter is certified for receipt or nonreceipt notification and indicates in the header information that a reply is requested. NONE means no letter attributes are set for the specified address.

CERTIFY_RECEIPT or CR

The originator receives a message when the recipient has seen the letter. The letter is considered seen if the recipient displays or copies the letter to a file.

CERTIFY_NON_RECEIPT or CNR

The originator receives a message if the letter is deleted without having been seen or is autoforwarded. The mail system generates a nonreceipt notice if the letter is unseen at the time of a deletion or the letter is autoforwarded.

REPLY_REQUESTED or RR

The letter indicates in the header information that the originator is expecting a reply. The system does not require the recipient to reply.

If omitted, NONE is used.

Remarks

- When you enter the ADD_COURTESY_COPY subcommand, the mail system performs the following address verification:
  - If you enter an ADD_COURTESY_COPY subcommand without any address parameters, the subcommand terminates with an error message.
  - If you enter an address that is in the directory, the system accepts the address.
  - If you enter an ambiguous address, the system displays a message to let you know and does not add the address to the recipient list.
  - If you enter an address that is not in the directory and if the organization attributes you give for that address specify the local host, the system displays an informative message.
  - If you enter an address that is not in the directory and if the organization attributes you give for that address specify a remote host, the system accepts the address. However, the address is not verified until the remote host attempts to deliver it.
  - If you enter an address that is not in the directory and if the organization attributes you give for that address have not been specified by the Mail/VE administrator, the subcommand terminates with a warning message.
- There are three methods of adding a BITNet or Internet address to a courtesy copy recipient list. How your Mail/VE administrator configured your mail network determines which methods work. Select the one your Mail/VE administrator recommends.
  - Use the PERSONAL_NAME parameter with a commercial at sign (@):

    ```
    personal_name='jdsmith@arhops.cdc.com'
    ```

    You can use this method any time. Because the commercial at sign indicates a BITNet or Internet address, the system automatically sends it to the gateway for routing.

- Enter the address the same way you do a Mail/VE address. For example, enter jdsmith@arhops.cdc.com as follows:

```
personal_name='jdsmith'
organization_unit='arhops'
organization_name='cdc'
```

  This method works only if your Mail/VE administrator has configured the Mail/VE gateway to map the organization name and/or organization units to a BITNet or Internet address of arhops.cdc.com.

- Enter the DOMAIN_DEFINED_ATTRIBUTES parameter to specify the address. The characters (a) replace the commercial at sign in this format.

```
domain_defined_attributes=( ..
('RFC-822','jdsmith(a)arhops.cdc.com'))
```

  If you use this method, you must also enter the organization name and/or organization units, private domain, administrative domain, and country of the Mail/VE gateway.

- If you enter a BITNet or Internet address, the CERTIFY_RECEIPT and CERTIFY_NON_RECEIPT options on the RECIPIENT_OPTIONS parameter do not apply.

- You cannot use the wildcard character * in mailbox or distribution list addresses you specify.

- The POSTAL_ADDRESS, POSTAL_ADDRESS_COUNTRY_NAME, and POSTAL_ADDRESS_CODE parameters do not define a mailbox address in the mail directory. They are used to send mail via physical postal delivery that regulates your mail system.

- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.

- For more information, see the Mail/VE Version 2 Usage manual.

Examples In the following example, a courtesy copy list is added to a reply:

```
Mail/write_reply
Wrir/add_letter_part $local.note
Wrir/cc 'Charlie Cantnor'
Wrir/cc 'Beryl Nelson'
Wrir/end_write_reply
```

ADD_LETTER_PARTS FORL, WRIL, and WRIR Subcommand

Purpose Adds a file to the list of parts that make up the body of a letter.

Format

ADD_LETTER_PARTS or
ADD_LETTER_PART or
ADDLP
LETTER_PARTS=list of file
LETTER_PART_TYPE=keyword
PLACEMENT=keyword
DESTINATION=keyword
STATUS=status variable

Parameters

LETTER_PARTS or **LETTER_PART** or **LP**

Specifies a list of file names to add to the list of letter parts.

This parameter is required.

LETTER_PART_TYPE or *LPT*

Specifies the kind of data contained in the letter parts. Keyword options:

NONTEXT or UNKNOWN

The letter parts contain non-ASCII characters. These parts are delivered as letter part type UNDEFINED.

TEXT

The letter files contain ASCII text. The NOS/VE FILE_CONTENT attribute of the file must be either LIST or LEGIBLE_DATA.

If LETTER_PART_TYPE is omitted, TEXT is used.

PLACEMENT or *P*

Parameter Attributes: BY_NAME

Specifies whether the letter parts are added before or after the letter part specified on the DESTINATION parameter. Options:

BEFORE or B

Places the letter parts before the destination.

AFTER or A

Places the letter parts after the destination.

If DESTINATION is omitted, AFTER is used.

DESTINATION or *D*

Parameter Attributes: BY_NAME

Specifies a letter part in the letter part list before or after which the added parts are placed. Options:

FIRST or F

Selects the first letter in the letter list.

LAST or L

Selects the last letter in the letter list.

If omitted, LAST is used.

Remarks

- If a file specified does not exist, Mail/VE issues an error message and does not send the letter.
- The Mail/VE gateway only accepts text files. If the Mail/VE gateway receives a nontext letter part for transmission to BITNet or Internet, the sender receives a notice. The letter is not delivered. For more information on letter parts, see chapter 4.
- If a letter part is added from a prolog to the FORWARD_LETTER subutility, then the added part must have the PLACEMENT parameter set to BEFORE and the DESTINATION parameter set to FIRST.

- Use the NOS/VE CHANGE_FILE_ATTRIBUTES command to change the FILE_CONTENT attribute as needed.

- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example specifies and positions a list of files to make up the body of a letter:

```
Mail/write_letter
Wril/to 'Randall McDonald'
Wril/addlp (.ltr.process.timecard_entry ..
Wril../.ltr.process.status_reports ..
Wril../.ltr.process.load_charts ..
Wril../.ltr.process.monthly_review)
Wril/collect_text $local.note
Colt/These process descriptions are recommended
Colt/reading for new members of the project.
Colt/**
Wril/addlp $local.note p=before d=first
Wril/add_subject 'Reporting Procedures'
Wril/end_write_letter
```

ADD_PERMISSION
CHADL, CHAM, CREDL, and CREM Subcommand

22

Purpose Adds an address to the list of permitted addresses for group mailboxes or group distribution lists.

Format

ADD_PERMISSION or
ADDP
PERSONAL_NAME=string or *record*
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
PRIVATE_DOMAIN=string
ADMINISTRATIVE_DOMAIN=string
COUNTRY=string
PERMISSIONS=keyword or *list of keyword*
STATUS=status variable

Parameters *PERSONAL_NAME* or *PN*

Specifies the name of the mailbox or distribution list to be added to the permitted list. Enter the name as a string of 1 to 256 characters or a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need further information on specifying a mailbox or distribution list name.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the address to be added to the permit list. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the address to be added to the permitted list. Enter a 1- to 64-character string.

PRIVATE_DOMAIN or *PD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the private domain associated with the address to be added to the permitted list. Enter a 1- to 16-character string.

ADMINISTRATIVE_DOMAIN or *AD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the administrative domain associated with the address to be added to the permitted list. Enter a 1- to 16-character string.

COUNTRY or *C*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country associated with the address to be added to the permitted list. Enter a 1- to 3-character string.

PERMISSIONS or *PERMISSION* or *P*

Parameter Attributes: BY_NAME

Specifies the list of permissions to be granted to the specified mailbox. Choose ALL or one or more of the following options. For mailboxes, ALL gives READ and WRITE permissions. For distribution lists, ALL gives READ and USE permissions.

READ or R

For mailboxes on the local host, letters can be read. For distribution lists, the addresses in the list can be displayed.

WRITE or W

For mailboxes only. Letters can be sent to the mailbox.

USE or U

For distribution lists only. The list can be included in a recipient list. However, without READ access, the addresses in the list cannot be displayed.

If PERMISSIONS is omitted, ALL is used.

Remarks

- Specify primary mailbox names on the PERSONAL_NAME parameter, not aliases. When another user attempts to access the mailbox or distribution list, the system will use their primary personal name to validate their access permission.
- Addresses are added to the permitted mailbox list exactly as they are entered. No verification or validation occurs.
- The wildcard character * is allowed in the address attributes. By using the * in your address, you can permit more than one mailbox with a single permission description.

- You can permit BITNet and Internet addresses to access group mailboxes or distribution lists. To do this, grant permission to the X.400 address of the Mail/VE gateway or to one of the address attributes, such as country.

  In the following example, the ADD_PERMISSION subcommand grants permission to all addresses within the country represented by country code DE to use a specified group mailbox.

  ```
  add_permission country='de' permission=all
  ```

  In other words, it adds all addresses in West Germany to the permitted list for the specified group mailbox. Since the converted address in this example contains DE as the COUNTRY parameter, the address is included in the permitted list.

- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.

- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example permits anyone in the organization unit named ENGINEERING to read or use the specified distribution list:

```
Mail/change_distribution_list ..
Mail../ 'technical_review_board'
Chadl/add_permission ou='engineering'
Chadl/end_change_distribution_list
```


ADD_TO
FORL, WRIL, and WRIR Subcommand

Purpose Adds an address to the primary (To) address list.

Format

ADD_TO or
ADDT or
TO
PERSONAL_NAME=string or *record*
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string

PRIVATE_DOMAIN=string
ADMINISTRATIVE_DOMAIN=string
COUNTRY=string
UA_IDENTIFIER=string
TERMINAL_IDENTIFIER=string
X121_ADDRESS=string
DOMAIN_DEFINED_ATTRIBUTE=list of record
POSTAL_ADDRESS=list of string
POSTAL_ADDRESS_COUNTRY_NAME=string
POSTAL_ADDRESS_CODE=string
RECIPIENT_OPTIONS=keyword or *list of keyword*
STATUS=status variable

Parameters

PERSONAL_NAME or *DISTRIBUTION_LIST_NAME* or *PN* or *DLN*

Specifies the mailbox or distribution list name you are adding to the To address list. Enter the name as a string of 1 to 256 characters or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need more information on specifying names.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the address to be added to the To address list. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the address to be added to the To address list. Enter a 1- to 64-character string.

PRIVATE _DOMAIN or *PD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the private domain associated with the address to be added to the To address list. Enter a 1- to 16-character string.

ADMINISTRATIVE _DOMAIN or *AD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the administrative domain associated with the address to be added to the To address list. Enter a 1- to 16-character string.

COUNTRY or *C*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country associated with the address to be added to the To address list. Enter a 1- to 3-character string.

UA _IDENTIFIER or *UI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the user agent (UA) address identifier relative to an administrative domain to be added to the address list. Enter the identifier as a 1- to 32-character string.

TERMINAL _IDENTIFIER or *TI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the terminal address identifier of the telematic terminal from which mail is read to be added to the address list. Enter the identifier as a 1- to 24-character string.

X121 _ADDRESS or *XA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies an X.121 standard network address to be added to the address list. For example, an X.121 address might reference a fax machine or telex location. Enter the address as a 1- to 15-character string of integers 0 to 9.

DOMAIN_DEFINED_ATTRIBUTE or *DDA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the non-X.400 address attributes defined for a mail system to be added to the address list. Enter the attributes as a list of one to four records in the format:

```
record
  name: string 1..8
  value: string 1..128
recend
```

POSTAL_ADDRESS or *PA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the physical postal service address to be added. Enter the address as a list of two to six strings of 1 to 30 characters each.

POSTAL_ADDRESS_COUNTRY_NAME or *PACN*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country name for the physical postal service address to be added. Enter the country name as a 1- to 30-character string.

POSTAL_ADDRESS_CODE or *PAC*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the national code designation (for example, the zip code in the U.S.) of the physical postal service address to be added. Enter the code as a 1- to 128-character string.

RECIPIENT_OPTIONS or *RECIPIENT_OPTION* or *RO*

Parameter Attributes: BY_NAME

Specifies the message attributes for the address you are adding to the To address list. This parameter does not apply to distribution lists that cannot be expanded.

Specify ALL, NONE, or one or more of the following keywords. ALL means the letter is certified for receipt or nonreceipt notification and indicates in the header information that a reply is requested. NONE means no letter attributes are set for the specified address.

CERTIFY_RECEIPT or CR

The originator receives a message when the recipient has seen the letter. The letter is considered seen if the recipient displays or copies the letter to a file.

CERTIFY_NON_RECEIPT or CNR

The originator receives a message if the letter is deleted without having been seen or is autoforwarded. The mail system generates the nonreceipt notice if the letter is unseen at the time of a deletion or the letter is autoforwarded.

REPLY_REQUESTED or RR

The letter header information indicates that the originator is expecting a reply. The system does not require the recipient to reply.

If omitted, NONE is assumed.

Remarks

- When you enter the ADD_TO subcommand, the mail system performs the following address verification:
  - When deleting an address, you must specify the same parameters used to create that address, or the subcommand terminates with an error message.
  - If you enter an address that is in the directory, the system accepts the address.
  - If you enter an ambiguous address, the system displays a message to let you know and does not add the address to the recipient list.
  - If you enter an address that is not in the directory and if the organization attributes you give for that address specify the local host, the system displays an informative message.
  - If you enter an address that is not in the directory and if the organization attributes you give for that address specify a remote host, the system accepts the address. However, the address is not verified until the remote host attempts to deliver it.

- If you enter an address that is not in the directory and if the organization attributes you give for that address have not been specified by the Mail/VE administrator, the subcommand terminates with a warning message.

- There are three methods of adding a BITNet or Internet address to the primary address list. How your Mail/VE administrator configured your mail network determines which methods work. Select the one your Mail/VE administrator recommends.

  - Enter the PERSONAL_NAME parameter with a commercial at sign (@):

    ```
    personal_name='jdsmith@arhops.cdc.com'
    ```

    You can use this method any time. Because the commercial at sign indicates a BITNet or Internet address, the system automatically sends it to the gateway for routing.

  - Enter the address the same way you do a Mail/VE address. For example, enter `jdsmith@arhops.cdc.com` as follows:

    ```
    personal_name='jdsmith'
    organization_unit='arhops'
    organization_name='cdc'
    ```

    This method works only if your Mail/VE administrator has configured the Mail/VE gateway to map the organization name and/or organization units to a BITNet or Internet address of `arhops.cdc.com`.

  - Enter the DOMAIN_DEFINED_ATTRIBUTES parameter to specify the address. The characters (a) replace the commercial at sign in this format.

    ```
    domain_defined_attributes=( ..
    ('RCF-822','jdsmith(a)arhops.cdc.com'))
    ```

  If you use this method, you must also enter the organization name and/or organization units, private domain, administrative domain, and country of the Mail/VE gateway.

- If you enter a BITNet or Internet address, the CERTIFY_RECEIPT and CERTIFY_NON_RECEIPT options of the RECIPIENT_OPTIONS parameter do not apply.
- Distribution lists that cannot be expanded (that is, lists created with the EXPAND attribute set to FALSE), will ignore any values specified for the RECIPIENT_OPTIONS parameter.
- Local addresses are verified as entered. Entry of an ambiguous address results in an informative message. The address is not added to the recipient list. Entry of a nonexistent address also results in an informative message.

  Nonlocal addresses are not verified until delivery is attempted.
- The POSTAL_ADDRESS, POSTAL_ADDRESS_COUNTRY_NAME, and POSTAL_ADDRESS_CODE parameters do not define a mailbox address in the mail directory. They are used to send mail via physical postal delivery.
- You cannot use the wildcard character * in the addresses you specify.
- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example creates a To address list made up of distribution lists:

```
Mail/write_letter
Wril/addlp ($local.note, .HR.Study)
Wril/subject 'Results of Employee Survey'
Wril/add_to 'Engineering_Department'
Wril/add_to 'Quality_Assurance'
Wril/add_to 'Test_Group'
Wril/end_write_letter
```

$ANY_MAIL
MAIL Function

Purpose Returns a boolean value telling you whether there is mail that meets the criteria you specify on the parameters.

Format

$ANY_MAIL
(*ADDRESS: string* or *record*
MAIL_OPTIONS: keyword or *list of keyword*)

Parameters

ADDRESS

Specifies the mailbox to be checked for mail. Only the personal name is required. Specify the organization unit(s) and organization name associated with the mailbox to further define the address. Enter the address as a string or as a record in the format:

```
record
  personal_name: record
    surname: string 1..40
    given_name: string 0..16
    initials: string 0..5
    generation_qualifier: string 0..3
  recend
  organization_units: list 0..4 of string 0..32
  organization_name: string 0..64
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need more information on specifying mailbox names.

If omitted, the value returned by the $IDENTITY function is used.

MAIL_OPTIONS

Specifies the types of letters to be included in the letter list from the mailbox checked for mail. You can specify one or more options for each option type.

If you select more than one option for each type, each selection applies. Selecting all options for any type has the same effect as selecting none of the options for that type.

Letter types:

LETTER or L

Lists all letters.

RECEIPTS or R

Displays letters generated upon receipt or nonreceipt of a letter.

Read Status types:

These options apply only to owned mailboxes.

SEEN or S

Lists letters that have been displayed.

UNSEEN or US

Lists letters that have not been displayed.

Sensitivity types:

PRIVATE or PR

Lists private letters.

PERSONAL or PE

Lists personal letters.

CONFIDENTIAL or CO

Lists confidential letters.

Receipt Certification types:

CERTIFIED or C

Lists certified letters.

UNCERTIFIED or UC

Lists letters not classified as certified.

Importance types:

LOW_IMPORTANCE or LI

Lists letters the sender assigned low importance.

NORMAL_IMPORTANCE or NI

Lists letters the sender assigned normal importance.

HIGH_IMPORTANCE or HI

Lists letters the sender assigned high importance.

Delivery Priority types:

NORMAL_PRIORITY or NP

Lists letters the sender assigned normal delivery priority.

URGENT_PRIORITY or UP

Lists letters the sender assigned urgent delivery priority.

NON_URGENT_PRIORITY or NUP

Lists letters the sender assigned nonurgent delivery priority.

Other types:

REPLY_REQUEST or RR

Lists letters for which the sender requested a reply.

AUTOFORWARDED or A

Lists letters that have been automatically forwarded from another mailbox.

ALL

Lists all mail.

If MAIL_OPTIONS is omitted, ALL is used.

Remarks

- The $ANY_MAIL function returns TRUE if any mail is found that satisfies the criteria specified by the function parameters.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples

In the following example, $ANY_MAIL is used to check for mail prior to executing LIST_MAIL:

```
Mail/if $any_mail((('olson' 'jane') 'umb')),
reply_request) then
Mail/list_mail
Mail/ifend
```

CHANGE_DISTRIBUTION_LIST MAIL Subcommand

Purpose Initiates the CHANGE_DISTRIBUTION_LIST subutility through which you can change the definition of any distribution list you own. When you enter the CHANGE_DISTRIBUTION_LIST subcommand, the system displays the prompt `Chadl/`.

Format

CHANGE_DISTRIBUTION_LIST or
CHADL
DISTRIBUTION_LIST_NAME = **string** or **record**
ORGANIZATION_UNITS = *list of string*
ORGANIZATION_NAME = *string*
STATUS = *status variable*

Parameters

DISTRIBUTION_LIST_NAME or **DLN**

Specifies the name of the distribution list to be changed. Enter the name as a 1- to 64-character string or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

Refer to the Mail/VE Version 2 manual for further information on specifying a distribution list name.

This parameter is required.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the distribution list to be changed. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the distribution list. Enter the name as a 1- to 64-character string.

Remarks

- Use the CHANGE_DISTRIBUTION_LIST subutility subcommands, listed next, to specify new distribution list attributes, edit the address list, and edit the access permission list of a group distribution list.

  ADD_ADDRESS
  ADD_PERMISSION
  DELETE_ADDRESS
  DELETE_PERMISSION
  DISPLAY_ADDRESS_LIST
  DISPLAY_ATTRIBUTES
  DISPLAY_PERMISSIONS
  END_CHANGE_DISTRIBUTION_LIST
  SET_ATTRIBUTES

  These subcommands are described under the Mailbox and Distribution List Subutility Subcommands section of the Mail/VE Version 2 manual.

- Only the owner of a distribution list (that is, the user who created the list) or the Mail/VE administrator can change the list.

- You cannot use the wildcard character * in address parameters.

- If the addressing parameters do not uniquely identify a distribution list you own, the CHANGE_DISTRIBUTION_LIST subutility terminates and displays an error message.

- The ORGANIZATION_UNIT and ORGANIZATION_NAME parameters are used to distinguish distribution lists with the same name and to facilitate transfer within a multihost network.

- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list defined organization names/units.

- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example adds an address to an existing distribution list.

```
Mail/change_distribution_list ..
Mail../dln='project_members'
Chadl/add_address personal_name='Janet Strong'
Chadl/end_change_distribution_list
```

The following example changes a private distribution list to a group distribution list.

```
Mail/change_distribution_list ..
Chadl/distribution_list_name='tiger_team'
Chadl/set_attributes permit_type=group
Chadl/add_permission ..
Chadl../organization_name='qa department'
Chadl/add_permission ..
Chadl../organization_name= ..
Chadl../'Continuous Improvement Task Force'
Chadl/end_change_distribution_list
```

CHANGE_MAILBOX
MAIL Subcommand

Purpose Initiates the CHANGE_MAILBOX subutility, through which you change the definition of an existing mailbox. When you enter the CHANGE_MAILBOX subcommand, the system displays the prompt `Cham/`.

Format

CHANGE_MAILBOX or
CHAM
PERSONAL_NAME=string or **record**
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
STATUS=status variable

Parameters

PERSONAL_NAME or PN

Specifies the name of the mailbox to be changed. Enter the name as a 1- to 64-character string or a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

Refer to Address Names in the Mail/VE Version 2 Usage manual if you need further information on specifying a mailbox name.

This parameter is required.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the mailbox to be changed. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the mailbox to be changed. Enter a string of 1 to 64 characters.

Remarks

- Use the CHANGE_MAILBOX subutility subcommands, listed next, to define or change the mailbox attributes and edit the list of permissions that define a group mailbox.

  ADD_PERMISSION
  ACTIVATE_AUTO_FORWARDING
  DEACTIVATE_AUTO_FORWARDING
  DELETE_PERMISSION
  DISPLAY_ATTRIBUTES
  DISPLAY_PERMISSIONS
  END_CHANGE_MAILBOX

SET_ATTRIBUTES

These subcommands are described under the Mailbox and Distribution List Subutility Subcommands section of the Mail/VE Version 2 manual.

- Only the owner of a mailbox or the MailVE administrator can change the list.
- Changing a group mailbox to private or public deletes the list of permissions that define the group.
- If the addressing parameters do not specify a mailbox you own, the CHANGE_MAILBOX subutility terminates and the system displays an error message to let you know the problem.
- You cannot use the wildcard character * in address parameters.
- If you are not validated for self-administration, you cannot alter any of the mailbox address attributes or the mailbox retention period using the SET_ATTRIBUTE subcommand of the CHANGE_MAILBOX subutility. See your Mail/VE administrator if you have questions.
- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list defined organization names/units.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example adds a permission to a group mailbox permission list:

```
Mail/change_mailbox personal_name='Project_Status'
Cham/add_permission personal_name='Janet Strong' ..
Cham../ou=('Field Support' 'Operations')
Cham/end_change_mailbox
```

COPY_LETTER_PARTS MAIL Subcommand

Purpose Copies parts of a letter body to a file.

Format

COPY_LETTER_PARTS or
COPY_LETTER_PART or
COPLP
LETTER=keyword or *integer* or *string*
PARTS=keyword or **list of range of integer**
OUTPUT=file
STATUS=status variable

Parameters *LETTER* or *L*

Specifies the letter to be copied. The value specified can be a keyword, letter number, or letter identifier.

Keyword options:

CURRENT or C

Copies letter parts from the current letter in the letter list. The current letter is the letter most recently displayed.

FIRST or F

Copies letter parts from the first letter in the letter list.

LAST or L

Copies letter parts from the last letter in the letter list.

NEXT or N

Copies letter parts from the letter following the current letter in the letter list.

PREVIOUS or P

Copies letter parts from the letter preceding the current letter in the letter list.

The letter number is the number assigned to the letter on the letter list. It can change every time you enter the LIST_MAIL subcommand. The letter identifier is permanently assigned to the letter by Mail/VE when it is delivered to the mailbox.

If LETTER is omitted, CURRENT is used.

PARTS or **PART** or **P**

Parameter Attributes: BY_NAME

Specifies the letter parts to be copied. The value specified can be a list of letter part ordinals, a range of letter part ordinals, or a keyword. Integer values can range from 1 to the number of parts that make up the letter body.

You can determine the number of letter parts by executing the LIST_MAIL subcommand with the DISPLAY_OPTIONS parameter set to FULL or by displaying the value of the $LETTER_ATTRIBUTE function with the OPTION parameter set to NUMBER_OF_LETTER_PARTS.

The following keyword options correspond to possible letter part types displayed when you read a letter using the READ_LETTER subcommand.

TEXT or T

Copies ASCII text type letter parts from the letter and writes them to the output file.

ENCRYPTED or E

X.400 data type.

GROUP_3_FACSIMILE or G3FAX

X.400 data type.

NATIONALLY_DEFINED or ND

X.400 data type.

SIMPLE_FORMATTABLE_DOCUMENT or SFD

X.400 data type.

TELEX or TLX

X.400 data type.

TELETEX or TTX

X.400 data type.

TEXT_INTERCHANGE_FORMAT_0 or TIF0

X.400 data type.

TEXT_INTERCHANGE_FORMAT_1 or TIF1

X.400 data type.

UNDEFINED or U

Copies the non-ASCII type letter parts from the letter and writes them to the file specified on the OUTPUT parameter.

VIDEOTEX or V

X.400 data type.

VOICE_DATA or VD

X.400 data type.

This parameter is required.

OUTPUT or O

Parameter Attributes: BY_NAME

Specifies the name of the file to which the letter parts are written. This parameter is required.

Remarks

- Mail/VE displays a message to tell you if the letter identifier does not exist, if no letter list exists, or if the letter number is not in the letter list.
- Only TEXT and NONTEXT letter part types can be created using Mail/VE. Other letter types can be relayed to an X.400 system that provides a service for processing the data (such as a fax machine).
- If you want to use letter numbers to specify letters to be copied, first enter the LIST_MAIL subcommand to create a numbered letter list.
- The $LETTER function returns the letter identifier of the letter that is copied.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example copies parts 1, 3, and 5 through 9 of the current letter into a file named LETTER_BODY_FILE:

```
Mail/copy_letter_parts current ..
Mail../parts=(1, 3, 5..9) ..
Mail../output=letter_body_file
```

CREATE_DISTRIBUTION_LIST MAIL Subcommand

Purpose Initiates the CREATE_DISTRIBUTION_LIST subutility, through which you create a distribution list. When you enter the CREATE_DISTRIBUTION_LIST subcommand, the system displays the prompt `Credl/`.

Format **CREATE_DISTRIBUTION_LIST** or
CREDL
STATUS=status variable

Remarks

- Use the CREATE_DISTRIBUTION_LIST subcommands, listed next, to define the distribution list.

  ADD_ADDRESS
  ADD_PERMISSION
  DELETE_ADDRESS
  DELETE_PERMISSION
  DISPLAY_ADDRESS_LIST
  DISPLAY_ATTRIBUTES
  DISPLAY_PERMISSIONS
  END_CREATE_DISTRIBUTION_LIST
  SET_ATTRIBUTES

  These subcommands are described under the Mailbox and Distribution List Subutility Subcommands chapter of the Mail/VE Version 2 Usage manual.

- The address attributes you specify for the distribution list through the SET_ATTRIBUTES subcommand must uniquely identify the list you are creating. If the address is not unique when you enter the END_CREATE_DISTRIBUTION_LIST subcommand, the distribution list is not created and the subutility terminates with an error message.

- Your distribution list can be made up of the addresses of mailboxes and/or other distribution lists.
- Your Mail/VE validation specifies the maximum number of distribution lists you can have. If you attempt to exceed that number, Mail/VE returns an error message.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example creates a distribution list made up of two addresses:

```
Mail/create_distribution_list
Credl/set_attribute ..
Credl../distribution_list_name=( ..
Credl../'Fellow_Trainees' 'dlft') ..
Credl../description='owned by Ben Churchill' ..
Credl../expand=true
Credl/add_address personal_name='Jim Rivers'
Credl/add_address ..
Credl../personal_name='Joan C Lohn-Cho'
Credl/end_create_distribution_list true
```

CREATE_MAILBOX MAIL Subcommand

Purpose Initiates the CREATE_MAILBOX subutility, through which you can create a mailbox. When you enter the CREATE_MAILBOX subcommand, the system displays the prompt Crem/.

Format

CREATE_MAILBOX or
CREM
STATUS=status variable

Remarks

- Use the CREATE_MAILBOX subutility subcommands, listed next, to define the mailbox.

  ACTIVATE_AUTO_FORWARDING
  ADD_PERMISSION
  DEACTIVATE_AUTO_FORWARDING
  DELETE_PERMISSION
  DISPLAY_ATTRIBUTES
  DISPLAY_PERMISSIONS
  END_CREATE_MAILBOX

SET_ATTRIBUTES

These subcommands are described under the Mailbox and Distribution List Subutility Subcommands section of the Mail/VE Version 2 Usage manual.

- Your Mail/VE validation set by the mail system administrator specifies the maximum number of mailboxes you can have. If you attempt to exceed that number, Mail/VE returns an error message.
- You can create a mailbox at any time during a Mail/VE session if your NOS/VE system administrator has validated you for self-administration. If you are not validated for self-administration, the mail administrator at your site will create mailboxes for you; if you try to create a mailbox, the CREATE_MAILBOX subutility terminates with an error status.
- Use the CHANGE_MAILBOX subcommand to change an existing mailbox definition.
- The address values assigned to the mailbox through the SET_ ATTRIBUTES subcommand must uniquely define the mailbox you want to create. If the mailbox address is not unique in the mail directory at the time you enter the END_CREATE_MAILBOX subcommand, the mailbox is not created and the subutility terminates with an error message.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example creates a mailbox:

```
Mail/create_mailbox
Crem/set_attributes ..
Crem../personal_name = ( ..
Crem../ 'Allison J McGinty' ..
Crem../ 'Ali McGinty' ..
Crem../ 'AJM') ..
Crem../organization_unit = ( ..
Crem../  'Site A' ..
Crem../  'Customer Support' ..
Crem../  'Engineering Department' ..
Crem../  'Advanced Products Division') ..
Crem../organization_name = 'Computer Services' ..
Crem../description='owned by Allison McGinty'
Crem/end_create_mailbox
```

DEACTIVATE_AUTO_FORWARDING CHAM and CREM Subcommand

Purpose Disables automatic forwarding of mail from the mailbox being created or changed.

Format **DEACTIVATE_AUTO_FORWARDING** or **DEAAF**

STATUS=status variable

Remarks

- If automatic forwarding is not enabled for the mailbox being created or changed, this command has no effect.
- For more information, see the Mail/VE Version 2 Usage manual.

DEACTIVATE_SCREEN MAIL Subcommand

Purpose Changes the interaction style with Mail/VE from screen mode to line mode.

Format **DEACTIVATE_SCREEN** or **DEAS**

STATUS=status variable

Remarks

- You can use this subcommand from any screen that has an active home line.
- When you enter the DEACTIVATE_SCREEN subcommand, the line mode prompt `Mail/` appears. Use the ACTIVATE_SCREEN subcommand to return to screen mode.
- If you enter DEACTIVATE_SCREEN from a primary screen, such as the Write Letter screen, the information entered on the screen is preserved. If you enter DEACTIVATE_SCREEN from a subscreen that has an active home line, such as the Primary Addresses subscreen accessed from the Write Letter screen, you lose any entries made on that subscreen prior to entering DEACTIVATE_ SCREEN. Entering DEACTIVATE_SCREEN from a subscreen is equivalent to pressing `Cancel`.
- DEACTIVATE_SCREEN applies only to the current operation; it does not permanently change your interaction style to line mode. For example, if you are working in screen mode and enter DEACTIVATE_SCREEN, Mail/VE switches you to line mode. When you specify another operation, by entering WRITE_LETTER for example, Mail/VE returns to screen mode. Your mode of interaction with Mail/VE is determined by whether the STYLE parameter on the NOS/VE CHANGE_INTERACTION_STYLE command is set to SCREEN or LINE.
- For more information, see the Mail/VE Version 2 Usage manual.

$DEFAULT_MAILBOX
MAIL Function

Purpose Returns a record containing the local address of the mailbox that is designated as your default mailbox. The default mailbox is automatically used as your identity to the mail system when you begin a Mail/VE session.

Format **$DEFAULT_MAILBOX**

Parameters None.

Remarks

- The first mailbox created by or for a user serves as the default mailbox until changed. Use the SET_DEFAULT_MAILBOX subcommand to change the default mailbox.
- The $DEFAULT_MAILBOX function returns the local address record in the format:

```
record
  personal_name: record
    surname: string 1..40
    given_name: string 0..16
    initials: string 0..5
    generation_qualifier: string 0..3
  recend
  organization_units: list of 0..4 of string 0..32
  organization_name: string 0..64
recend
```

- For more information, see the Mail/VE Version 2 Usage manual.

Examples

This example uses the NOS/VE command DISPLAY_VALUE to display the default mailbox data structure:

```
Mail/display_value $default_mailbox ..
Mail../display_option=data_structure

"RECORD"
  PERSONAL_NAME: "RECORD"
                   SURNAME: "STRING" 'Jones'
                   GIVEN_NAME: "STRING" 'John'
                   INITIALS: "STRING" 'J'
                   GENERATION_QUALIFIER:  "STRING" ''
                 "RECORD END"
  ORGANIZATION_UNITS: "LIST"
                        1: "STRING" 'CDCNET'
                      "LIST END"
  ORGANIZATION_NAME:  "STRING" ''
"RECORD END"
```

The following example uses the NOS/VE command DISPLAY_VALUE to display the default mailbox source:

```
Mail/display_value $default_mailbox..
Mail../display_option=source

(('Jones', 'John', 'J', ''), 'CDCNET', '')
```

DELETE_ADDRESS
CHADL and CREDL Subcommand

Purpose Deletes an address from a specified distribution list.

Format

DELETE_ADDRESS or
DELA
PERSONAL_NAME=string or *record*
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
PRIVATE_DOMAIN=string
ADMINISTRATIVE_DOMAIN=string
COUNTRY=string
UA_IDENTIFIER=string
TERMINAL_IDENTIFIER=string
X121_ADDRESS=string
DOMAIN_DEFINED_ATTRIBUTE=list of record
POSTAL_ADDRESS=list of string
POSTAL_ADDRESS_COUNTRY_NAME=string
POSTAL_ADDRESS_CODE=string
STATUS=status variable

Parameters *PERSONAL_NAME* or *DISTRIBUTION_LIST_NAME* or *PN* or *DLN*

Specifies the name of the mailbox or distribution list to be deleted from the distribution list you are changing or creating. Enter the name as a string of 1 to 256 characters or a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need more information on specifying addresses.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the address to be deleted. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the address to be deleted. Enter a 1- to 64-character string.

PRIVATE_DOMAIN or *PD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the private domain associated with the address to be deleted from the distribution list. Enter a 1- to 16-character string.

ADMINISTRATIVE_DOMAIN or *AD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the administrative domain associated with the address to be deleted from the distribution list. Enter a 1- to 16-character string.

COUNTRY or *C*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country associated with the address to be deleted from the distribution list. Enter a 1- to 3-character string.

UA_IDENTIFIER or *UI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the user agent (UA) address identifier relative to an administrative domain to be deleted from the distribution list. Enter the identifier as a 1- to 32-character string.

TERMINAL_IDENTIFIER or *TI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the terminal address identifier of a telematic terminal to be deleted from the distribution list. Enter the identifier as a 1- to 24-character string.

X121_ADDRESS or *XA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies an X.121 standard network address to be deleted from the distribution list. For example, an X.121 address might reference a fax machine or telex location. Enter the address as a 1- to 15-character string of integers 0 to 9.

DOMAIN_DEFINED_ATTRIBUTE or *DDA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the non-X.400 address attributes defined for a mail system to be deleted from the distribution list. Enter the attributes as a list of one to four records in the format:

```
record
  name: string 1..8
  value: string 1..128
recend
```

POSTAL_ADDRESS or *PA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the physical postal service address to be deleted. Enter the address as a list of two to six strings of 1 to 30 characters each.

POSTAL_ADDRESS_COUNTRY_NAME or *PACN*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country name for the physical postal service address to be deleted. Enter the country name as a 1- to 30-character string.

POSTAL_ADDRESS_CODE or *PAC*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the national code designation (for example, the zip code in the U.S.) of the physical postal service address to be deleted. Enter the code as a 1- to 128-character string.

Remarks

- You cannot use the wildcard character * in the addresses you specify.
- Addresses are processed exactly as they are entered. If a matching address is not found in the distribution list, the command terminates and the system displays an error message.
- The POSTAL_ADDRESS, POSTAL_ADDRESS_COUNTRY_NAME, and POSTAL_ADDRESS_CODE parameters do not define a mailbox address in the mail directory. They are used to send mail via physical postal delivery.
- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names and units.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example deletes an address from an existing distribution list:

```
Mail/change_distribution_list 'Photo_Club_News'
Chadl/delete_address 'Carol Parker' ..
Chadl../organization_name='Public_Relations'
Chadl/end_change_distribution_list
```

DELETE_BLIND_COURTESY_COPY FORL, WRIL, and WRIR Subcommand

Purpose Deletes an address from the blind courtesy copy address list.

Format

DELETE_BLIND_COURTESY_COPY or
DELBCC
PERSONAL_NAME=string or *record*
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
PRIVATE_DOMAIN=string
ADMINISTRATIVE_DOMAIN=string
COUNTRY=string
UA_IDENTIFIER=string
TERMINAL_IDENTIFIER=string

X121_ADDRESS=string
DOMAIN_DEFINED_ATTRIBUTE=list of record
POSTAL_ADDRESS=list of string
POSTAL_ADDRESS_COUNTRY_NAME=string
POSTAL_ADDRESS_CODE=string
STATUS=status variable

Parameters

PERSONAL_NAME or *DISTRIBUTION_LIST_NAME* or *PN* or *DLN*

Specifies the mailbox or distribution list name you want to delete from the blind courtesy copy list. Enter the name as a string of 1 to 256 characters or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need more information on specifying a name.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the address to be deleted from the blind courtesy copy list. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the address to be deleted from the blind courtesy copy list. Enter a 1- to 64-character string.

PRIVATE_DOMAIN or *PD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the private domain address associated with the address. Enter a 1- to 16-character string.

ADMINISTRATIVE_DOMAIN or *AD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the administrative domain associated with the address. Enter a 1- to 16-character string.

COUNTRY or *C*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country associated with the address. Enter a 1- to 3-character string.

UA_IDENTIFIER or *UI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the user agent (UA) address identifier relative to an administrative domain to be deleted. Enter the identifier as a 1- to 32-character string.

TERMINAL_IDENTIFIER or *TI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the terminal address identifier of a telematic terminal to be deleted. Enter the identifier as a 1- to 24-character string.

X121_ADDRESS or *XA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies an X.121 standard network address to be deleted. For example, an X.121 address might reference a fax machine or telex location. Enter the address as a 1- to 15-character string of integers 0 to 9.

DOMAIN_DEFINED_ATTRIBUTE or *DDA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the non-X.400 address attributes defined for a mail system to be deleted from the address list. Enter the attributes as a list of one to four records in the format:

```
record
  name: string 1..8
  value: string 1..128
recend
```

POSTAL_ADDRESS or *PA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the physical postal service address to be deleted. Enter the address as a list of two to six strings of 1 to 30 characters each.

POSTAL_ADDRESS_COUNTRY_NAME or *PACN*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country name for the physical postal service address to be deleted. Enter the country name as a 1- to 30-character string.

POSTAL_ADDRESS_CODE or *PAC*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the national code designation (for example, the zip code in the U.S.) of the physical postal service address to be deleted. Enter the code as a 1- to 128-character string.

Remarks

- You cannot use the wildcard character * in addresses you specify.
- When deleting an address, you must specify the same parameters used to create that address, or the subcommand terminates with a warning message.
- The POSTAL_ADDRESS, POSTAL_ADDRESS_COUNTRY_NAME, and POSTAL_ADDRESS_CODE parameters do not define a mailbox address in the mail directory. They identify the physical postal delivery address to be deleted.
- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example deletes an address from the blind courtesy copy address list:

```
Wril/display_blind_courtesy_copy
          Personal name: Ben Krantz
     Organization units: PERSONNEL

          Personal name: Alan Wescott
     Organization units: PERSONNEL

          Personal name: Carrie Young
     Organization units: LEGAL
Wril/delete_blind_courtesy_copy ..
Wril../personal_name='Ben Krantz'
```

DELETE_COURTESY_COPY FORL, WRIL, and WRIR Subcommand

Purpose Deletes an address from the courtesy copy address list.

Format

DELETE_COURTESY_COPY or
DELCC
PERSONAL_NAME=string or *record*
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
PRIVATE_DOMAIN=string
ADMINISTRATIVE_DOMAIN=string
COUNTRY=string
UA_IDENTIFIER=string
TERMINAL_IDENTIFIER=string
X121_ADDRESS=string
DOMAIN_DEFINED_ATTRIBUTE=list of record
POSTAL_ADDRESS=list of string
POSTAL_ADDRESS_COUNTRY_NAME=string
POSTAL_ADDRESS_CODE=string
STATUS=status variable

Parameters *PERSONAL_NAME* or *DISTRIBUTION_LIST_NAME* or *PN* or *DLN*

Specifies the mailbox or distribution list you want to delete from the courtesy copy address list. Enter the name as a string of 1 to 256 characters or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 manual if you need more information on specifying Usage an address.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the address to be deleted from the courtesy copy address list. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the address to be deleted from the courtesy copy list. Enter a 1- to 64-character string.

PRIVATE_DOMAIN or *PD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the private domain associated with the address to be deleted from the courtesy copy address list. Enter a 1- to 16-character string.

ADMINISTRATIVE_DOMAIN or *AD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the administrative domain associated with the address to be deleted from the courtesy copy address list. Enter a 1- to 16-character string.

COUNTRY or *C*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country associated with the address to be deleted from the courtesy copy address list. Enter a 1- to 3-character string.

UA_IDENTIFIER or *UI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the user agent (UA) address identifier relative to an administrative domain to be deleted. Enter the identifier as a 1- to 32-character string.

TERMINAL_IDENTIFIER or *TI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the terminal address identifier of a telematic terminal to be deleted. Enter the identifier as a 1- to 24-character string.

X121_ADDRESS or *XA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies an X.121 standard network address to be deleted from the address list. For example, an X.121 address might reference a fax machine or telex location. Enter the address as a 1- to 15-character string of integers 0 to 9.

DOMAIN_DEFINED_ATTRIBUTE or *DDA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the non-X.400 address attributes defined for a mail system to be deleted from the address list. Enter the attributes as a list of one to four records in the format:

```
record
  name: string 1..8
  value: string 1..128
recend
```

POSTAL_ADDRESS or *PA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the physical postal service address to be deleted. Enter the address as a list of two to six strings of 1 to 30 characters each.

POSTAL_ADDRESS_COUNTRY_NAME or *PACN*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country name for the physical service address to be deleted. Enter the country name as a 1- to 30-character string.

POSTAL_ADDRESS_CODE or *PAC*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the national code designation (for example, the zip code in the U.S.) of the physical postal service address to be deleted. Enter the code as a 1- to 128-character string.

Remarks

- You cannot use the wildcard characters * in addresses you specify.
- When deleting an address, you must specify the same parameters used to create that address, or the subcommand terminates with a warning message.
- The POSTAL_ADDRESS, POSTAL_ADDRESS_COUNTRY_NAME, and POSTAL_ADDRESS_CODE parameters do not define a mailbox address in the mail directory. They identify the physical postal delivery address to be deleted.
- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples

The following example deletes an address from the courtesy copy address list:

```
Wril/display_courtesy_copy
          Personal name: Cris Fedo
     Organization units: SALES

          Personal name: Glen Kalin
     Organization units: SALES
          Personal name: Melanie H Hart
     Organization units: SALES
Wril/delete_courtesy_copy ..
Wril../personal_name='Chris Fedo'
```

DELETE_DISTRIBUTION_LIST MAIL Subcommand

Purpose Removes a distribution list from the Mail/VE directory.

Format

DELETE_DISTRIBUTION_LIST or
DELDL
DISTRIBUTION_LIST_NAME=string or **record**
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
STATUS=status variable

Parameters

DISTRIBUTION_LIST_NAME or **DLN**

Specifies the name of the distribution list to be deleted. Enter the name as a 1- to 64-character string or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need more information on specifying a distribution list name.

This parameter is required.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the distribution list to be deleted. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the distribution list to be deleted. Enter a string of 1 to 64 characters.

Remarks

- The name of the distribution list is not deleted from other distribution lists that contain it.
- Only the owner of a distribution list or the Mail/VE administrator can delete the list.
- If the address specified by the addressing attribute values does not uniquely identify a distribution list you own, the command terminates with an error message.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples

The following example deletes the AD_HOC_COMMITTEE distribution list from the Mail/VE directory:

```
Mail/delete_distribution_list ..
Mail../dln='ad_hoc_committee'
```

DELETE_LETTER MAIL Subcommand

Purpose

Deletes a letter from a mailbox.

Format

DELETE_LETTER or
DELL
LETTER=integer or *string*
STATUS=status variable

Parameters

LETTER or *L*

Specifies the letter to be deleted. Identify the letter by letter number or letter identifier.

The letter number is the number assigned to the letter on the letter list displayed when you enter the LIST_MAIL subcommand. It can change each time you use LIST_MAIL. The letter identifier is permanently assigned to the letter by Mail/VE when it is delivered to the mailbox.

Remarks

- Only the owner of a mailbox or the Mail/VE administrator can delete letters from a mailbox.
- If no letter corresponds to the letter number or letter identifier, Mail/VE writes a message to the error file.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example deletes letter number 1 in the letter list:

```
Mail/delete_letters letter=1
```

In the following example, the letter to be deleted is specified using the letter identifier:

```
Mail/delete_letters letter='GW14'
```

DELETE_LETTER_PARTS FORL, WRIL, and WRIR Subcommand

Purpose Deletes an entry from the list of parts that make up the body of a letter.

Format

DELETE_LETTER_PARTS or
DELETE_LETTER_PART or
DELLP
LETTER_PARTS=list of file
STATUS=status variable

Parameters

LETTER_PARTS or **LETTER_PART** or **LP**

List of letter parts to be deleted from the letter body. This parameter is required.

Remarks

- If the letter part to be deleted is not in the letter body, the command terminates with an error status.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example deletes a letter part from the list of parts that make up the letter body:

```
Wril/display_letter_parts
 :NVE.LTR.PROCESS.MONTHLY_REVIEW (TEXT)
 :NVE.JMW.EXAMPLES.P0527 (TEXT)
 :NVE.JMW.EXAMPLES.P0531 (TEXT)
 :NVE.JMW.EXAMPLES.P0533 (TEXT)
 :NVE.JMW.EXAMPLES.P05116 (TEXT)
Wril/delete_letter_part ..
Wril/letter_part=.ltr.process.monthly_review
```

DELETE_MAILBOX
MAIL Subcommand

Purpose Deletes a mailbox and all letters in the mailbox from the Mail/VE directory.

Format

DELETE_MAILBOX or
DELM
PERSONAL_NAME=string or **record**
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
STATUS=status variable

Parameters

PERSONAL_NAME or **PN**

Specifies the name of the mailbox to be deleted. Enter the name as a 1- to 64-character string or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need details on naming the mailbox.

This parameter is required.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the mailbox to be deleted. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the mailbox to be deleted. Enter a string of 1 to 64 characters.

Remarks

- You can only delete a mailbox if you have been validated for self-administration by the NOS/VE system administrator.
- If you delete the mailbox that is your default mailbox identity, you must select a new default and identity mailbox to continue your mail session. Use the SET_DEFAULT_MAILBOX and SELECT_IDENTITY subcommands to specify a default and identity mailbox. (See also the Examples section, below.)
- A mailbox can only be deleted by its owner (that is, the user who created it). The deleted mailbox address is not deleted from distribution lists that contain it.
- If the address specified by the addressing parameters does not uniquely identify a mailbox you own, the command terminates with an error message.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples

The following example deletes a mailbox:

```
Mail/delete_mailbox ..
Mail../personal_name='Program_News' ..
Mail../organization_unit=( ..
Mail../'SL410C' ..
Mail../'Purchasing' ..
Mail../'Consumer Products')
```

In the following example, the user deletes the mailbox serving as the identity to Mail/VE. After the system displays an informative message, the user then specifies a new identity using SELECT_IDENTITY and SET_DEFAULT_MAILBOX.

```
Mail/delete_mailbox ..
Mail../personal_name='Lonnie Mason'
 --You do not have a mailbox selected.
Mail/select_identity ..
Mail../personal_name='Project_Coordinator'
Mail/set_default_mailbox ..
Mail../personal_name='Project_Coordinator'
```

DELETE_PERMISSION CHADL, CHAM, CREDL, and CREM Subcommand

Purpose Deletes a mailbox address from the list of permitted mailboxes for the distribution list or mailbox you are changing or creating. This subcommand applies only to GROUP mailboxes and distribution lists.

Format

DELETE_PERMISSION or
DELP
PERSONAL_NAME=string or *record*
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
PRIVATE_DOMAIN=string
ADMINISTRATIVE_DOMAIN=string
COUNTRY=string
STATUS=status variable

Parameters *PERSONAL_NAME* or *PN*

Specifies the name of the mailbox to be deleted from a group definition. Enter the name as a 1- to 256-character string or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need more information on specifying a mailbox name.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the address to be deleted from the permitted group of addresses. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the address to be deleted from the permitted group. Enter a 1- to 64-character string.

PRIVATE_DOMAIN or *PD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the private domain associated with the address to be deleted. Enter a 1- to 16-character string.

ADMINISTRATIVE_DOMAIN or *AD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the administrative domain associated with the address to be deleted from the permitted group. Enter a 1- to 16-character string.

COUNTRY or *C*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country associated with the address to be deleted from the permitted group. Enter a 1- to 3-character string.

Remarks

- You can use the wildcard character * in the address attributes.
- Addresses are processed exactly as they are entered.
- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.

- For more information, see the Mail/VE Version 2 Usage manual.

Examples In the following example, two mailboxes are deleted from the list of permissions for a group mailbox:

```
Mail/change_mailbox 'Demo_Log'
Cham/delete_permission 'Jack Holland'
Cham/delete_permission 'Laura Phillips'
Cham/end_change_mailbox
```

DELETE_TO
FORL, WRIL, and WRIR Subcommand

Purpose Deletes an address from the To (primary) address list.

Format

DELETE_TO or
DELT
PERSONAL_NAME=string or *record*
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
PRIVATE_DOMAIN=string
ADMINISTRATIVE_DOMAIN=string
COUNTRY=string
UA_IDENTIFIER=string
TERMINAL_IDENTIFIER=string
X121_ADDRESS=string
DOMAIN_DEFINED_ATTRIBUTE=list of record
POSTAL_ADDRESS=list of string
POSTAL_ADDRESS_COUNTRY_NAME=string
POSTAL_ADDRESS_CODE=string
STATUS=status variable

Parameters *PERSONAL_NAME* or *DISTRIBUTION_LIST_NAME* or *PN* or *DLN*

Specifies the mailbox or distribution list name you want to delete from the To address list. Enter the name as a string of 1 to 256 characters or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need information on specifying a mailbox or distribution list name.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the address to be deleted from the To address list. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the address to be deleted from the To address list. Enter a 1- to 64-character string.

PRIVATE_DOMAIN or *PD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the private domain associated with the address to be deleted from the To address list. Enter a 1- to 16-character string.

ADMINISTRATIVE_DOMAIN or *AD*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the administrative domain associated with the address to be deleted from the To address list. Enter a 1- to 16-character string.

COUNTRY or *C*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country associated with the address to be deleted from the To address list. Enter a 1- to 3-character string.

UA_IDENTIFIER or *UI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the user agent (UA) address identifier relative to an administrative domain to be deleted from the address list. Enter the identifier as a 1- to 32-character string.

TERMINAL_IDENTIFIER or *TI*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the terminal address identifier of a telematic terminal to be deleted from the address list. Enter the identifier as a 1- to 24-character string.

X121_ADDRESS or *XA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies an X.121 standard network address to be deleted from the address list. For example, an X.121 address might reference a fax machine or telex location. Enter the address as a 1- to 15-character string of integers 0 to 9.

DOMAIN_DEFINED_ATTRIBUTE or *DDA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the non-X.400 address attributes defined for a mail system to be deleted from the address list. Enter the attributes as a list of one to four records in the format:

```
record
  name: string 1..8
  value: string 1..128
recend
```

POSTAL_ADDRESS or *PA*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the physical postal service address to be deleted. Enter the address as a list of two to six strings of 1 to 30 characters each.

POSTAL_ADDRESS_COUNTRY_NAME or *PACN*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the country name for the physical postal service address to be deleted. Enter the country name as a 1- to 30-character string.

POSTAL_ADDRESS_CODE or *PAC*

Parameter Attributes: BY_NAME, ADVANCED

Specifies the national code designation (for example, the zip code in the U.S.) of the physical postal service address to be deleted. Enter the code as a 1- to 128-character string.

Remarks

- You cannot use the wildcard characters * in addresses.
- When deleting an address, you must specify the same parameters used to create that address, or the subcommand terminates with a warning message.
- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example deletes an address from the To address list:

```
Wril/display_to
          Personal name: Karen McDonald
     Organization units: PRODUCT RESEARCH
      Organization name: MARKETING

          Personal name: Cecil Pfeiffer
     Organization units: US SALES
                         MIDWEST DISTRICT
      Organization name: MARKETING

          Personal name: Michael Carpenter
     Organization units: US SALES
                         NORTHEAST DISTRICT
      Organization name: MARKETING

          Personal name: Nancy A. Reed
     Organization units: US SALES
                         NORTHEAST DISTRICT
      Organization name: MARKETING
Wril/delete_to ..
Wril../personal_name='Cecil Pfeiffer'
```

DISPLAY_ATTRIBUTES
CHADL, CHAM, CREDL, and CREM Subcommand

Purpose Displays mailbox or distribution list attributes.

Format

DISPLAY_ATTRIBUTES or
DISPLAY_ATTRIBUTE or
DISA
OUTPUT=file
STATUS=status variable

Parameters

OUTPUT or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file to which the display is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Examples The following example creates a mailbox and then requests a display of the attributes of the new mailbox:

```
Mail/create_mailbox
Crem/set_attributes ..
Crem../personal_name=( ..
Crem../ 'Terence Johnson' ..
Crem../ 'Terry Johnson' ..
Crem../ 'TJJ') ..
Crem../organization_name='music group ..
Crem../description='owned by Terry Johnson'
Crem/display_attributes
```

DISPLAY_ATTRIBUTES
CHAM and CREM Subcommand

Purpose Displays mailbox or distribution list attributes.

Format

DISPLAY_ATTRIBUTES or
DISPLAY_ATTRIBUTE or
DISA
OUTPUT=file
STATUS=status variable

Parameters *OUTPUT* or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file to which the display is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks For more information, see the Mail/VE Version 2 Usage manual.

DISPLAY_ADDRESS _LIST
CHADL and CREDL Subcommand

Purpose Displays the addresses that make up a distribution list.

Format

DISPLAY_ADDRESS _LIST or
DISAL
DISPLAY_OPTIONS=keyword
OUTPUT=file
STATUS=status variable

Parameters *DISPLAY_OPTIONS* or *DISPLAY_OPTION* or *DO*

Specifies the information displayed. Options:

BRIEF or B

Displays local address attributes.

FULL or F

Displays all address attributes.

If omitted, BRIEF is used.

OUTPUT or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file to which the display is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks

- If there are no addresses in the distribution list, the command terminates and the system displays an informative message.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples

The following example produces a brief version of an address list display.

```
Mail/change_distribution_list 'photo club dl'
Chadl/display_address_list
     Personal name: Trevor Porter
Organization units: EDUCATION
 Organization name: MASTER ENTERPRISE
     Personal name: Mary Stockman
Organization units: EDUCATION
 Organization name: MASTER ENTERPRISE
     Personal name: Brad Collins
Organization units: EDUCATION
 Organization name: MASTER ENTERPRISE
     Personal name: Raymond Ellerbee
Organization units: ACCOUNTING
 Organization name: MASTER ENTERPRISE
     Personal name: Denise Jones
Organization units: ACCOUNTING
 Organization name: MASTER ENTERPRISE
     Personal name: Kate Martin
Organization units: ACCOUNTING
 Organization name: MASTER ENTERPRISE
```

DISPLAY_BLIND _COURTESY_COPY FORL, WRIL, and WRIR Subcommand

Purpose Displays the blind courtesy copy address list.

Format

DISPLAY_BLIND _COURTESY_COPY or
DISBCC
DISPLAY_OPTIONS=keyword
OUTPUT=file
STATUS=status variable

Parameters *DISPLAY_OPTIONS* or *DISPLAY_OPTION* or *DO*

Specifies the information that is to appear in the display. Options are:

BRIEF or B

Displays local address attributes.

FULL or F

Displays address attributes and recipient options for each recipient.

If DISPLAY_OPTIONS is omitted, BRIEF is used.

OUTPUT or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file to which the display is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks

- If there are no addresses in the blind courtesy copy list, the command terminates with an informative message.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example produces the brief version of the address list display:

```
Wril/display_blind_courtesy_copy
          Personal name: Alan Wescott
     Organization units: PERSONNEL

          Personal name: Carrie Young
     Organization units: LEGAL
```

The following example produces the full version of the address list display:

```
Wril/display_blind_courtesy_copy ..
Wril../display_option=full
          Personal name: Alan Wescott
     Organization units: PERSONNEL
         Private domain: TBN INC.
  Administrative domain: ATTMAIL
                Country: US

          Personal name: Carrie Young
     Organization units: LEGAL
         Private domain: TBN INC.
  Administrative domain: ATTMAIL
                Country: US
```


DISPLAY_COURTESY_COPY FORL, WRIL, and WRIR Subcommand

Purpose Displays the courtesy copy address list.

Format

DISPLAY_COURTESY_COPY or
DISCC
DISPLAY_OPTIONS=keyword
OUTPUT=file
STATUS=status variable

Parameters *DISPLAY_OPTIONS* or *DISPLAY_OPTION* or *DO*

Specifies the information that is to appear in the display. Options are:

BRIEF or B

Displays local address attributes.

FULL or F

Displays address attributes and recipient options for each recipient.

If DISPLAY_OPTIONS is omitted, BRIEF is used.

OUTPUT or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file to which the display is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks

- If there are no addresses in the courtesy copy list, the command terminates with an informative message.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples

The following example produces the brief version of an address list display.

```
Wri1/display_courtesy_copy
          Personal name: Cris Fedo
     Organization units: SALES

          Personal name: Glen Kalin
     Organization units: SALES

          Personal name: Melanie H Hart
     Organization units: SALES
```

The following example produces the full version of an address list display.

```
Wril/display_courtesy_copy do=full
          Personal name: Cris Fedo
     Organization units: SALES
         Private domain: TANNER GROUP
  Administrative domain: ATTMAIL
                Country: US

          Personal name: Glen Kalin
     Organization units: SALES
         Private domain: TANNER GROUP
  Administrative domain: ATTMAIL
                Country: US

          Personal name: Melanie H Hart
     Organization units: SALES
         Private domain: TANNER GROUP
  Administrative domain: ATTMAIL
                Country: US
```

DISPLAY_DELIVERY_OPTIONS FORL, WRIL, and WRIR Subcommand

Purpose Displays the delivery options of the letter in progress.

Format

DISPLAY_DELIVERY_OPTIONS or
DISDO or
DISPLAY_ATTRIBUTE or
DISPLAY_ATTRIBUTES or
DISPLAY_DELIVERY_OPTION or
DISA
OUTPUT=file
STATUS=status variable

Parameters *OUTPUT* or *O*

Specifies the name of the file to which the output is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks The delivery options you can specify for a letter are:

IMPORTANCE
SENSITIVITY
DELIVERY_PRIORITY
CONVERSION_PROHIBITED

RETURN_CONTENTS
ALTERNATE_RECIPIENT_ALLOWED
DISCLOSE_OTHER_RECIPIENTS
DELIVERY_CONFIRMATION

You set these options on the SET_DELIVERY_OPTIONS subcommand. See the description of that subcommand for details on each option.

For more information, see the Mail/VE Version 2 Usage manual.

Examples

The following example displays the delivery options of a letter sent from the mailbox with personal name Sarah Lofquist.

```
Mail/write_letter
Wril/set_subject 'Question about schedule.'
Wril/collect_text $local.note
ct?  Please clarify the milestone labeled
ct?  design review.  Does the corresponding
ct?  date indicate completion?
ct?**
Wril/add_letter_part lp=$local.note
Wril/add_to ..
Wril../personal_name='Jean Beinert'
Wril/set_delivery_options
Wril../delivery_confirmation=delivery
Wril/display_delivery_options
 Message id: 580B090E2A31004A-PINK
 Personal name: Sarah Lofquist
 Organization units: SYSTEM A
 Importance: NORMAL IMPORTANCE
 Sensitivity: NOT SENSITIVE
 Delivery priority: NORMAL PRIORITY
 Conversion prohibited: FALSE
 Content return requested: FALSE
 Alternate recipient allowed: FALSE
 Disclose other recipients: FALSE
 Delivery confirmation: DELIVERY
```

DISPLAY_DISTRIBUTION_LISTS MAIL Subcommand

Purpose Displays information about one or more of your distribution lists.

Format

DISPLAY_DISTRIBUTION_LISTS or
DISPLAY_DISTRIBUTION_LIST or
DISDL
DISTRIBUTION_LIST_NAME=**keyword** or **string** or **record**
ORGANIZATION_UNITS=*list of string*
ORGANIZATION_NAME=*string*
DISPLAY_OPTIONS=*keyword*
OUTPUT=*file*
STATUS=*status variable*

Parameters

DISTRIBUTION_LIST_NAME or **DLN**

Specifies the name of the distribution list to be displayed. Specify the keyword ALL to display information about all of the distribution lists you own. Otherwise, enter the distribution list name as a 1- to 64-character string or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need further information on specifying a distribution list name.

This parameter is required.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the distribution list to be displayed. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the distribution list to be displayed. Specify a string of 1 to 64 characters.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Parameter Attributes: BY_NAME

Specifies the information to be displayed. Options are:

BRIEF or B

Displays the local address attributes of the distribution list, the distribution list type, and the expand attribute.

FULL or F

Displays all distribution list attributes.

PERMITS or PERMIT or P

Displays the distribution list type and, if it is a group distribution list, the permission list entries.

CONTENTS or C

Displays the addresses contained in the distribution list.

If omitted, BRIEF is used.

OUTPUT or *O*

Parameter Attributes: BY_NAME

Specifies the file to which the display is written. If omitted, the display is written to the output file you specified on the EMAIL command or, by default, to your terminal screen.

Remarks

- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined names/units.
- To list the contents of distribution lists you do not own but are permitted to display (that is, to which you have READ permission), use the EXPAND_DISTRIBUTION_LIST subcommand described in the Mail/VE Commands, Subcommands, and Functions chapter of the Mail/VE Usage manual.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples

This example requests a display of the contents of a distribution list:

```
Mail/display_distribution_list ..
Mail../dln='process_review_task_force' ..
Mail../display_options=contents
   1.          Personal name: Tyler Ross
         Organization units: EMAIL PROJECT
                             SYSTEMS DEPARTMENT
                             PROGRAMMING DIVISION
          Organization name: MICRO SOLUTIONS
   2.          Personal name: Mary Landon
         Organization units: NETWORK CONTROL
                             SYSTEMS DEPARTMENT
                             PROGRAMMING DIVISION
          Organization name: MICRO SOLUTIONS
   3.          Personal name: Harry Jenson
         Organization units: AI TOOLS
                             SYSTEMS DEPARTMENT
                             PROGRAMMING DIVISION
          Organization name: MICRO SOLUTIONS
   4.          Personal name: Rene Walters
         Organization units: AI TOOLS
                             SYSTEMS DEPARTMENT
                             PROGRAMMING DIVISION
          Organization name: MICRO SOLUTIONS
```

The order of organization units in the example is from least inclusive to most inclusive. For example, Tyler Ross is on the EMAIL Project in the Systems Department in the Programming Division.

DISPLAY_DOMAINS
MAIL Subcommand

Purpose Displays the private and administrative domains and countries registered in the mail directory.

Format

DISPLAY_DOMAINS or
DISPLAY_DOMAIN or
DISD
OUTPUT=file
STATUS=status variable

Parameters *OUTPUT* or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file to which the output is written. If omitted, output is written to the output file you specified on the EMAIL command or, by default, to your terminal screen.

Remarks

- Use the list of domains as a reference in determining active interdomain routes for sending mail.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example displays the domain information for a mail system:

```
Mail/display_domain
Country: US
Administrative domain: ATTMAIL
```

DISPLAY_LETTER_PARTS FORL, WRIL, and WRIR Subcommand

Purpose Displays the list of parts that make up the letter body.

Format

DISPLAY_LETTER_PARTS or
DISPLAY_LETTER_PART or
DISLP
OUTPUT=file
STATUS=status variable

Parameters *OUTPUT* or *O*

Specifies the name of the file to which the display is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks

- If the letter file list is empty, the command terminates with an informative message.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example shows the format of the display:

```
Wril/display_letter_parts
 :$LOCAL.NOTE (TEXT)
 :NVE.JMW.EXAMPLES.P0533 (TEXT)
 :NVE.JMW.EXAMPLES.P0531 (TEXT)
```


DISPLAY_RECIPIENT_OPTIONS FORL, WRIL, and WRIR Subcommand

Purpose Displays the current value of the recipient options.

Format

DISPLAY_RECIPIENT_OPTIONS or
DISPLAY_RECIPIENT_OPTION or
DISRO
OUTPUT=file
STATUS=status variable

Parameters *OUTPUT* or *O*

Specifies the name of the file to which the display is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks The recipient options you can specify when composing a letter are:

```
CERTIFY_RECEIPT
CERTIFY_NONRECEIPT
REPLY_REQUESTED
```

You set these options on the SET_RECIPIENT_OPTIONS subcommand. See the description of that subcommand for details on each option.

For more information, see the Mail/VE Version 2 Usage manual.

DISPLAY_MAILBOXES MAIL Subcommand

Purpose Displays information about any mailbox you own.

Format

DISPLAY_MAILBOXES or
DISPLAY_MAILBOX or
DISM
PERSONAL_NAME=keyword or *string* or *record*
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
DISPLAY_OPTIONS=keyword
OUTPUT=file
STATUS=status variable

Parameters *PERSONAL_NAME* or *PN*

Specifies the name of the mailbox to be displayed. Specify the keyword ALL to display information about all of the mailboxes you own. Otherwise, enter the mailbox name as a 1- to 64-character string or a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need more information on specifying mailbox names.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the mailbox to be displayed. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the mailbox to be displayed. Enter a string of 1 to 64 characters.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Parameter Attributes: BY_NAME

Specifies the information to be displayed. Options:

BRIEF or B

Displays the local address attributes of the mailbox, the letter retention period, and the mailbox type.

FULL or F

Displays all mailbox address attributes.

PERMITS, PERMIT, or P

Displays the mailbox type and, if it is a group mailbox, the permission list entries.

If DISPLAY_OPTIONS is omitted, BRIEF is used.

OUTPUT or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file to which the display is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks

- The wildcard character * is not permitted in mailbox addresses.
- The addressing parameters are PERSONAL_NAME, ORGANIZATION_UNITS, and ORGANIZATION_NAME.
  - If you do not specify any of the addressing parameters, your identity mailbox is displayed.
  - If the addressing parameters do not uniquely specify a mailbox you own, the command terminates with an error message.
- You can use the mailbox alias name(s) to reference a mailbox.
- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples

The following example specifies that the display include the list of permissions that define a group mailbox:

```
Mail/display_mailbox 'Project_Notes' do=permits
   1. Personal name: Mincher, Alex
      Permissions: WRITE READ

   2. Organization name: RESEARCH GROUP
      Permissions: WRITE READ
```

DISPLAY_ORGANIZATIONS
MAIL Subcommand

Purpose Displays the organization names and units registered in the mail directory.

Format

DISPLAY_ORGANIZATIONS or
DISPLAY_ORGANIZATION or
DISO
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
DISPLAY_OPTIONS=keyword
OUTPUT=file
STATUS=status variable

Parameters

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) used to determine the display content. The display will include only those registered organization names/units that include the value specified. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name used to determine display content. Enter a string of 1 to 64 characters.

The display will include only those registered organization names/units that include the value specified.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Parameter Attributes: BY_NAME

Specifies the display content. Options are:

LOCAL or L

Lists registered organizations residing on this host.

NON_LOCAL or NL

Lists registered organizations not on this host.

ALL

Lists local and nonlocal organizations.

If omitted, LOCAL is used.

OUTPUT or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file to which the output is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks

- Organizations are defined by the Mail/VE administrator. Entries are defined using units (one to four), a name, or both.
- The display lists organization unit/name combinations that are valid for mailbox definition or mail transfer.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples

The following example displays the organizations registered on the local host for local mailbox definition and local mail transfer:

```
Mail/display_organizations do=local
Organization units: PROGRAMMING

Organization units: ACCOUNTING

Organization units: PURCHASING
```

DISPLAY_PERMISSIONS
CHADL, CHAM, CREDL, and CREM Subcommand

Purpose Displays the list of permitted mailboxes for a group mailbox or distribution list.

Format

DISPLAY_PERMISSIONS or
DISPLAY_PERMISSION or
DISP
DISPLAY_OPTIONS=keyword
OUTPUT=file
STATUS=status variable

Parameters

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Specifies what information is displayed. Options:

BRIEF or B

Displays the address attributes for each permission entry.

FULL or F

Displays all of the address attributes and permissions granted.

If omitted, BRIEF is used.

OUTPUT or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file to which the display is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example produces a brief display of the permitted mailbox list:

```
Mail/change_mailbox 'mvedev'
Cham/display_permissions
    1. Personal name: Akers, Ben
       Organization units: SALES

    2. Personal name: Post, Linda
       Organization units: SALES

    3. Organization name: MARKETING
```

The following example produces a full display of the permitted mailbox list:

```
Mail/change_mailbox 'mvedev'
Cham/display_permissions do=full
    1. Personal name: Akers, Ben
       Organization units: SALES
       Permissions: READ WRITE

    2. Personal name: Post, Linda
       Organization units: SALES
       Permissions: READ WRITE

    3. Organization name: MARKETING
       Permissions: READ WRITE
```


DISPLAY_PERMISSIONS CHAM and CREM Subcommand

Purpose Displays the list of permitted mailboxes for a group mailbox or distribution list.

Format

DISPLAY_PERMISSIONS or
DISPLAY_PERMISSION or
DISP
DISPLAY_OPTIONS=keyword
OUTPUT=file
STATUS=status variable

Parameters *DISPLAY_OPTIONS* or *DISPLAY_OPTION* or *DO*

Specifies what information is displayed. Options:

BRIEF or B

Displays the address attributes for each permission entry.

FULL or F

Displays all of the address attributes and permissions granted.

If omitted, BRIEF is used.

OUTPUT or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file to which the display is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks For more information, see the Mail/VE Version 2 Usage manual.

DISPLAY_SUBJECT FORL, WRIL, and WRIR Subcommand

Purpose Displays the subject of the letter being composed.

Format

DISPLAY_SUBJECT or
DISS
OUTPUT=file
STATUS=status variable

Parameters *OUTPUT* or *O*

Specifies the name of the file to which the output is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example displays the subject of the letter currently being composed:

```
Wril/display_subject
 Subject: Question about schedule.
```

DISPLAY_TO
FORL, WRIL, and WRIR Subcommand

Purpose Displays the primary (To) address list.

Format

DISPLAY_TO or
DIST
DISPLAY_OPTIONS=keyword
OUTPUT=file
STATUS=status variable

Parameters *DISPLAY_OPTIONS* or *DISPLAY_OPTION* or *DO*

Specifies the display content. Options:

BRIEF or B

Displays local address attributes.

FULL or F

Displays address attributes and recipient options for each recipient.

If omitted, BRIEF is used.

OUTPUT or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file to which the output is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks

- If there are no addresses in the To address list, the command terminates with an informative message.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example produces the brief version of the address list display.

```
Wril/display_to
          Personal name: Karen McDonald
     Organization units: PRODUCT RESEARCH
      Organization name: MARKETING

          Personal name: Jerry Pfeiffer
     Organization units: US SALES
                         MIDWEST DISTRICT
      Organization name: MARKETING

          Personal name: Michael Carpenter
     Organization units: US SALES
                         NORTHEAST DISTRICT
      Organization name: MARKETING

          Personal name: Nancy A. Reed
     Organization units: US SALES
                         NORTHEAST DISTRICT
      Organization name: MARKETING
```

The following example produces the full version of the address list display.

```
Wril/display_to do=full
          Personal name: Karen McDonald
     Organization units: PRODUCT RESEARCH
      Organization name: MARKETING
         Private domain: CARTER ELECTRONICS
  Administrative domain: ATTMAIL
                Country: US

          Personal name: Jerry Pfeiffer
     Organization units: US SALES
                         MIDWEST DISTRICT
      Organization name: MARKETING
         Private domain: CARTER ELECTRONICS
  Administrative domain: ATTMAIL
                Country: US

          Personal name: Michael Carpenter
     Organization units: US SALES
                         NORTHEAST DISTRICT
      Organization name: MARKETING
         Private domain: CARTER ELECTRONICS
  Administrative domain: ATTMAIL
                Country: US

          Personal name: Nancy A. Reed
     Organization units: US SALES
                         NORTHEAST DISTRICT
      Organization name: MARKETING
         Private domain: CARTER ELECTRONICS
  Administrative domain: ATTMAIL
                Country: US
```


$DISTRIBUTION_LIST_ATTRIBUTE MAIL Function

Purpose Returns the value of a specified distribution list attribute.

Format

```
$DISTRIBUTION_LIST_ATTRIBUTE
   (ADDRESS: string or record
   OPTION: keyword )
```

Parameters

ADDRESS

Specifies a distribution list address.

You must own the distribution list you specify or a null string will be returned when you use this function.

Only the distribution list name is required. Specify the organization unit(s) and organization name associated with the distribution list to further define the address. Enter the address as a string or as a record in the format:

```
record
  personal_name: record
    surname: string 1..40
    given_name: string 0..16
    initials: string 0..5
    generation_qualifier: string 0..3
  recend
  organization_units: list 0..4 of string 0..32
  organization_name: string 0..64
recend
```

See chapter 2, Mail/VE Addresses, in the Mail/VE Version 2 Usage manual if you need more information on specifying an address.

This parameter is required.

OPTION

Specifies the attribute whose value is to be returned. Keyword options:

ADDRESS or A

Returns the local address record.

ADDRESS_LIST or AL

Returns the list of addresses that make up the distribution list. The value is returned as a list of global address records.

ADDRESS_LIST_SIZE or ALS

Returns an integer specifying the number of entries in the distribution list.

EXPAND or E

Returns a boolean value that indicates whether the distribution list will be expanded in the letter header.

PERMISSION_LIST or PL

Returns the permission list entries that define a group distribution list. The value is returned as a list of global address records.

PERMISSION_LIST_SIZE or PLS

Returns an integer specifying the number of entries in the permission list for a group distribution list.

PERMIT_TYPE or PT

Returns a string containing PRIVATE, PUBLIC, or GROUP.

DESCRIPTION or D

Returns a string of 1 to 64 characters that describes the distribution list.

TELEPHONE_NUMBER or TN

Returns a string of up to 32 characters that contains the telephone number associated with the distribution list.

This parameter is required.

Remarks

- Distribution list attributes that have not been defined return a null value.
- This function returns address records in the following formats.

  Local address records:

```
record
  personal_name: record
    surname: string 1..40
    given_name: string 0..16
    initials: string 0..5
    generation_qualifier: string 0..3
  recend
  organization_units: list 0..4 of string 0..32
  organization_name: string 0..64
recend
```

  Global address records:

```
record
  personal_name: record
    surname: string 1..40
    given_name: string 0..16
    initials: string 0..5
    generation_qualifier: string 0..3
  recend
  organization_units: list 0..4 of string 0..32
  organization_name: string 0..64
  private_domain: string 0..16
  administrative_domain: string 0..16
  country: string 0..3
  x121_address: string 0..15
  terminal_identifier: string 0..24
  ua_identifier: string 0..32
  domain_defined_attributes: list 0..4
    record
      name: string 1..8
      value: string 1..128
    recend
recend
```

- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example uses the $DISTRIBUTION_LIST_ATTRIBUTE function to test the size of a distribution list, which determines whether or not the list is displayed:

```
Mail/if $distribution_list_attribute('mrdl' als) < 5 then
if/display_distribution_list 'mrdl'
if/ifend
```


END_CHANGE_DISTRIBUTION_LIST CHADL Subcommand

Purpose Ends a CHANGE_DISTRIBUTION_LIST subutility session.

Format **END_CHANGE_DISTRIBUTION_LIST** or
ENDCDL or
QUI or
QUIT or
END
SAVE=boolean
STATUS=status variable

Parameters *SAVE* or *S*

Specifies whether or not the changed distribution list definition should be saved. Options:

TRUE

Saves the changed distribution list definition.

FALSE

Makes no changes to the existing distribution list definition.

If SAVE is omitted, TRUE is used.

Remarks For more information, see the Mail/VE Version 2 Usage manual.

END_CHANGE_MAILBOX CHAM Subcommand

Purpose Ends a CHANGE_MAILBOX subutility session.

Format

END_CHANGE_MAILBOX or
ENDCM or
QUI or
QUIT or
END
SAVE=boolean
STATUS=status variable

Parameters *SAVE* or *S*

Specifies whether or not the changed mailbox definition will be saved. Options:

TRUE

Saves the changed mailbox definition.

FALSE

Makes no changes to the existing mailbox definition.

If SAVE is omitted, TRUE is used.

Remarks For more information, see the Mail/VE Version 2 Usage manual.

END_CREATE_DISTRIBUTION_LIST CREDL Subcommand

Purpose Ends a CREATE_DISTRIBUTION_LIST subutility session.

Format

END_CREATE_DISTRIBUTION_LIST or
ENDCDL or
QUI or
QUIT or
END
SAVE=boolean
STATUS=status variable

Parameters *SAVE* or *S*

Specifies whether or not the new distribution list definition will be saved. Options:

TRUE

Saves the new distribution list.

FALSE

The new distribution list is not saved.

If SAVE is omitted, TRUE is used.

Remarks For more information, see the Mail/VE Version 2 Usage manual.

END_CREATE_MAILBOX CREM Subcommand

Purpose Ends a CREATE_MAILBOX subutility session.

Format

END_CREATE_MAILBOX or
ENDCM or
QUI or
QUIT or
END
SAVE=boolean
STATUS=status variable

Parameters *SAVE* or *S*

Specifies whether or not the new distribution list definition will be saved.

Options:

TRUE

Saves the mailbox definition.

FALSE

Does not save the mailbox definition.

If SAVE is omitted, TRUE is used.

Remarks For more information, see the Mail/VE Version 2 Usage manual.

END_EMAIL
MAIL Subcommand

Purpose Terminates a Mail/VE session.

Format

END_EMAIL or
ENDE or
QUI or
QUIT or
END
STATUS=status variable

Remarks

- Each subutility includes an END subcommand that terminates the subutility and returns you to the previous command level.

  To terminate a Mail/VE session, enter the END subcommand following the `Mail/` prompt.

- If you enter END_EMAIL or ENDE from within a subutility, you terminate the mail session.

- For more information, see the Mail/VE Version 2 Usage manual.

END_FORWARD_LETTER FORL Subcommand

Purpose Ends the FORWARD_LETTER subutility session.

Format

END_FORWARD_LETTER or
ENDFL or
QUI or
QUIT or
END
SEND=boolean
STATUS=status variable

Parameters *SEND* or *S*

Specifies whether the letter being composed will be forwarded. Options:

TRUE

The letter will be forwarded.

FALSE

The letter in progress will not be forwarded.

If SEND is omitted, TRUE is used.

Remarks For more information, see the Mail/VE Version 2 Usage manual.

END_WRITE_LETTER WRIL Subcommand

Purpose Ends the WRITE_LETTER subutility session.

Format

END_WRITE_LETTER or
ENDWL or
QUI or
QUIT or
END
SEND=boolean
STATUS=status variable

Parameters *SEND* or *S*

Specifies whether the letter being composed will be sent. Options:

TRUE

The letter being composed will be sent.

FALSE

The letter being composed will not be sent.

If SEND is omitted, TRUE is used.

Remarks For more information, see the Mail/VE Version 2 Usage manual.

END_WRITE_REPLY WRIR Subcommand

Purpose Ends the WRITE_REPLY subutility session.

Format

END_WRITE_REPLY or
ENDWR or
QUI or
QUIT or
END
SEND=boolean
STATUS=status variable

Parameters *SEND* or *S*

Specifies whether the reply in progress will be sent. Options:

TRUE

The reply letter will be sent.

FALSE

The reply letter will not be sent.

If SEND is omitted, TRUE is used.

Remarks For more information, see the Mail/VE Version 2 Usage manual.

EXPAND_DISTRIBUTION_LIST MAIL Subcommand

Purpose Displays the list of addresses in a distribution list and the content of any distribution lists contained within that list.

Format

EXPAND_DISTRIBUTION_LIST or
EXPDL
DISTRIBUTION_LIST_NAME=string or **record**
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
OUTPUT=file
STATUS=status variable

Parameters

DISTRIBUTION_LIST_NAME or **DLN**

Specifies the name of the distribution list to be expanded. Enter the name as a 1- to 64-character string or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual if you need details on specifying a distribution list name.

This parameter is required.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the distribution list to be expanded. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the distribution list to be expanded. Enter a string of 1 to 64 characters.

OUTPUT or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file on which the display is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks

- Distribution lists are expanded only if the user has READ permission to the list.
- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example expands a distribution list to show its contents:

```
Mail/expand_distribution_list ..
Mail../distribution_list_name='process_task_force'
    Personal name: Tyler Ross
Organization unit: EMAIL PROJECT
                   SYSTEMS DEPARTMENT
                   PROGRAMMING DIVISION
Organization name: MICRO SOLUTIONS

    Personal name: Mary Landon
Organization unit: NETWORK CONTROL
                   SYSTEMS DEPARTMENT
                   PROGRAMMING DIVISION
Organization name: MICRO SOLUTIONS

    Personal name: Harry Jenson
Organization unit: AI TOOLS
                   SYSTEMS DEPARTMENT
                   PROGRAMMING DIVISION
Organization name: MICRO SOLUTIONS
```


FORWARD_LETTER
MAIL Subcommand

Purpose Initiates the FORWARD_LETTER subutility, through which you can forward a letter including optional comments to other addresses. When you enter FORWARD_ LETTER, the system displays the prompt `For1/`.

Format **FORWARD_LETTER** or
FORL
LETTER=keyword or *integer* or *string*
PROLOG=file
STATUS=status variable

Parameters *LETTER* or *L*

Identifies the letter to be forwarded. Specify a letter number, letter identifier, or a keyword.

The letter number is the number assigned to the letter on the letter list. It can change each time you enter the LIST_MAIL subcommand. The letter identifier is permanently assigned to the letter by Mail/VE.

Keyword options:

CURRENT or C

Selects the current letter in the letter list. (The current letter is the letter most recently displayed.)

FIRST or F

Selects the first letter in the letter list.

LAST or L

Selects the last letter in the letter list.

NEXT or N

Selects the letter following the current letter in the letter list.

PREVIOUS or P

Selects the letter preceding the current letter in the letter list.

If LETTER is omitted, CURRENT is used.

PROLOG or *P*

Parameter Attributes: BY_NAME

Specifies the name of the file from which commands are read before giving control to the FORWARD_LETTER subutility. If omitted, $NULL is used.

The $NULL default can be overridden by creating an SCL variable called MVD$MAIL_WRITE_PROLOG and setting its value to the name of the prolog file; specify this value as a string.

Remarks

- Use the FORWARD_LETTER subutility subcommands, listed next, to add comments, specify letter attributes, define the address list, and forward the letter.

  ADD_BLIND_COURTESY_COPY
  ADD_COURTESY_COPY
  ADD_LETTER_PARTS
  ADD_TO
  DELETE_BLIND_COURTESY_COPY
  DELETE_COURTESY_COPY
  DELETE_LETTER_PARTS
  DELETE_TO
  DISPLAY_BLIND_COURTESY_COPY
  DISPLAY_COURTESY_COPY
  DISPLAY_DELIVERY_OPTIONS
  DISPLAY_LETTER_PARTS
  DISPLAY_RECIPIENT_OPTIONS
  DISPLAY_SUBJECT
  DISPLAY_TO
  END_FORWARD_LETTER
  SET_DELIVERY_OPTIONS
  SET_RECIPIENT_OPTIONS
  SET_SUBJECT

  These subcommands are described in the Letter Posting Subutility Subcommands section of the Mail/VE Version 2 Usage manual.

- If you list an address more than once in the address list, only one copy of the letter is sent to the address.

- The address in the From field is set to the value returned by the $IDENTITY function.

- If an address is repeated, only one copy of the letter is sent to the address. This is true regardless of the form of the address or whether the address is included in more than one distribution list.

- Receipts are generated by the system when certified letters are read, copied, or printed.

- Notices are generated by the system, unless suppressed by the sender, if delivery confirmation is requested or when the mail system determines the letter cannot be delivered to an address.

- Nondelivery notices are not generated for incorrect local addresses in a distribution list or for any addresses in a distribution list whose EXPAND attribute is set to FALSE. Local addresses are validated when the distribution list name is entered in the recipient list and incorrect addresses are simply ignored.

  However, nondelivery notices are generated for incorrect nonlocal addresses in a local distribution list whose EXPAND attribute is set to TRUE. These addresses are not validated when they are added to the distribution list; they are validated when Mail/VE attempts to deliver mail to them.

- For more information, see the Mail/VE Version 2 Usage manual.

Examples This example forwards the current letter with comments to another mailbox:

```
Mail/forward_letter
For1/add_to 'Glen Johnson'
For1/set_subject 'Request for Comments'
For1/collect_text $local.notes
Colt/Please review the attached document
Colt/by Wednesday and return comments
Colt/to Art Pehler.
Colt/**
For1/add_letter_part $local.notes
For1/end_forward_letter
```


$IDENTITY
MAIL Function

Purpose Returns a record that contains the address by which you are known to the mail system.

Format $IDENTITY

Parameters None.

Remarks

- The $IDENTITY function returns a local address record in the format:

```
record
  personal_name: record
    surname: string 1..40
    given_name: string 0..16
    initials: string 0..5
    generation_qualifier: string 0..3
  recend
  organization_units: list of 0..4 of string 0..32
  organization_name: string 0..64
recend
```

- The address of the default mailbox is automatically used as your identity when you enter the mail system. Identity is used within Mail/VE:
  - As the From address for any WRITE_LETTER, WRITE_REPLY, or FORWARD_ LETTER subcommand you enter.
  - To determine which mailboxes you can access for reading mail, which mailboxes can be delivered mail, and which distribution lists are available.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples

The following example uses the NOS/VE command DISPLAY_VALUE to display $IDENTITY. It shows Ken Nordquist as the personal name and purchasing as the organization unit.

```
Mail/display_value $identity
Nordquist
Ken

PURCHASING

Mail/
```

$LETTER
MAIL Function

Purpose Returns a string containing the letter identifier of the current letter (that is, the most recently displayed letter).

Format **$LETTER**

Parameters None.

Remarks

- Use LIST_MAIL to establish a letter list in the mailbox before you use the $LETTER function. If a letter list does not exist, a null string is returned.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples This example uses the NOS/VE command DISPLAY_VALUE to display $LETTER:

```
Mail/display_value $letter
JA27
```

$LETTER_ATTRIBUTE
MAIL Function

Purpose Returns the value of a selected letter attribute.

Format **$LETTER_ATTRIBUTE**
(OPTION: keyword
LETTER: keyword or *integer* or *string*)

Parameters **OPTION**

The attribute whose value is to be returned. Options:

SEEN

Returns a boolean value specifying whether the letter has been displayed. Applies only to owned mailboxes.

SUBJECT

Returns the subject of the letter in a string of up to 128 characters.

ORIGINATOR

Returns the sender's address in a global address record.

MESSAGE_IDENTIFIER

Returns the X.400 message identifier in a string of 0 to 64 characters.

IMPORTANCE

Returns a string specifying the sender's rating of the the importance of the letter as NORMAL, LOW, or HIGH.

SENSITIVITY

Returns a string specifying the sender's rating of the sensitivity of the letter as PERSONAL, PRIVATE, or CONFIDENTIAL.

REPLY_REQUEST

Returns the boolean value TRUE if the sender requested a reply; otherwise, returns FALSE.

DELIVERY_PRIORITY

Returns a string that specifies the sender's rating of the urgency of the letter as URGENT or NON_URGENT.

CERTIFIED

Returns the boolean value TRUE if the sender requested certified delivery; otherwise, returns FALSE.

DATE_TIME_SENT

Returns a record containing the date and time the letter was sent.

DATE_TIME_RECEIVED

Returns a record containing the date and time the letter was received.

DATE_TIME_EXPIRES

Returns a record containing the date and time that the sender specifies as the expiration of the letter.

RETAIN_UNTIL

Returns the date and time until which the letter is retained in the system.

MESSAGE_SIZE

Returns the size of the message (in bytes) as an integer.

AUTO_FORWARDED

Returns the boolean value TRUE if the letter was automatically forwarded from another mailbox; otherwise, returns FALSE.

NUMBER_OF_LETTER_PARTS

Returns an integer specifying the number of parts in the letter body. The number can be zero if the subject is the entire letter body.

LETTER_PARTS

Returns a list of one or more strings specifying the type(s) of information contained in the message.

MESSAGE_TYPE

Returns a string containing LETTER or RECEIPT.

LETTER

Specifies the letter for which an attribute is returned. Specify the letter by letter number (integer), identifier (a string of 1 to 6 characters), or keyword.

The letter number is the number from the previous letter list. It can change each time you enter the LIST_MAIL subcommand.

The letter identifier is permanently assigned to the letter and is not related to previous execution of any command.

Keyword options:

FIRST or F

Displays an attribute value from the first letter in the list.

NEXT or N

Displays an attribute value from the next letter in the list.

PREVIOUS or P

Displays an attribute value from the previous letter in the list.

LAST or L

Displays an attribute value from the last letter in the list.

CURRENT or C

Displays a value from the current letter in the letter list. (The current letter is the letter most recently displayed.)

If omitted, $LETTER is used.

Remarks

- If a letter list does not exist, a null string is returned.
- The $LETTER function returns the value of the current letter from the letter list (that is, the letter most recently displayed).
- For more information, see the Mail/VE Version 2 Usage manual.

Examples

In the following example the subject of the current letter is displayed:

```
Mail/letter_subject=$letter_attribute(subject)
Mail/display_value letter_subject
Reply to   How about lunch on Monday?
```


$LETTER_LIST
MAIL Function

Purpose

Returns the letter list as a list of strings containing letter identifiers.

Format

$LETTER_LIST

Parameters

None.

Remarks

- Use LIST_MAIL to establish a letter list in the mailbox before using the $LETTER_LIST function. If a letter list does not exist, an empty list is returned.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples

The following example searches a letter list for a specific date to begin reading.

```
Mail/date_to_find=$now
Mail/if date_to_find.month > 1 then
if/date_to_find.month = date_to_find.month -1
if/else
if/date_to_find.month = 12
if/ifend
Mail/list_mail 'newswire' output = $null
Mail/for each ltr in $letter_list do
for/date = $letter_attribute(..
for../date_time_received ltr)
for/if date.month = date_to_find.month then
if/read_letter ltr o=new_letters.$eoi
if/ifend
for/forend
Mail/
```

LIST_MAIL MAIL Subcommand

Purpose

Displays the number of letters in one or more mailboxes, and a numbered list of letters for the first mailbox in the list. Displays the letters from a specified mailbox. The letters are shown in a numbered list.

Format

LIST_MAIL or
LISM or
LIST or
READ_MAIL or
REAM
PERSONAL_NAME=string or *record* or *keyword*
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
MAIL_OPTIONS=keyword or *list of keyword*
DISPLAY_OPTIONS=keyword
OUTPUT=file
STATUS=status variable

Parameters *PERSONAL_NAME* or *PN*

Specifies the name of the mailbox for which you want to display a letter list. Enter the name as the keyword OWN or as a 1- to 64-character string.

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual a mailbox name.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the names of from one to four organization unit(s) defined in the mailbox address. Enter each unit as a 1- to 32-character string.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the name of the organization associated with the mailbox address. Enter the name as a string of 1 to 64 characters.

MAIL_OPTIONS or *MAIL_OPTION* or *MO*

Specifies the type(s) of letters to be included in the letter list. You can specify ALL or one or more keyword options for each type.

If you select more than one option for a type, all of your selections apply. If you select all of the options for any type, the effect is the same as if you had selected none of the options for that type.

Letter types:

If you do not select any letter type options, all letters are listed.

LETTER or L

Includes letters.

RECEIPT or R

Includes letters generated upon receipt or nonreceipt of a letter.

Read Status types:

These options apply only to owned mailboxes.

SEEN or S

Includes letters that were previously displayed using READ_LETTER.

UNSEEN or US

Includes letters that have not been displayed.

Sensitivity options:

PRIVATE or PR

Includes letters that the sender classified as private.

PERSONAL or PE

Includes letters that the sender classified as personal.

CONFIDENTIAL or CO

Includes letters that the sender classified as confidential.

Receipt Certification types:

CERTIFIED or C

Includes letters that the sender classified as certified.

UNCERTIFIED or UC

Includes letters that are not classified as certified.

Importance types:

LOW_IMPORTANCE or LI

Includes letters that the sender classified low importance.

NORMAL_IMPORTANCE or NI

Includes letters that the sender classified normal importance.

HIGH_IMPORTANCE or HI

Includes letters that the sender classified high importance.

Delivery Priority types:

NORMAL_PRIORITY or NP

Includes letters that the sender classified for normal delivery.

URGENT_PRIORITY or UP

Includes letters that the sender classified for urgent delivery.

NON_URGENT_PRIORITY or NUP

Includes letters that the sender classified for nonurgent delivery.

Other types:

REPLY_REQUEST or RR

Includes letters for which the sender requested a reply.

AUTOFORWARDED or A

Includes letters that were autoforwarded to the mailbox for which the letter list is being produced.

If you do not specify any options and you own the mailbox, UNSEEN is used.

If the mailbox is not owned by the caller of the LIST_MAIL subcommand, ALL is used.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Parameter Attributes: BY_NAME

Specifies the amount of information that appears in the letter list display. Options:

BRIEF or B

Displays a one-line description of each letter that specifies the date/time received, the sender's mailbox name (truncated when necessary), the subject (truncated when necessary), the length in bytes, and a flag if the letter has high importance or a reply is requested.

FULL or F

Displays a multiline description of each letter that specifies the letter identifier, the date/time received, the sender's mailbox name and organization, the subject, the length of the letter in bytes, and a flag if the letter has high importance or a reply is requested.

If DISPLAY_OPTIONS is omitted, BRIEF is used.

OUTPUT or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file to which the display is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks

- If you do not specify any of the addressing parameters (PERSONAL_NAME, ORGANIZATION_UNITS, or ORGANIZATION_NAME), all of your mailboxes are checked for mail.
- The LIST_MAIL subcommand produces an ordered list of letters that satisfies the options you specify on the MAIL_OPTIONS parameter. You can then use the letter number or letter identifier from this list in subsequent READ_LETTER subcommands.
- The mailbox whose letters you want to list must be one of the mailboxes you own or are permitted to read.
- If the mailbox specified does not contain any mail that meets the mail options criteria, the letter list is empty. An informative message is displayed.
- The $MAILBOX function returns the local address of the mailbox for which a letter list is created. If the letter list is empty, $MAILBOX returns a null record.
- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example checks a mailbox for unseen mail:

```
Mail/list_mail personal_name='Tyler Hanson'
-- The letter selection criteria found no letters to display.
```

The following example displays a brief listing of receipts in a mailbox:

```
Mail/list_mail pn='Tyler Hanson' mo=receipt do=brief
 Hanson, Tyler contains the following letters:
    1. May 20, 1989.8:23 PM Bradley, Jan  RECEIPT 219
```

LOOKUP_ADDRESS
MAIL Subcommand

Purpose Displays information about one or more addresses.

Format

LOOKUP_ADDRESS or
LOOA
PERSONAL_NAME=string or *record*
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
ADDRESS_OPTIONS=keyword or *list of keyword*
DISPLAY_OPTIONS=keyword or *list of keyword*
OUTPUT=file
STATUS=status variable

Parameters *PERSONAL_NAME* or *DISTRIBUTION_LIST_NAME* or *PN* or *DLN*

Specifies the mailbox or distribution list name of the address for which you are searching. Enter the name as a string of 1 to 64 characters or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual for details on specifying a mailbox or distribution list name.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the address(es) for which you are searching. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the address(es) for which you are searching. Enter a string of 1 to 64 characters.

ADDRESS_OPTIONS or *ADDRESS_OPTION* or *AO*

Parameter Attributes: BY_NAME

Specifies the criteria used to search for the address. Specify ALL or one or more of the following keyword options. ALL means the address search is not limited by address type (that is, mailbox versus distribution list) or by permit type (that is, private, group, or public).

MAILBOX or M

Limits the search to mailbox addresses.

DISTRIBUTION_LIST or DL

Limits the search to distribution list addresses.

PRIVATE or PR

Limits the search to private addresses.

GROUP or G

Limits the search to group addresses.

PUBLIC or PU

Limits the search to public addresses.

If ADDRESS_OPTIONS is omitted, ALL is used.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Parameter Attributes: BY_NAME

Specifies the information to include in the address display. Specify ALL or one or more of the following keyword options:

PERSONAL_NAME or PN

Displays the primary mailbox or distribution list name for each address.

ALIAS_NAME or AN

Displays the alias name or names of the mailbox or distribution list, if any, for each address.

ORGANIZATION_UNITS, ORGANIZATION_UNIT, or OU

Displays the organization units, if any, for each address.

ORGANIZATION_NAME or ON

Displays the organization name, if any, for each address.

TELEPHONE_NUMBER or TN

Displays the telephone number, if any, for each address.

DESCRIPTION or D

Displays the description, if any, for each address.

If omitted, PERSONAL_NAME is used.

OUTPUT or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file on which the display is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks

- If your Mail/VE administrator has established a domain-wide directory, you can look up addresses on remote hosts as well as on your own host.
- You need to specify at least one of the addressing parameters (PERSONAL_NAME, ORGANIZATION_UNITS, or ORGANIZATION_NAME).

- You can use the wildcard character * to indicate parts of the address not specified.

  If a wildcard lookup finds more than 500 addresses, Mail/VE will request that you enter more restrictive lookup criteria.

- The search is not case-sensitive. That is, searching for 'Ziggy' will find 'ziggy', 'ZIGGY', 'Ziggy', and so on.

- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.

- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example lists all mailboxes with the surname Jackson.

```
Mail/lookup_address '* Jackson' ..
Mail../address_options=(mailbox)
   1.   Mailbox:    Personal name: Jackson, John B.
   2.   Mailbox:    Personal name: Jackson, Ken
```

The following example lists all mailbox names whose given name begins with the letter J and whose surname is Hanson:

```
Mail/lookup_address 'J* Hanson' ..
Mail../address_options=(mailbox)
   1.   Mailbox:    Personal name: Hanson, John A.
   2.   Mailbox:    Personal name: Hanson, Jonathon
```

The following example lists all public mailboxes:

```
Mail/lookup_address '*' ao=(public mailbox)
    1.   Mailbox:    Personal name: EMAIL_NEWS
    2.   Mailbox:    Personal name: SDS_NEWS
```

$MAILBOX
MAIL Function

Purpose Returns a local address record containing the address of the mailbox selected for read operations.

Format **$MAILBOX**

Parameters None.

Remarks

- Use LIST_MAIL to establish a letter list before using the $MAILBOX function. $MAILBOX returns the local address of the mailbox for which a letter list is created. If a letter list does not exist, $MAILBOX returns a null value.
- The format of the local address record:

```
record
  personal_name: record
    surname: string 1..40
    given_name: string 0..16
    initials: string 0..5
    generation_qualifier: string 0..3
  recend
  organization_units: list 0..4 of string 0..32
  organization_name: string 0..64
recend
```

- For more information, see the Mail/VE Version 2 Usage manual.

Examples In the following example, the name associated with the mailbox selected for read operations is displayed:

```
Mail/display_value $mailbox.personal_name
Jones
Walter
```

$MAILBOX_ATTRIBUTE MAIL Function

Purpose Returns the value of a specified mailbox attribute.

Format

```
$MAILBOX_ATTRIBUTE
   (ADDRESS: string or record
   OPTION: keyword )
```

Parameters

ADDRESS

Specifies the address of the mailbox for which an attribute value is to be returned. Only the personal name is required. Specify organization unit(s) and organization name to further define the address.

The mailbox must be owned by the caller or a null string is returned.

Enter the address as a 1- to 64-character string or as a record in the format:

```
record
  personal_name: record
    surname: string 1..40
    given_name: string 0..16
    initials: string 0..5
    generation_qualifier: string 0..3
  recend
  organization_units: list 0..4 of string 0..32
  organization_name: string 0..64
recend
```

See chapter 2, Mail/VE Addresses, in the Mail/VE Version 2 Usage manual if you need more information on addressing.

This parameter is required.

OPTION

Specifies the name of the attribute value to be returned. Options are:

ADDRESS or A

Returns the mailbox address. If the mailbox is not owned by the caller, a null string is returned. The address is returned as a list of local address records in the following format:

```
record
  personal_name: record
    surname: string 1..40
    given_name: string 0..16
    initials: string 0..5
    generation_qualifier: string 0..3
  recend
  organization_units: list of 0..4 of string 0..32
  organization_name: string 0..64
recend
```

PERMISSION_LIST or PL

Returns the permission list entries. The value is returned as a list of access permission records.

PERMISSION_LIST_SIZE or PLS

Returns an integer specifying the number of entries in the permission list.

PERMIT_TYPE or PT

Returns a string containing PRIVATE, PUBLIC, or GROUP.

DESCRIPTION or D

Returns a string of 1 to 64 characters that describes the mailbox.

RETENTION_PERIOD or RP

Returns an integer (from 1 to 365) specifying how long a letter that has been displayed remains in the mailbox to which it was sent before being deleted from the system.

TELEPHONE_NUMBER of TN

Returns the phone number associated with the mailbox in a string of up to 32 characters.

This parameter is required.

Remarks

- Mailbox attributes that have not been defined return a null string.
- This function returns address records in the following formats.

  Local address records:

```
record
  personal_name: record
    surname: string 1..40
    given_name: string 0..16
    initials: string 0..5
    generation_qualifier: string 0..3
  recend
  organization_units: list 0..4 of string 0..32
  organization_name: string 0..64
recend
```

  Global address records:

```
record
  personal_name: record
    surname: string 1..40
    given_name: string 0..16
    initials: string 0..5
    generation_qualifier: string 0..3
  recend
  organization_units: list 0..4 of string 0..32
  organization_name: string 0..64
  private_domain: string 0..16
  administrative_domain: string 0..16
  country: string 0..3
  x121_address: string 0..15
  terminal_identifier: string 0..24
  ua_identifier: string 0..32
  domain_defined_attributes: list 0..4
    record
      name: string 1..8
      value: string 1..128
    recend
recend
```

- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example displays the description for each owned mailbox:

```
Mail/mailbox_list=$owned_mailboxes
Mail/for each box in mailbox_list do
for/disv $mailbox_attribute(box d)
for/forend
```

$OWNED_DISTRIBUTION_LISTS
MAIL Function

Purpose Returns a list of records containing the addresses of the distribution lists you own.

Format **$OWNED_DISTRIBUTION_LISTS**

Parameters None.

Remarks

- The local address record is returned in the following format:

```
record
  personal_name: record
   surname: string 1..40
   given_name: string 0..16
   initials: string 0..5
   generation_qualifier: string 0..3
  recend
  organization_units: list 0..4 of string 0..32
  organization_name: string 0..64
recend
```

- If you do not own any distribution lists, an empty list is returned.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example shows the number of addresses in each of your distribution lists:

```
Mail/for each dl in $owned_distribution_list
for/display_value dl.distribution_list_name//
for../' contains '// ..
for../$strrep( ..
for../$distribution_list_attribute ..
for../ (dl,address_list_size)// ..
for../' addresses'
for/forend
DTEAM contains 6 addresses
DEPT_LIST contains 9 addresses
Mail/
```

$OWNED_MAILBOXES
MAIL Function

Purpose Returns a list of records containing the local address of all of the mailboxes you own.

Format **$OWNED_MAILBOXES**

Parameters None.

Remarks

- The local address record is returned in the following format:

```
record
  personal_name: record
    surname: string 1..40
    given_name: string 0..16
    initials: string 0..5
    generation_qualifier: string 0..3
  recend
  organization_units: list 0..4 of string 0..32
  organization_name: string 0..64
recend
```

- If you have no mailboxes, an empty list is returned.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example uses Mail/VE functions to check owned mailboxes for mail:

```
Mail/mailbox_list = $owned_mailboxes
Mail/for each box in mailbox_list do
for/read_mail personal_name = ..
for../box.personal_name mail_options = all
for/forend
Mail/
```

READ_LETTERS
MAIL Subcommand

Purpose Displays one or more letters, receipts, or notices from the letter list.

Format

READ_LETTERS or
READ_LETTER or
REAL or
READ
LETTERS=list of range of: keyword or *integer* or *string*
PARTS=keyword or *list of range of integer*
RECIPIENTS=keyword or *list of keyword*
DISPLAY_OPTIONS=keyword
OUTPUT=file
STATUS=status variable

Parameters *LETTERS* or *LETTER* or *L*

Specifies the letter to be displayed. Specify a letter number, letter identifier, or keyword.

The letter number is the number assigned to the letter on the letter list. It can change each time you enter the LIST_MAIL subcommand. The letter identifier is permanently assigned to the letter by Mail/VE when it is delivered to the mailbox.

Keyword options:

CURRENT or C

Selects the current letter from the letter list. (The current letter is the one most recently displayed.)

FIRST or F

Selects the first letter in the letter list.

LAST or L

Selects the last letter in the letter list.

NEXT or N

Selects the letter following the current letter in the letter list.

PREVIOUS or P

Selects the letter preceding the current letter in the letter list.

If LETTER is omitted, CURRENT is used.

PARTS or *PART* or *P*

Parameter Attributes: BY_NAME

Specifies a list of letter part ordinals, a range of letter part ordinals, or a keyword. Integer values can range from 1 to the number of parts that make up the letter body.

Determine the number of letter parts by executing the LIST_MAIL subcommand with the DISPLAY_OPTIONS parameter set to FULL, or by displaying the value of the $LETTER_ATTRIBUTE function with the OPTION parameter set to NUMBER_OF_LETTER_PARTS.

Options:

TEXT

Includes letter parts containing text.

FORWARD

Includes forwarded letter parts.

NONE

Displays no letter parts. The letter header is displayed depending on the display option specified.

If omitted, TEXT and FORWARD are used. Letter parts of other types are represented in the display with a header line.

RECIPIENTS or *RECIPIENT* or *R*

Parameter Attributes: BY_NAME

Specifies the recipient lists to include in the display. Specify ALL, NONE, or one or more of the following keyword options:

TO

Includes the primary recipient list.

CC

Includes the courtesy copy recipient list.

BCC

Includes the blind courtesy copy recipient list. The BCC list can be displayed only by members of the list.

If omitted, ALL is used.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Parameter Attributes: BY_NAME

Specifies the information from the letter header to display. Options:

BRIEF or B

Includes the letter identifier, subject, date/time sent, letter attributes, and local information about the originator and recipients.

FULL or F

Includes all information contained in the letter header.

NONE

Includes no letter header information.

If DISPLAY_OPTION is omitted, BRIEF is used.

OUTPUT or *O*

Parameter Attributes: BY_NAME

Specifies the name of the file to which the display is written. If omitted, the display is written to the output file that you specified on the EMAIL command or, by default, to your terminal screen.

Remarks

- Only TEXT and FORWARD letter parts can be displayed using this subcommand. See the COPY_LETTER_PARTS subcommand for processing other letter part types.
- If no letter list exists, or if the letter you specify is not in the current letter list, Mail/VE displays an error message.
- The $LETTER function always returns the value of the letter last displayed.
- If execution of a LIST_MAIL subcommand results in an empty letter list, there is no letter available for display using the READ_LETTER subcommand.
- REPLY REQUESTED is the only recipient/delivery option displayed when you set the DISPLAY_OPTIONS parameter to BRIEF.
- All mail system messages are preceded and followed with double asterisks.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example displays the full letter header and the primary recipient list along with the letter:

```
Mail/read_letter ..
Mail../display_option=full ..
Mail../recipients=to
                Letter id: KI2
          Date/time sent: May 1, 1989.12:02 PM
      Date/time received: May 1, 1989.12:03 PM
            Retain until: May 9, 1989.12:02 PM
       Date/time expires: May 9, 1989.12:02 PM
       Delivery priority: NORMAL PRIORITY
              Conversion: ALLOWED
              Message is: UNCERTIFIED UNSEEN LETTER
            Message size: 277 (bytes)
              Message id: 580B90B52B27-MTA
           Personal name: Jan Banks
     Organization units: STAFF
Administrative domain: ATTMAIL
                 Country: US


                Importance: NORMAL IMPORTANCE

To:      Personal name: John Watercott
     Organization units: COMMUNICATIONS
Administrative domain: ATTMAIL
                 Country: US
From:    Personal name: Jan Banks
     Organization units: STAFF
Administrative domain: ATTMAIL
                 Country: US
Subject: Automobile Leasing

------------- Letter Body Part 1 - Text -------------

Please notify employees of the new auto leasing
program available through Crystal Motors.
See file A1 Crystal for details.

 ** END OF MESSAGE **
```

The following example displays only the letter header in brief mode.

```
Mail/read_letter ..
Mail../parts=none ..
Mail../recipients=none
Date:  May 1, 1989.12:02 PM  Msg-ID: 580B90B52B27-MTA
                             Jan Banks  OU=STAFF

To:    John Watchcott        OU=COMMUNICATIONS
From:  Jan Banks             OU=STAFF
Subj:  Automobile Leasing
 ** END OF MESSAGE **
```

RETAIN_LETTER MAIL Subcommand

Purpose Changes the length of time before a letter in the current letter list is deleted from the mail system. The length of time is called the retention period.

Format

RETAIN_LETTER or
RETL
LETTER=keyword or *integer* or *string*
DAYS=keyword or *integer*
STATUS=status variable

Parameters *LETTER* or *L*

Specifies the letter for which the retention period is to be changed. Identify the letter by letter identifier, letter number, or keyword.

The letter number is the number assigned to the letter on the letter list generated when you enter the LIST_MAIL subcommand. This number is subject to change each time you use LIST_MAIL. The letter identifier is permanently assigned to the letter by Mail/VE when it is delivered to the mailbox.

Keyword options:

CURRENT or C

Selects the current letter from the letter list. (The current letter is the letter most recently displayed.)

FIRST or F

Selects the first letter in the letter list.

NEXT or N

Selects the letter following the current letter in the letter list.

LAST or L

Selects the last letter in the letter list.

PREVIOUS or P

Selects the letter preceding the current letter in the letter list.

If LETTER is omitted, CURRENT is used.

DAYS or *DAY* or *D*

Parameter Attributes: BY_NAME

Specifies the number of days from today that the letter is to be retained in the mailbox.

Specify an integer (1 to 365) or the keyword MAXIMUM. MAXIMUM means the letter is kept in the mailbox for the maximum retention period for which you are validated. If you specify a value greater than your validated limit, the value of your MAXIMUM retention period is used.

If omitted, MAXIMUM is used.

Remarks

- Only the owner of a mailbox or the Mail/VE administrator can change the retention period of letters in a mailbox. They can also delete the letters.
- If the letter you specify on the LETTER parameter is not in the letter list or has been deleted, Mail/VE displays an error message.
- If a letter list does not exist, the subcommand terminates and the system displays a message.
- The $LETTER function always returns the value of the letter last displayed (called the current letter).
- Your NOS/VE system administrator sets retention period limits for each user, and users cannot set retention periods longer than the limit.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example changes the retention period of letter number 7 in the current letter list to 14 days from today:

```
Mail/retain_letter letter=7 days=14
```

The following example changes the retention period of the current letter in the letter list to 10 days from today:

```
Mail/retain_letter days=10
```

The following example changes the retention period of the letter with letter identifier JA42 to the maximum number of days the user is allowed to retain letters:

```
Mail/retain_letter letter='ja42'
```

SELECT_IDENTITY MAIL Subcommand

Purpose Selects the mailbox you want to use as your identity within Mail/VE for the current mail session.

Format

SELECT_IDENTITY or
SELI
PERSONAL_NAME=string or **record**
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
STATUS=status variable

Parameters

PERSONAL_NAME or **PN**

Specifies the name of the mailbox you are selecting as your identity. Enter the name as a 1- to 64-character string or a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual for details on specifying a mailbox name.

This parameter is required.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the mailbox address. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the mailbox address. Enter a string of 1 to 64 characters.

Remarks

- The first mailbox created by or for you is automatically selected as your identity and as your default mailbox.

  Your identity is used:

  - To verify your access to group mailboxes and distribution lists.
  - As the From address when you send a letter.

  Your default mailbox is the identity under which you begin each mail session. Use the SET_DEFAULT_MAILBOX subcommand described later in this chapter to change your default mailbox.

  See the chapter 1 of the Mail/VE Version 2 Usage manual, under Mail/VE Identity, if you need further information on identity and default mailboxes.

- If the address parameters do not specify a mailbox you own, the command terminates and the system displays an error message. To list your mailboxes, enter:

```
display_mailboxes personal_name=all ..
  display_options=full
```

- The value returned for the $IDENTITY function is the local address of the mailbox you select using the SELECT_IDENTITY subcommand. To display your Mail/VE identity, enter DISPLAY_VALUE $IDENTITY.

- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.

- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example shows selection of a new identity mailbox, which is then automatically used as the From address for the letters sent by this user during the current mail session:

```
Mail/select_identity 'Test_Coordinator'
Mail/write_letter
Wril/set_subject 'Test_Update'
Wril/add_letter_part $local.note
Wril/add_to 'Test_Distribution'
Wril/end_write_letter
```

SELECT_LETTER MAIL Subcommand

Purpose Specifies the value returned when you use the $LETTER function.

Format

SELECT_LETTER or
SELL
LETTER=keyword or *integer* or *string*
STATUS=status variable

Parameters *LETTER* or *L*

Specifies the letter to be selected and makes it the current letter. The value returned when you use the $LETTER function is set to the letter identifier of the letter selected. Specify a letter number, letter identifier, or keyword.

The letter number is the number assigned to the letter on the letter list. It can change each time you enter the LIST_MAIL subcommand. The letter identifier is permanently assigned to the letter by Mail/VE when it is delivered to the mailbox.

Keyword options:

CURRENT or C

Selects the current letter from the letter list. (The current letter is the one most recently displayed.)

FIRST or F

Selects the first letter in the letter list.

LAST or L

Selects the last letter in the letter list.

NEXT or N

Selects the letter following the current letter in the letter list.

PREVIOUS or P

Selects the letter preceding the current letter in the letter list.

If LETTER is omitted, $LETTER is used.

Remarks

- If no letter list exists or the specified letter is not included in the letter list, an informative status is returned.
- If no letter is selected, a null string is returned when you use the $LETTER function.
- For more information, see the Mail/VE Version 2 Usage manual.

SET_ATTRIBUTES CHADL and CREDL Subcommand

Purpose Defines or changes distribution list attributes.

Format

SET_ATTRIBUTES or
SET_ATTRIBUTE or
SETA
DISTRIBUTION_LIST_NAME=list of: keyword or *string* or *record*
ORGANIZATION_UNITS=list of: keyword or *string*
ORGANIZATION_NAME=string
DESCRIPTION=string
TELEPHONE_NUMBER=string
EXPAND=boolean
PERMIT_TYPE=keyword
STATUS=status variable

Parameters *DISTRIBUTION_LIST_NAME* or *DLN*

Specifies the name of the distribution list to be defined or changed.

For a new distribution list, specify a list of one to three names. The first name is the primary name, used when information on the list is displayed. The second and third names are aliases. Use the NO_CHANGE keyword as a place holder in the list if you want to change an alias. For an example, see the Examples section under the description of the SET_ATTRIBUTES subcommand in the Mail/VE Version 2 Usage manual.

Enter each name as either a 1- to 64-character string or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 manual if you need help on naming distribution lists.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) to be associated with the distribution list being created or changed. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified. Use the NO_CHANGE keyword as a place holder if you want to change the organization unit list. For an example, see the Examples section under the description of the SET_ATTRIBUTES subcommand in the Mail/VE Version 2 Usage manual.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the distribution list. Enter a 1- to 64-character string.

DESCRIPTION or *D*

Parameter Attributes: BY_NAME

Specifies a 1- to 64-character string describing the distribution list.

TELEPHONE_NUMBER or *TN*

Parameter Attributes: BY_NAME

Specifies a 1- to 32-character phone number to be associated with the distribution list.

EXPAND or *E*

Parameter Attributes: BY_NAME

Indicates whether or not the distribution list will be expanded when used. The EXPAND parameter applies only to distribution lists residing on the same host as the sender. Otherwise, no expansion can occur regardless of the value of the EXPAND attribute.

TRUE

The distribution list name does not appear in the letter header but is replaced by the addresses contained in the distribution list.

FALSE

The distribution list name is retained in the letter header but the addresses contained in the distribution list are not in the header.

If EXPAND is omitted, this attribute remains unchanged.

PERMIT_TYPE or *PT*

Parameter Attributes: BY_NAME

Specifies the type of distribution list permissions allowed. Keyword options are:

PRIVATE or PR

Only the owner of the distribution list can display the addresses in the list and use the list to address letters.

PUBLIC or PU

The distribution list is available for anyone to use. No further permissions are required.

GROUP or G

A group of mailboxes is permitted to use the distribution list and/or display the addresses in the distribution list.

Use the ADD_PERMISSION and DELETE_PERMISSION subcommands to edit the permissions. Use the DISPLAY_PERMISSION subcommand to display the list of mailboxes permitted to use a group distribution list.

If omitted when creating a distribution list, PRIVATE is used. If omitted when changing a distribution list, the attribute remains unchanged.

Remarks

- The DESCRIPTION and TELEPHONE_NUMBER attributes are informative only. They are not considered in address lookup procedures.
- If a group distribution list is changed to a public or private distribution list, the permission list associated with the distribution list is deleted.
- Organization names and units must be defined by the mail administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples

The following example creates a distribution list:

```
Mail/create_distribution_list
Cred1/set_attributes ..
Cred1../dln=('Market_Research' 'MRDL') ..
Cred1../ou=('Consumer Surveys' ..
Cred1../'Interior Products') ..
Cred1../description='owned by Jerry Bystrom' ..
Cred1../telephone_number='402-981-6743' ..
Cred1../expand=false permit_type=public
Cred1/add_address 'Barry Fisher'
Cred1/add_address 'Lonnie Merrill'
Cred1/add_address 'Shirley Dreckman'
Cred1/end_create_distribution_list
```

In the following example, the third organization unit associated with a distribution list is changed. Notice the use of the NO_CHANGE (NC) keyword as a place holder.

```
Mail/change_distribution_list 'Market_Research'
Chadl/set_attribute ..
Chadl../ou=(nc,nc,'Home Furnishings')
Chadl/end_change_distribution_list
```

SET_ATTRIBUTES
CHAM and CREM Subcommand

Purpose Defines or changes mailbox attributes.

Format

SET_ATTRIBUTES or
SET_ATTRIBUTE or
SETA
PERSONAL_NAME=list of: keyword or *string* or *record*
ORGANIZATION_UNITS=list of: keyword or *string*
ORGANIZATION_NAME=string
DESCRIPTION=string
TELEPHONE_NUMBER=string
RETENTION_PERIOD=keyword or *integer*
PERMIT_TYPE=keyword
STATUS=status variable

Parameters *PERSONAL_NAME* or *PN*

Specifies the personal name address attribute of the mailbox.

To define a new mailbox, specify a list of one to three names. The first name is the primary name used when information is displayed about the address. The second and third names are aliases. Use the NO_CHANGE keyword as a place holder in the list if you want to change an alias. For an example, see the Examples section under the description of SET_ATTRIBUTES in the Mail/VE Version 2 Usage manual. Enter each name as a 1- to 64-character string or as a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual for further information on mailbox names.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) to be associated with the mailbox. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies an organization name to be used in addressing the mailbox. Enter a 1- to 64-character string.

DESCRIPTION or *D*

Parameter Attributes: BY_NAME

Specifies a string of 1 to 64 characters describing the mailbox.

TELEPHONE_NUMBER or *TN*

Parameter Attributes: BY_NAME

Specifies a 1- to 32-character telephone number to be associated with the mailbox.

RETENTION_PERIOD or *RP*

Parameter Attributes: BY_NAME

Specifies the number of days a letter is retained in the mailbox after it is displayed. Enter an integer (1 to 365) or the keyword MAXIMUM. If you specify MAXIMUM, the letter retention is the maximum as defined by your NOS/VE user validation.

If omitted when creating a mailbox, MAXIMUM is used.

PERMIT_TYPE or *PT*

Parameter Attributes: BY_NAME

Specifies the type of mailbox permissions allowed. Options:

PRIVATE or PR

Only the owner of the mailbox can read the contents and originate letters from it.

PUBLIC or PU

The mailbox is available for anyone to read from or send to. No further permissions are required.

GROUP or G

A group of mailboxes is permitted to read the contents of the mailbox and/or write letters to it.

Use the ADD_PERMISSION and DELETE_PERMISSION subcommands to edit the permissions. Use DISPLAY_PERMISSION to display the permission list that defines a group mailbox.

If omitted when creating a mailbox, PRIVATE is used. If omitted when changing a mailbox, the attribute remains unchanged.

Remarks

- The RETENTION_PERIOD parameter specifies the retention period of new letters arriving in this mailbox. The retention period of letters already in the mailbox is not changed.
- The DESCRIPTION and TELEPHONE_NUMBER parameters specify documentary information that is not considered when evaluating a mailbox address in lookup procedures.
- If a group mailbox is changed to a public or private mailbox, the permission list associated with the mailbox is deleted.
- Organization names and units must be defined by the Mail/VE administrator. Use the DISPLAY_ORGANIZATIONS subcommand to list the defined organization names/units.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples

The following example creates a new mailbox:

```
Mail/create_mailbox
Crem/set_attributes ..
Crem../personal_name=( ..
Crem../'David M Watkins' ..
Crem../'DMW' ..
Crem../'Dave Watkins') ..
Crem../description='owned by Dave Watkins' ..
Crem../telephone_number='239-816-5903'
Crem/end_create_mailbox
```

The following example changes the third alias of the mailbox name. Notice the use of the NO_CHANGE (NC) keyword as a place holder.

```
Mail/change_mailbox 'LPR'
Cham/set_attribute ..
Cham../pn=(nc,nc,'Lana Roche')
Cham/end_change_mailbox
```

SET_DELIVERY_OPTIONS FORL, WRIL, and WRIR Subcommand

Purpose Defines or changes the delivery options of the letter being composed.

Format

SET_DELIVERY_OPTIONS or
SETDO or
SET_ATTRIBUTE or
SET_ATTRIBUTES or
SET_DELIVERY_OPTION or
SETA
IMPORTANCE=keyword
SENSITIVITY=keyword
DELIVERY_PRIORITY=keyword
CONVERSION_PROHIBITED=boolean
DISCLOSE_OTHER_RECIPIENTS=boolean
ALTERNATE_RECIPIENT_ALLOWED=boolean
RETURN_CONTENTS=boolean
DELIVERY_CONFIRMATION=keyword
STATUS=status variable

Parameters *IMPORTANCE* or *I*

Specifies the importance of the letter. This attribute is informative only, and does not imply special handling by Mail/VE or any other receiving mail system. Keyword options are:

LOW or L

The letter is of minor importance.

NORMAL or N

The letter is of routine importance.

HIGH or H

The letter is important.

NONE

The letter does not include an importance rating.

If omitted, NONE is used.

SENSITIVITY or *SE*

Specifies the sensitivity of the letter. Keyword options:

PERSONAL or PE

The letter is personal.

PRIVATE or PR

The letter content should be kept private.

CONFIDENTIAL or C

The letter content should be kept confidential.

NONE

The letter does not include a sensitivity marking.

If omitted, NONE is used.

DELIVERY_PRIORITY or *DP*

Specifies how the mail system should handle delivery of the letter through the network. This attribute implies no special handling by Mail/VE but may have significance for other X.400 mail systems. Keyword options:

URGENT or U

The letter should be delivered using the fastest network route available.

NON_URGENT or NU

The letter should be delivered using the slowest network route available.

NONE

The letter should be delivered within the default timeframe for the mail network at a site.

If omitted, NONE is assumed.

CONVERSION_PROHIBITED or *CP*

Specifies whether the letter can be converted to a different encoding for delivery. Mail/VE does not convert letters to a different encoding. This parameter applies to other X.400 mail systems. Options are:

TRUE

The letter cannot be converted. You may want to set this parameter to TRUE if a faulty conversion would result in a significant loss of information.

FALSE

The letter can be converted.

If omitted, FALSE is used.

DISCLOSE_OTHER_RECIPIENTS or *DISCLOSE_OTHER_RECIPIENT* or *DOR*

Specifies whether the names in the delivery address list should be displayed to letter recipients. This attribute implies no special handling by Mail/VE but may be used by other mail systems. The delivery list may be all or part of the primary, courtesy copy, and blind courtesy copy lists, plus any distribution list expansions. The delivery address list may not contain addresses for which the local mail system does not have delivery responsibility. Keyword options are:

TRUE

The names in the delivery address list will be disclosed to letter recipients.

FALSE

The names in the delivery address list will not be disclosed to letter recipients.

If omitted, FALSE is used.

ALTERNATE_RECIPIENT_ALLOWED or *ARA*

Specifies whether the letter can be delivered to an alternate recipient. This attribute implies no special handling by Mail/VE but may have significance for other X.400 mail systems. Keyword options are:

TRUE

The letter will be delivered to an alternate recipient if it cannot be delivered to one of the recipients named in the primary, courtesy copy, or blind courtesy copy list.

FALSE

If delivery cannot be made to recipients identified in the primary, courtesy copy, or blind courtesy copy lists, it will not be delivered to an alternate recipient.

If omitted, FALSE is used.

RETURN_CONTENTS or *RETURN_CONTENT* or *RC*

Specifies whether the letter contents are to be returned if nondelivery occurs. Mail/VE does not provide a way to display or resend the returned contents. Mail/VE does not return the contents of a message with a nondelivery message. This parameter applies to message transfers to other mail systems that do return the contents. Keyword options are:

TRUE

The letter contents are returned in a notice.

FALSE

The letter is not returned.

If omitted, FALSE is used.

DELIVERY_CONFIRMATION or *DC*

Specifies what delivery notices, if any, are to be returned. Keyword options:

DELIVERY or D

Returns a delivery notice for every delivery and nondelivery. For BITNet and Internet addresses, a delivery notice means the message has been transferred by Mail/VE; the notice does not mean that the intended recipient actually received it.

NON_DELIVERY or ND

Returns a delivery notice for every nondelivery. For BITNet and Internet addresses, the Mail/VE gateway returns a nondelivery if the transfer does not occur.

NONE

Does not return any delivery notices. This option does not apply to BITNet and Internet addresses.

If omitted, NON_DELIVERY is used.

Remarks

- When preparing a letter, if you enter more than one SET_DELIVERY_OPTIONS subcommand with the same options, the value specified on the last subcommand is used.
- The options you select on SET_DELIVERY_OPTIONS apply to all recipients of the letter. You cannot set these delivery options on an individual basis.
- For more information, see the Mail/VE Version 2 Usage manual.

SET_RECIPIENT_OPTIONS FORL, WRIL, and WRIR Subcommand

Purpose Defines a set of recipient options that apply to all members of recipient address lists. When issued, this subcommand sets defaults that are used only if subsequent calls to the ADD_TO, ADD_COURTESY_COPY, and ADD_BLIND_COURTESY_COPY subcommands do not include the RECIPIENT_OPTIONS parameter.

Format

SET_RECIPIENT_OPTIONS or
SET_RECIPIENT_OPTION or
SETRO
RECIPIENT_LISTS=keyword or **list of keyword**
RECIPIENT_OPTIONS=keyword or *list of keyword*
CHANGE_RECIPIENTS_IN_LISTS=boolean
STATUS=status variable

Parameters

RECIPIENT_LISTS or **RECIPIENT_LIST** or **RL**

Specifies the recipient list for which options are set. Choose ALL or one or more of the following keyword options:

TO

Sets the recipient options for the To address list.

BLIND_COURTESY_COPY or BCC

Sets the recipient options for the blind courtesy copy address list.

COURTESY_COPY or CC

Sets the recipient options for the courtesy copy address list.

This parameter is required.

RECIPIENT_OPTIONS or *RECIPIENT_OPTION* or *RO*

Specifies the recipient options. The values specified are used as the default recipient options for subsequent ADD_TO, ADD_BLIND_COURTESY_COPY, and ADD_COURTESY_COPY subcommands.

Specify ALL, NONE, or one or more of the following keywords. ALL means the letter is certified for receipt or nonreceipt notification and indicates in the header information that a reply is requested. NONE means no letter attributes are set for the specified address.

CERTIFY_RECEIPT or CR

The originator receives a message when the recipient has seen the letter. The letter is considered seen if the recipient displays or copies the letter to a file.

CERTIFY_NON_RECEIPT or CNR

The originator receives a message if the letter is deleted without having been seen. The mail system generates a nonreceipt notice if the letter is unseen at the time of a deletion or the letter is autoforwarded.

REPLY_REQUESTED or RR

The letter header information indicates that the originator is expecting a reply. The system does not require a reply.

If omitted, NONE is assumed.

CHANGE_RECIPIENTS_IN_LISTS or *CRIL*

Specifies a boolean value to indicate whether or not the recipient options are to be applied to existing recipient address lists.

TRUE

The recipient options value(s) are applied to existing recipient lists.

FALSE

The recipient options are not applied to existing recipient lists. The recipient options will only be used as default values for the RECIPIENT_OPTIONS parameter of subsequent ADD_TO, ADD_BLIND_COURTESY_COPY, and ADD_COURTESY_COPY subcommands.

If omitted, FALSE is used.

Remarks SET_RECIPIENT_OPTIONS does not apply to distribution lists that cannot be expanded (that is, lists created with the EXPAND parameter on the SET_ATTRIBUTES subcommand set to FALSE) or to lists not located on the same system as the sender.

For more information, see the Mail/VE Version 2 Usage manual.

SET_DEFAULT_MAILBOX MAIL Subcommand

22

Purpose Establishes the mailbox address that will be used as the identity under which you begin a Mail/VE session.

Format

SET_DEFAULT_MAILBOX or
SETDM
PERSONAL_NAME=string or **record**
ORGANIZATION_UNITS=list of string
ORGANIZATION_NAME=string
STATUS=status variable

Parameters

PERSONAL_NAME or PN

Specifies the name of the mailbox to be used as your identity. Enter the name as a 1- to 64-character string or a record in the format:

```
record
  surname: string 1..40
  given_name: string 0..16
  initials: string 0..5
  generation_qualifier: string 0..3
recend
```

See Address Names in the Mail/VE Version 2 Usage manual for details on specifying a mailbox name.

This parameter is required.

ORGANIZATION_UNITS or *ORGANIZATION_UNIT* or *OU*

Specifies the organization unit(s) associated with the mailbox address. Enter a list of one to four strings of 1 to 32 characters each.

Address lookup procedures determine a match based on the order specified.

ORGANIZATION_NAME or *ON*

Specifies the organization name associated with the specified mailbox. Enter a string of 1 to 64 characters.

Remarks

- To display your default mailbox, enter DISPLAY_VALUE $DEFAULT_MAILBOX.
- If the addressing parameters do not specify a mailbox you own, the command terminates and the system displays an error message. To list mailboxes you own, enter:

```
display_mailboxes personal_name=all ..
  display_options=full
```

- The first mailbox created by or for you is automatically selected as your Mail/VE identity and becomes your default mailbox. Your identity is used:
  - To verify your access to group mailboxes and distribution lists.

- As the From address when you send a letter.

  Use the SELECT_IDENTITY subcommand, described earlier in this chapter to change your identity.

- Use the $IDENTITY function to display the local address of the mailbox used as your identity. To display your Mail/VE identity, enter DISPLAY_VALUE $IDENTITY.

  See chapter 1 of the Mail/VE Version 2 Usage manual, under Mail/VE Identity, for further information on identity and default mailboxes.

- For more information, see the Mail/VE Version 2 Usage manual.

Examples The following example establishes an identity and default mailbox within Mail/VE and then displays the name of the identity mailbox.

```
Mail/select_identity pn='Rose Mendoza'
Mail/set_default_mailbox pn='Rose Mendoza'
Mail/end_email
/email
Mail/display_value $identity
Rose Mendoza
```

22

SET_SUBJECT
FORL, WRIL, and WRIR Subcommand

Purpose Defines a subject for the letter being composed.

Format

SET_SUBJECT or
SETS or
SUBJECT
SUBJECT=string
STATUS=status variable

Parameters

SUBJECT

Specifies the subject of the letter. Enter a string of 1 to 128 characters. This parameter is required.

Remarks For more information, see the Mail/VE Version 2 Usage manual.

WRITE_LETTER
MAIL Subcommand

Purpose Initiates the WRITE_LETTER subutility, through which you can compose and send a letter to one or more addresses. When you enter WRITE_LETTER, the system displays the prompt Wril/.

Format

WRITE_LETTER or
WRIL
PROLOG=file
STATUS=status variable

Parameters

PROLOG or *P*

Parameter Attributes: BY_NAME

Specifies the name of the file from which SCL or Mail/VE commands are read before giving control to the WRITE_LETTER subutility. If omitted, $NULL is used.

The $NULL default can be overridden by creating an SCL default variable called MVD$MAIL_WRITE_PROLOG and setting its value to the name of the prolog file. Specify the value of the name as a string.

Remarks

- The WRITE_LETTER subutility subcommands, listed next, allow you to add the letter body, specify letter attributes, define the recipient list, and send the letter.

  ADD_BLIND_COURTESY_COPY
  ADD_COURTESY_COPY
  ADD_LETTER_PARTS
  ADD_TO
  DELETE_BLIND_COURTESY_COPY
  DELETE_COURTESY_COPY
  DELETE_LETTER_PARTS
  DELETE_TO
  DISPLAY_BLIND_COURTESY_COPY
  DISPLAY_COURTESY_COPY
  DISPLAY_DELIVERY_OPTIONS
  DISPLAY_LETTER_PARTS
  DISPLAY_RECIPIENT_OPTIONS
  DISPLAY_SUBJECT
  DISPLAY_TO
  END_WRITE_LETTER
  SET_DELIVERY_OPTIONS

SET_RECIPIENT_OPTIONS
SET_SUBJECT

These subcommands are described under the Letter Posting Subutility Subcommands section of the Mail/VE Version 2 Usage manual.

- The address in the From field is set to the value returned by the $IDENTITY function.
- If an address is repeated, only one copy of the letter is sent to the address. This is true regardless of the form of the address or whether the address is included in more than one distribution list.
- Receipts are generated by the system when certified letters are read, copied, or printed.
- Notices are generated by the system, unless suppressed by the sender, if delivery confirmation is requested or when the mail system determines the letter cannot be delivered to an address.
- Nondelivery notices are not generated for incorrect local addresses in a distribution list or for any addresses in a distribution list whose EXPAND attribute is set to FALSE. Local addresses are validated when the distribution list name is entered in the recipient list and incorrect addresses are simply ignored.

  However, nondelivery notices are generated for incorrect nonlocal addresses in a local distribution list whose EXPAND attribute is set to TRUE. These addresses are not validated when they are added to the distribution list; they are validated when Mail/VE attempts to deliver mail to them.
- For more information, see the Mail/VE Version 2 Usage manual.

Examples In the following example the letter text is written on a file and sent to the members of a distribution list:

```
Mail/write_letter
Wril/collect_text $local.letter
Colt/The project meeting scheduled for
Colt/Wednesday has been cancelled.
Colt/**
Wril/add_letter_part $local.letter
Wril/set_subject 'Project Meeting Notice'
Wril/add_to 'Recreation_Unit' ..
Wril../recipient_options=certify_receipt
Wril/end_write_letter
```

WRITE_REPLY
MAIL Subcommand

Purpose Initiates the WRITE_REPLY subutility, through which you can compose and send a reply letter to the sender of a letter. When you enter WRITE_REPLY, the system displays the prompt `Wrir/`.

Format

WRITE_REPLY or
WRIR
LETTER=keyword or *integer* or *string*
PROLOG=file
STATUS=status variable

Parameters *LETTER* or *L*

Specifies the letter for which a reply is to be sent. Enter a letter number, letter identifier, or keyword value.

The letter number is from the letter list established when you enter the LIST_MAIL subcommand. Letter numbers can change each time you enter LIST_MAIL. The letter identifier is permanently assigned to the letter by Mail/VE when it is delivered to the mailbox.

Keyword options:

CURRENT or C

Selects the current letter from the letter list. (The current letter is the letter most recently displayed.)

FIRST or F

Selects the first letter in the list.

NEXT or N

Selects the letter following the current letter in the list.

PREVIOUS or P

Selects the letter preceding the current letter in the list.

LAST or L

Selects the last letter in the list.

If LETTER is omitted, CURRENT is used.

PROLOG or *P*

Parameter Attributes: BY_NAME

Specifies the name of the file from which commands are read before giving control to the WRITE_REPLY subutility.

If omitted, $NULL is used.

You can override the $NULL default by creating an SCL default variable called MVD$MAIL_WRITE_PROLOG and setting its value to the name of the prolog file. Specify the value of the name as a string.

Remarks

- Use the WRITE_REPLY subutility subcommands, listed next, to add the letter body, specify letter attributes, define recipient lists, and send a reply.

  ADD_BLIND_COURTESY_COPY
  ADD_COURTESY_COPY
  ADD_LETTER_PARTS
  ADD_TO
  DELETE_BLIND_COURTESY_COPY
  DELETE_COURTESY_COPY
  DELETE_LETTER_PARTS
  DELETE_TO
  DISPLAY_BLIND_COURTESY_COPY
  DISPLAY_COURTESY_COPY
  DISPLAY_DELIVERY_OPTIONS
  DISPLAY_LETTER_PARTS
  DISPLAY_RECIPIENT_OPTIONS
  DISPLAY_SUBJECT
  DISPLAY_TO
  END_WRITE_REPLY
  SET_DELIVERY_OPTIONS

SET_RECIPIENT_OPTIONS
SET_SUBJECT

These subcommands are described in the Letter Posting Subutility Subcommands section of the Mail/VE Version 2 Usage manual.

- The From field is set to the value of $IDENTITY and cannot be changed from within the WRITE_REPLY subutility.
- When the same mailbox is addressed more than once on the COURTESY_COPY or BLIND_COURTESY_COPY parameters, only one copy of the letter is sent to the address. The address, however, will appear in both the blind courtesy copy and courtesy copy address lists if you enter the address in both lists.
- The subject of the reply is set to the original subject preceded by `Reply to`. You can change the subject through the SET_SUBJECT subcommand.
- The address of the sender to whom you are writing a reply is automatically added to the primary recipient list (the addresses listed in the To field).
- Receipt letters are generated by the system when certified letters are read, copied, or printed.
- If an address is repeated, only one copy of the letter is sent to the address. This is true regardless of the form of the address or whether the address is included in more than one distribution list.
- Receipts are generated by the system when certified letters are read, copied, or printed.
- Notices are generated by the system, unless suppressed by the sender, if delivery confirmation is requested or when the mail system determines the letter cannot be delivered to an address.

- Nondelivery notices are not generated for incorrect local addresses in a distribution list or for any addresses in a distribution list whose EXPAND attribute is set to FALSE. Local addresses are validated when the distribution list name is entered in the recipient list and incorrect addresses are simply ignored.

  However, nondelivery notices are generated for incorrect nonlocal addresses in a local distribution list whose EXPAND attribute is set to TRUE. These addresses are not validated when they are added to the distribution list; they are validated when Mail/VE attempts to deliver mail to them.

  - For more information, see the Mail/VE Version 2 Usage manual.

Examples This example sends a reply to the letter most recently displayed from a file:

```
Mail/write_reply
Wrir/add_letter_part $local.note
Wrir/end_write_reply
```

MANAGE_FORM 23

MANAGE_FORM Command

Purpose MANAGE_FORM begins the MANAGE_FORMS utility. This utility allows you to display and manage forms. A form is a related group of objects shown on a user's terminal screen. Using the MANAGE_FORMS utility, you can display any form created through Screen Formatting.

Format

MANAGE_FORM or
MANAGE_FORMS or
MANF
VARIABLE_CREATION=keyword
VARIABLE_EVALUATION=keyword
STATUS=status variable

Parameters *VARIABLE_CREATION* or *VC*

The keyword indicating how the utility is to create variables when a form is opened. Use one of the following keywords:

FORM_VARIABLE

Create one variable for each form that is opened. The variable is an SCL record that contains a field for each variable text object on the form. The name of the form becomes the name of the variable. FORM_VARIABLE is the default.

SINGLE

Create one variable for each variable defined on the form. The type of the variable depends on the definition of the variable on the form.

NONE

Create no variables. You are responsible for defining all variables used on the forms you open.

VARIABLE_EVALUATION or *VE*

The keyword indicating how the utility is to evaluate variables. Use one of the following keywords:

AUTOMATIC

Automatic evaluation of variables. Screen Formatting updates all variables from added or combined forms when you execute either READ_FORMS or SHOW_FORMS. You do not need to enter the GET_FORM_VARIABLE and REPLACE_FORM_VARIABLE subcommands.

MANUAL

Manual evaluation of variables. Screen Formatting does not update variables automatically. You must update them using GET_FORM_VARIABLE and REPLACE_FORM_VARIABLE subcommands.

The default is MANUAL.

Remarks

- The forms used with the MANAGE_FORMS utility can be created through the Screen Design Facility (SDF) or with a CYBIL program that makes calls to Screen Formatting procedures.
- When executing the utility interactively, the *mf/* prompt is displayed.
- For more information, see the NOS/VE Screen Formatting manual.

ADD_FORM
MANF Subcommand

Purpose ADD_FORM schedules a form for display on the application user's screen.

Format

ADD_FORM or
ADDF
FORM_NAME=data_name
STATUS=status variable

Parameters **FORM_NAME** or **FN**

The name established when the form was opened. This parameter is required.

Remarks

- When you enter either the READ_FORMS or SHOW_FORMS subcommand, Screen Formatting displays the added form on the terminal screen. The added form is placed on top of other forms occupying the same area on the screen.
- When displayed, each form that is added operates independently from other forms that have been added. When a user executes a normal event, Screen Formatting validates and updates only those variables on the form associated with the event.

  To have forms share events, use the COMBINE_FORM subcommand.
- Before you add a form, you must open it.
- You cannot add a pushed form.
- For more information, see the NOS/VE Screen Formatting manual.

CHANGE_TABLE_SIZE MANF Subcommand

Purpose CHANGE_TABLE_SIZE changes the size of the table during application execution.

Format

CHANGE_TABLE_SIZE or
CHATS
FORM_NAME=data_name
TABLE_NAME=data_name
TABLE_SIZE=integer
STATUS=status variable

Parameters

FORM_NAME or FN

The name established when the form was opened. This parameter is required.

TABLE_NAME or TN

The name of the table to change in size. This parameter is required.

TABLE_SIZE or *TS*

The size of the table. While this subcommand is in effect, Screen Formatting limits the number of stored occurrences allowed for a table to the value you specify on this parameter. How many occurrences are displayed at one time depends on the number of visible occurrences defined in the form.

If you specify zero for the table size, no occurrences appear on the form.

Remarks

- The table must be present in an open form.
- The size limitation remains in effect until the next time you enter the CHANGE_TABLE_SIZE subcommand.
- The maximum size for a table is identified by the form as the maximum number of stored occurrences. If you specify a table size larger than the maximum, you receive an error message (FDE$INVALID_TABLE_SIZE).
- When you are evaluating variables manually, all entries of the table must have valid values. It is not sufficient to supply values for just those entries available after the execution of this subcommand.
- For more information, see the NOS/VE Screen Formatting manual.

Examples

The following examples describe how changing the size of a table affects the application user. On the form, the table's specifications are a maximum of 20 stored occurrences, of which 6 occurrences can be visible at one time.

- If you specify a table size of 10, Screen Formatting displays 6 occurrences and allows the application user to page to the 10th occurrence.
- If you specify a table size of 4, Screen Formatting displays 4 occurrences and does not allow the application user to page.

CLOSE_FORM
MANF Subcommand

Purpose CLOSE_FORM releases resources used to process a form and deletes the form from the list scheduled for display.

Format

CLOSE_FORM or
CLOF
FORM_NAME=data_name
STATUS=status variable

Parameters **FORM_NAME** or **FN**

The name established when the form was opened. This parameter is required.

Remarks

- When you enter either the READ_FORMS or SHOW_FORMS subcommand, Screen Formatting removes the closed form from the terminal screen as a result of entering this subcommand.
- Before you can close a form, you must open it.
- You cannot close a pushed form.
- For more information, see the NOS/VE Screen Formatting manual.

COMBINE_FORM
MANF Subcommand

Purpose COMBINE_FORM combines a form with a previously added form and schedules the combined form for display on the terminal screen.

Format

COMBINE_FORM or
COMBINE_FORMS or
COMF
ADDED_FORM_NAME=data_name
COMBINE_FORM_NAME=data_name
STATUS=status variable

Parameters

ADDED_FORM_NAME or AFN

The name of the previously added form. This parameter is required.

COMBINE_FORM_NAME or CFN

The name of the form you are combining with the previously added form. This parameter is required.

Remarks

- You cannot combine a pushed form.
- The combined form inherits the event definitions of the previously added form.
- Before you combine a form with a previously added form, you must open both forms.
- When you enter either the READ_FORMS or SHOW_FORMS subcommand, Screen Formatting displays the combined form. The combined form is placed on top of other forms occupying the same area on the screen.
- When you start the MANAGE_FORMS utility specifying VARIABLE_EVALUATION=AUTOMATIC and the application user executes an event to return to the utility normally, Screen Formatting updates all SCL variables associated with both the added and combined forms.
- When you start the MANAGE_FORMS utility specifying VARIABLE_EVALUATION=MANUAL and you enter the REPLACE_FORM_VARIABLE subcommand, Screen Formatting updates the variable on both the added and combined forms.
- To combine several forms with a previously added form, execute this subcommand more than once.
- For more information, see the NOS/VE Screen Formatting manual.

DELETE_FORM
MANF Subcommand

Purpose DELETE_FORM deletes the form from the list of forms scheduled for display.

Format

DELETE_FORM or
DELF
FORM_NAME=data_name
STATUS=status variable

Parameters

FORM_NAME or **FN**

The name established when the form was opened. This parameter is required.

Remarks

- When you enter either the READ_FORMS or SHOW_FORMS subcommand, Screen Formatting removes the deleted form from the terminal screen and replots any forms uncovered by the deleted form.
- When you add a form (ADD_FORM) again that you previously deleted, the data in the form is retained.
- Before you delete a form, you must open it.
- You cannot delete a pushed form.
- If the form was added and has any combined forms associated with it, the combined forms are also deleted.
- When you delete a combined form, only that form is deleted. Areas covered by the combined form are replotted after the combined form is deleted.
- For more information, see the NOS/VE Screen Formatting manual.

$EVENT_NAME
MANF Function

Purpose $EVENT_NAME returns the name of the event the application user executed to complete his or her interaction with a form.

Format

$EVENT_NAME

Parameters None.

Remarks

- An event is usually executed when the user presses the return key or a function key.
- For more information, see the NOS/VE Screen Formatting manual.

$EVENT_NORMAL MANF Function

Purpose $EVENT_NORMAL returns a boolean value specifying whether the event the user executed is defined as normal. The type of each event is defined when the form is created. The value of $EVENT_NORMAL is TRUE when the event is a normal event; it is FALSE when the event is not a normal event.

Format **$EVENT_NORMAL**

Parameters None.

Remarks

- When an event is normal, variables are validated and updated.
- When an event is abnormal, variables are not validated or updated.
- For more information, see the NOS/VE Screen Formatting manual.

$EVENT_POSITION MANF Function

Purpose $EVENT_POSITION returns information about the position of the event executed by the application user to complete interaction with a form. The information returned is determined by the keyword you specify in the parameter.

Format **$EVENT_POSITION**
(OPTION: keyword)

Parameters

OPTION

The keyword that specifies the type of information to be returned about an event position. Use one of the following keywords: .

CHARACTER_POSITION (CP)

Returns the character position within the object where the event occurred. The value returned is an integer; the first character position is 1. The CHARACTER_POSITION value is valid only if $EVENT_POSITION (OBJECT_EVENT) returns a TRUE value.

FORM_NAME (FN)

Returns the name of the form where the event occurred.

FORM_X_POSITION (FXP)

Returns the x position of the event on the form. The x position is an integer; 1 indicates the upper left corner of the form. The x position increases by 1 for each character, counting from left to right.

FORM_Y_POSITION (FYP)

Returns the y position of the event on the form. The y position is an integer; 1 indicates the upper left corner of the form. The y position increases by 1 for each character, counting from top to bottom.

OBJECT_EVENT (OE)

Returns a boolean value specifying whether the event occurred in an object on the form. The value returned is TRUE when the event occurred in an object and FALSE when it did not.

OBJECT_NAME (ON)

Returns the name of the object where the event occurred. The OBJECT_NAME value is valid only if $EVENT_POSITION (OBJECT_EVENT) returns a TRUE value.

OCCURRENCE (O)

Returns an integer indicating in which occurrence of the object the event occurred. The OCCURRENCE value is valid only if $EVENT_POSITION (OBJECT_EVENT) returns a TRUE value.

OBJECT_TYPE (OT)

Returns a keyword that indicates the type of object in which the event occurred. One of the following keywords is returned:

BOX
CONSTANT_TEXT
CONSTANT_TEXT_BOX
LINE
VARIABLE_TEXT
VARIABLE_TEXT_BOX

OBJECT_X_POSITION (OXP)

Returns the x position of the object on the form. The x position is an integer; 1 indicates the upper left corner of the form. The x position increases by 1 for each character, counting from left to right. The OBJECT_X_POSITION value is valid only if $EVENT_POSITION (OBJECT_EVENT) returns a TRUE value.

OBJECT_Y_POSITION (OYP)

Returns the y position of the object on the form. The y position is an integer; 1 indicates the upper left corner of the form. The y position increases by 1 for each character, counting from top to bottom. The OBJECT_Y_POSITION value is valid only if $EVENT_POSITION (OBJECT_EVENT) returns a TRUE value.

SCREEN_X_POSITION (SXP)

Returns the x position of the event on the screen. The x position is an integer; 1 indicates the upper left corner of the screen. The x position increases by 1 for each character, counting from left to right.

SCREEN_Y_POSITION (SYP)

Returns the y position of the event on the screen. The y position is an integer; 1 indicates the upper left corner of the screen. The y position increases by 1 for each character, counting from top to bottom.

Remarks For more information, see the NOS/VE Screen Formatting manual.

GET_FORM_VARIABLE MANF Subcommand

Purpose GET_FORM_VARIABLE gets the value the user entered on the form for a variable and transfers it to SCL.

Format

GET_FORM_VARIABLE or
GET_FORM_VARIABLES or
GETFV
FORM_NAME=data_name
VARIABLE_NAME=data_name
VALUE=any variable
OCCURRENCE=integer
STATUS=status variable

Parameters

FORM_NAME or **FN**

The name of the form where the variable resides. This parameter is required.

VARIABLE_NAME or **VN**

The name of the variable on the form to get and transfer to SCL. This name was defined when the form was created. This parameter is required.

VALUE or **V**

The variable that is to hold the value Screen Formatting gets from the form. This variable is either created automatically when the form is opened or created manually. This parameter is required.

See the VARIABLE_CREATION parameter on the MANAGE_FORM command for more information.

OCCURRENCE or *O*

The occurrence of the variable name. The values allowed are 1 through 1000. Use 1 for the first or only occurrence. The default is 1.

Remarks

- Before you get a variable, you must open its form. If you get the variable after opening the form and before reading or replacing the variable on the form, the utility returns the initial value specified by the form designer.
- If the form designer specifies data validation rules and error processing to display an error message or form, you do not need to look at the STATUS parameter.

  If the form designer specifies data validation rules and no error processing, you must look at the STATUS parameter.

  If the form designer specifies no data validation rules, you must look at the STATUS parameter to determine if the subcommand executed properly.
- For more information, see the NOS/VE Screen Formatting manual.

OPEN_FORM MANF Subcommand

Purpose OPEN_FORM locates a form and prepares it for use by the utility.

Format

OPEN_FORM or
OPEF
FORM_NAME=data_name
STATUS=status variable

Parameters

FORM_NAME or **FN**

The name of the form you want to open. This parameter is required.

Remarks

- When you open forms, Screen Formatting creates SCL variables for form variables. The scope of the variables is LOCAL. Before creating a variable, Screen Formatting checks to see if the variable already exists. If it does, Screen Formatting does not try to create it again.

- Screen Formatting locates a form by searching the command library list to find the form name on the object libraries. (You specify the order in which Screen Formatting searches the list using the NOS/VE command CREATE_COMMAND_LIST_ENTRY).
- Executing OPEN_FORM does not display the form on the screen. (See ADD_FORM, READ_FORMS, or SHOW_FORMS.)
- For more information, see the NOS/VE Screen Formatting manual.

POP_FORM
MANF Subcommand

Purpose POP_FORMS deletes forms scheduled (added or combined) since the last PUSH_FORMS call.

Format

POP_FORM or
POP_FORMS or
POPF
STATUS=status variable

Remarks

- Events associated with the last list of pushed forms become active.
- For more information, see the NOS/VE Screen Formatting manual.

POSITION_FORM
MANF Subcommand

Purpose POSITION_FORM schedules moving a form to a new location. Using this subcommand, you can define a form at one location and display it at another location, or you can move a form from where it is currently displayed to a new location.

Format

POSITION_FORM or
POSF
FORM_NAME=data_name
X_POSITION=integer
Y_POSITION=integer
STATUS=status variable

Parameters

FORM_NAME or FN

The form name established when the form was opened. This parameter is required.

X_POSITION or *XP*

The x position on the screen. The character position in the upper left corner of the screen is 1, and the x position increases by 1 for each character counting from left to right. The default is 1.

Y_POSITION or *YP*

The y position on the screen. The character position in the upper left corner of the screen is 1, and the y position increases by 1 for each character counting from top to bottom. The default is 1.

Remarks

- When you enter either the READ_FORMS or SHOW_FORMS subcommand, Screen Formatting displays the form on the screen at the position specified in the POSITION_FORM subcommand.
- If you enter this subcommand while the form is displayed, the form is deleted from its current location and added at the new location. The added form is displayed on top of any other form occupying the same area on the screen.
- If you enter this subcommand before the form is displayed, the form is displayed at the specified location.
- Before you position a form, you must open it.
- You cannot position a pushed form.
- For more information, see the NOS/VE Screen Formatting manual.

PUSH_FORM
MANF Subcommand

Purpose PUSH_FORMS causes Screen Formatting to record added and combined forms so you can return to them later.

Format

PUSH_FORM or
PUSH_FORMS or
PUSF
STATUS=status variable

Remarks

- Events associated with these forms are not passed to you.
- You cannot change or close a pushed form.
- Pushed forms are displayed on the screen. If you want newly added forms to appear on a blank screen, first add a blank form that covers the screen.

  Updates to the screen continue to show the pushed forms.
- This subcommand deactivates the events associated with forms scheduled for display (added or combined) since the last PUSH_FORMS subcommand.
- For more information, see the NOS/VE Screen Formatting manual.

QUIT
MANF Subcommand

Purpose QUIT ends the MANAGE_FORMS utility session.

Format

QUIT or
QUI
STATUS=status variable

Remarks

- All open forms are closed and the resources used by Screen Formatting are returned to NOS/VE.
- For more information, see the NOS/VE Screen Formatting manual.

READ_FORM
MANF Subcommand

Purpose READ_FORMS updates the terminal screen and accepts input from the application user.

Format

READ_FORM or
READ_FORMS or
REAF
STATUS=status variable

Remarks

- Executing READ_FORMS:
  - Displays all the forms you scheduled for display and have not deleted. If you added or combined forms since the last READ_FORMS or SHOW_FORMS subcommand, it displays them for the first time.
  - Removes from the screen the forms you deleted since the last READ_FORMS or SHOW_FORMS subcommand.
  - Updates on the screen the variables replaced since the last READ_FORMS or SHOW_FORMS call subcommand.
  - Updates on the screen the objects for which display attributes were set or reset since the last READ_FORMS or SHOW_FORMS subcommand.
- Events not retrieved with the $EVENT_NAME function are deleted before any input is accepted from the user.
- The READ_FORMS subcommand does not execute unless the forms scheduled for display contain at least one active event.
- After issuing this request, you do not regain control until the user issues a normal event and Screen Formatting validates all the data, or the user issues an abnormal event.
- For more information, see the NOS/VE Screen Formatting manual.

REPLACE_FORM_VARIABLE MANF Subcommand

Purpose REPLACE_FORM_VARIABLE transfers an SCL variable to Screen Formatting.

Format

REPLACE_FORM_VARIABLE or
REPLACE_FORM_VARIABLES or
REPFV
FORM_NAME=data_name
VARIABLE_NAME=data_name
VALUE=any
OCCURRENCE=*integer*
STATUS=*status variable*

Parameters

FORM_NAME or **FN**

The name of the form where the variable resides. This parameter is required.

VARIABLE_NAME or **VN**

The name of the variable to replace. This name was defined when the form was created. This parameter is required.

VALUE or **V**

The value Screen Formatting will replace on the form. This parameter is required.

For more information, see the VARIABLE_CREATION parameter on the MANAGE_FORM command.

OCCURRENCE or *O*

The occurrence of the variable name. The values allowed are 1 through 1000. Use 1 for the first or only occurrence.

Remarks

- When you execute either the READ_FORMS or SHOW_FORMS subcommand, Screen Formatting replaces the variable on the terminal screen.
- Before you replace a variable, you must open the form on which it is replaced.
- You cannot replace a variable for a pushed form.
- If the value is not valid, it is not replaced.

- For more information, see the NOS/VE Screen Formatting manual.

RESET_FORM MANF Subcommand

Purpose RESET_FORM resets the form to the state specified by the form definition.

Format

RESET_FORM or
RESF
FORM_NAME=data_name
STATUS=status variable

Parameters

FORM_NAME or **FN**

The name of the form to reset. This parameter is required.

Remarks

- When you execute either the READ_FORMS or SHOW_FORMS subcommand, Screen Formatting displays the form on the terminal screen with the reset specifications.
- All variables belonging to the form have their initial values and display attributes. The form is in its defined position.
- Before you reset a form, you must open it.
- You cannot reset a pushed form.
- For more information, see the NOS/VE Screen Formatting manual.

SET_CURSOR_POSITION MANF Subcommand

Purpose SET_CURSOR_POSITION sets the cursor to a selected position for later display.

Format

SET_CURSOR_POSITION or
SETCP
FORM_NAME=*data_name*
OBJECT_NAME=*data_name*
OCCURRENCE=*integer*
CHARACTER_POSITION=*integer*
STATUS=*status variable*

Parameters

FORM_NAME or **FN**

The name established when the form was opened. This parameter is required.

OBJECT_NAME or **ON**

The name of the object on which you want to set the cursor. This name was defined when the form was created. This parameter is required.

OCCURRENCE or *O*

The integer specifying the occurrence of the object name. Use 1 for the first occurrence. The default is 1.

CHARACTER_POSITION or *CP*

The character position to which you want to set the cursor. Use 1 for the first character position. The default is 1.

Remarks

- One use of this subcommand is to alter the default sequence of the application user's entry of variables. In the default sequence, Screen Formatting places the cursor on the first input variable of the highest priority form. The highest priority form is the form last added, combined, or positioned.

  At terminals with protected fields, the user then tabs from one variable text object to the next. The cursor starts at the top line of the form. It moves from left to right on each line. When no variable text object appears on a line, the cursor moves down to the next line. At terminals without protected fields, the user must move the cursor using the arrow keys or use the tab and return keys.

- When you execute either the READ_FORMS or SHOW_FORMS subcommand, Screen Formatting updates the terminal screen with the cursor at the specified position.
- If the position you specify is not visible on the screen, Screen Formatting shifts the data to make the cursor visible.
- The cursor position is in effect only for the next screen update from reading or showing forms.
- Before you set the cursor position on a form, you must open the form and either add or combine it.
- You cannot set the cursor position in a pushed form.
- For more information, see the NOS/VE Screen Formatting manual.

SET_OBJECT_ATTRIBUTE MANF Subcommand

Purpose SET_OBJECT_ATTRIBUTE changes a display attribute for an object.

Format

SET_OBJECT_ATTRIBUTE or
SET_OBJECT_ATTRIBUTES or
SETOA
FORM_NAME=data_name
OBJECT_NAME=data_name
ATTRIBUTE=keyword or *data_name*
OCCURRENCE=integer
STATUS=status variable

Parameters

FORM_NAME or FN

The name of the form containing the object. This parameter is required.

OBJECT_NAME or ON

The name of the object whose display attribute is being reset. This parameter is required.

ATTRIBUTE or *A*

The name given the display attribute being set when the attribute was defined on the form. The attribute used here is defined for the form and not for a specific object. When using Screen Design Facility, screen attributes are defined through the ATTRIB function key. When using a CYBIL program, the ADD_DISPLAY_DEFINITION attribute record defines form attributes.

Specifying the keyword INITIAL, resets the object to the attribute defined in the form definition. The default is INITIAL.

OCCURRENCE or *O*

The occurrence of the object. For the first or only occurrence, use 1. The default is 1.

Remarks

- You can set the attributes of objects that are variable text, constant text, lines, or boxes.
- Changed attributes replace existing attributes.
- When you execute either the READ_FORMS or SHOW_FORMS subcommand, Screen Formatting displays the object using the set attributes.
- If the object you specify is not visible on the screen, Screen Formatting shifts the data to make the object visible.
- Before you set the attribute of an object, you must open the form the object is on and either add or combine it.
- You cannot set attributes of objects on a pushed form.
- For more information, see the NOS/VE Screen Formatting manual.

SHOW_FORM
MANF Subcommand

Purpose SHOW_FORMS updates the terminal screen.

Format

SHOW_FORM or
SHOW_FORMS or
SHOF
STATUS=status variable

Remarks

- When none of the forms scheduled for display has an event or input variable defined, use this subcommand instead of READ_FORMS.
- When you do not want any input from the terminal user, use this subcommand.
- Executing SHOW_FORMS:
  - Displays all the forms you have scheduled for display and have not deleted. If you added or combined forms since the last READ_FORMS or SHOW_FORMS subcommand, it displays them for the first time.
  - Removes from the screen the forms you deleted since the last READ_FORMS or SHOW_FORMS subcommand.
  - Displays variables replaced since the last READ_FORMS or SHOW_FORMS subcommand.
  - Displays objects with attributes set or reset since the last READ_FORMS or SHOW_FORMS subcommand.
- For more information, see the NOS/VE Screen Formatting manual.

MANAGE_JOB 24

MANAGE_JOB Command

Purpose Initiates the utility that manages the selection and control of one or more jobs.

Format

MANAGE_JOB or
MANAGE_JOBS or
MANJ
STATUS=status variable

Remarks For more information, see the NOS/VE System Usage manual.

QUIT MANJ Subcommand

Purpose Ends a MANAGE_JOB utility session.

Format

QUIT or
QUI
STATUS=status variable

SELECT_JOB MANJ Subcommand

Purpose Selects one or more jobs matching criteria specified on this subcommand.

Format

SELECT_JOB or
SELECT_JOBS or
SELJ
CONTROL_FAMILY=list of name
CONTROL_USER=list of name
JOB_CATEGORY_NAME=list of name
JOB_CLASS=list of name
JOB_DEFERRED_BY_OPERATOR=boolean
JOB_DEFERRED_BY_USER=boolean
JOB_QUALIFIER=list of name
JOB_STATE=keyword or *list of keyword*
LOGIN_ACCOUNT=list of name
LOGIN_FAMILY=list of name

LOGIN_PROJECT=list of name
LOGIN_USER=list of name
NAME=list of name
SITE_INFORMATION=list of string
USER_INFORMATION=list of string
MAXIMUM_SELECTION=keyword or *integer*
JOB_SELECTION_LIST=list of name variable
STATUS=status variable

Parameters

CONTROL_FAMILY or *CONTROL_FAMILIES* or *CF*

Specifies the control family names of the jobs to be selected. If omitted, this parameter is ignored.

CONTROL_USER or *CONTROL_USERS* or *CU*

Specifies the control user names of the jobs to be selected. If omitted, this parameter is ignored.

JOB_CATEGORY_NAME or *JOB_CATEGORY_NAMES* or *JCN*

Specifies the categories to which the selected jobs must belong. See your site personnel for information on the categories defined at your site. If omitted, this parameter is ignored.

JOB_CLASS or *JOB_CLASSES* or *JC*

Specifies the job classes to which the selected jobs must belong. See your site personnel for information on the job classes defined at your site. If omitted, this parameter is ignored.

JOB_DEFERRED_BY_OPERATOR or *JDBO*

Specifies the operator-controlled scheduling state that the selected jobs must have. Options are:

TRUE

The operator has deferred the execution of the job.

FALSE

The operator has not deferred execution of the job.

If omitted, this parameter is ignored.

JOB_DEFERRED_BY_USER or *JDBU*

Specifies the user controlled scheduling state that the selected jobs must have. Options are:

TRUE

Execution of the job has been deferred by the user.

FALSE

Execution of the job has not been deferred by the user.

If omitted, this parameter is ignored.

JOB_QUALIFIER or *JOB_QUALIFIERS* or *JQ*

Specifies the job qualifiers the selected jobs must have. See your site personnel for a list of the job qualifiers at your site. If omitted, this parameter is ignored.

JOB_STATE or *JOB_STATES* or *JS*

Specifies the state the selected jobs must have. Options are:

ALL

Uses all of the following keyword values.

DEFERRED (D)

Specifies jobs not yet eligible to be initiated.

QUEUED (Q)

Specifies jobs waiting to be initiated.

INITIATED (I)

Specifies jobs that have been initiated.

TERMINATED (T)

Specifies jobs that are terminating.

If omitted, this parameter is ignored.

LOGIN_ACCOUNT or *LOGIN_ACCOUNTS* or *LA*

Specifies the login accounts of the jobs to be selected. If omitted, this parameter is ignored.

LOGIN_FAMILY or *LOGIN_FAMILIES* or *LF*

Specifies the login families of the jobs to be selected. If omitted, this parameter is ignored.

LOGIN_PROJECT or *LOGIN_PROJECTS* or *LP*

Specifies the login projects of the jobs to be selected. If omitted, this parameter is ignored.

LOGIN_USER or *LOGIN_USERS* or *LU*

Specifies the login users of the jobs to be selected. If omitted, this parameter is ignored.

NAME or *NAMES* or *N*

Specifies the names of the jobs to be selected. The names may be either the system-supplied or the user-supplied job names. If omitted, this parameter is ignored.

SITE_INFORMATION or *SI*

Specifies the site information strings of the jobs to be selected. If omitted, this parameter is ignored.

USER_INFORMATION or *UI*

Specifies the user information strings of the jobs to be selected. If omitted, this parameter is ignored.

MAXIMUM_SELECTION or *MS*

Specifies the maximum number of jobs that can be selected. If omitted, the number of jobs meeting the criteria specified on this subcommand is used.

JOB_SELECTION_LIST or *JSL*

Specifies the variable that will contain the system job names of the selected jobs. This variable has a LOCAL scope. If omitted, JMV$SELECTED_JOBS is used.

Remarks

- If all selection parameters are omitted, all jobs for which you are login user, control user, or parent job will be returned. The system operator can select all jobs.
- For more information, see the NOS/VE System Usage manual.

MANAGE_OUTPUT 25

MANAGE_OUTPUT Command

Purpose Initiates the utility that manages the selection and control of one or more output files in the system.

Format

MANAGE_OUTPUT or
MANO
STATUS=status variable

Remarks For more information, see the NOS/VE System Usage manual.

QUIT MANO Subcommand

Purpose Ends a MANAGE_OUTPUT utility session.

Format

QUIT or
QUI
STATUS=status variable

SELECT_OUTPUT MANO Subcommand

Purpose Selects one or more output files matching criteria specified on this subcommand.

Format

SELECT_OUTPUT or
SELO
COMMENT_BANNER=list of string
CONTROL_FAMILY=list of name
CONTROL_USER=list of name
DATA_MODE=list of keyword
DEVICE=list of: keyword or *name*
EXTERNAL_CHARACTERISTICS=list of: keyword or *string*
FORMS_CODE=list of: keyword or *string*
LOGIN_ACCOUNT=list of name
LOGIN_FAMILY=list of name
LOGIN_PROJECT=list of name
LOGIN_USER=list of name

NAME=list of name
OPERATOR_FAMILY=list of name
OPERATOR_USER=list of name
OUTPUT_CLASS=list of name
OUTPUT_DEFERRED_BY_OPERATOR=boolean
OUTPUT_DEFERRED_BY_USER=boolean
OUTPUT_DESTINATION=list of: name or *string*
OUTPUT_DESTINATION_USAGE=list of: keyword or *name*
OUTPUT_PRIORITY=list of name
OUTPUT_STATE=keyword or *list of keyword*
REMOTE_HOST_DIRECTIVE=list of string
ROUTING_BANNER=list of string
SITE_INFORMATION=list of string
STATION=list of: keyword or *name*
SYSTEM_JOB_NAME=list of name
USER_INFORMATION=list of string
VERTICAL_PRINT_DENSITY=list of keyword
VFU_LOAD_PROCEDURE=list of: keyword or *name*
MAXIMUM_SELECTION=keyword or *integer*
OUTPUT_SELECTION_LIST=list of name variable
STATUS=status variable

Parameters

COMMENT_BANNER or *COMMENT_BANNERS* or *CB*

Specifies the comment banner strings of the output files to be selected. If omitted, this parameter is ignored.

CONTROL_FAMILY or *CONTROL_FAMILIES* or *CF*

Specifies the control family names of the output files to be selected. If omitted, this parameter is ignored.

CONTROL_USER or *CONTROL_USERS* or *CU*

Specifies the control user names of the output files to be selected. If omitted, this parameter is ignored.

DATA_MODE or *DM*

Specifies the data modes of the output files to be selected. Options are:

CODED (C)

Specifies that the output files with a DATA_MODE attribute value of CODED are to be selected.

TRANSPARENT (T)

Specifies that the output files with a DATA_MODE attribute value of TRANSPARENT are to be selected.

If omitted, this parameter is ignored.

DEVICE or *DEVICES* or *D*

Specifies the device names of the output files to be selected. The keyword AUTOMATIC selects the output files given the DEVICE attribute value of AUTOMATIC. If omitted, this parameter is ignored.

EXTERNAL_CHARACTERISTICS or *EC*

Specifies the external characteristic strings of the output files to be selected. If the keyword NORMAL is specified, the output files with a EXTERNAL_CHARACTERISTICS attribute value of NORMAL are selected. If omitted, this parameter is ignored.

FORMS_CODE or *FORMS_CODES* or *FC*

Specifies the forms code strings of the output files to be selected. If the keyword NORMAL is specified, the output files with a FORMS_CODE attribute value of NORMAL are selected. If omitted, this parameter is ignored.

LOGIN_ACCOUNT or *LOGIN_ACCOUNTS* or *LA*

Specifies the login accounts of the output files to be selected. If omitted, this parameter is ignored.

LOGIN_FAMILY or *LOGIN_FAMILIES* or *LF*

Specifies the login families of the output files to be selected. If omitted, this parameter is ignored.

LOGIN_PROJECT or *LOGIN_PROJECTS* or *LP*

Specifies the login projects of the output files to be selected. If omitted, this parameter is ignored.

LOGIN_USER or *LOGIN_USERS* or *LU*

Specifies the login users of the output files to be selected. If omitted, this parameter is ignored.

NAME or *NAMES* or *N*

Specifies the names of the output files to be selected. The names may be either the system-supplied or the user-supplied file names. If omitted, this parameter is ignored.

OPERATOR_FAMILY or *OPERATOR_FAMILIES* or *OF*

Specifies the default private stations or remote system operator family names for the output files to be selected. If omitted, this parameter is ignored.

OPERATOR_USER or *OPERATOR_USERS* or *OU*

Specifies the operator user names of the output files to be selected. If omitted, this parameter is ignored.

OUTPUT_CLASS or *OUTPUT_CLASSES* or *OC*

Specifies the output classes of the files to be selected. If omitted, this parameter is ignored.

OUTPUT_DEFERRED_BY_OPERATOR or *ODBO*

Specifies the operator controlled scheduling state that the selected output files must have. Options are:

TRUE

The operator has deferred the printing of the files.

FALSE

The operator has not deferred the printing of the files.

If omitted, this parameter is ignored.

OUTPUT_DEFERRED_BY_USER or *ODBU*

Specifies the user controlled scheduling state that the selected output files must have. Options are:

TRUE

Printing of the files has been deferred by the user.

FALSE

Printing of the files has not been deferred by the user.

If omitted, this parameter is ignored.

OUTPUT_DESTINATION or *OUTPUT_DESTINATIONS* or *ODE*

Specifies the output destinations of the output files to be selected. If omitted, this parameter is ignored.

OUTPUT_DESTINATION_USAGE or *ODU*

Specifies the OUTPUT_DESTINATION_USAGE attribute values of the output files to be selected. Options are:

PUBLIC

Indicates that the file is to be printed at a public CDCNET batch I/O station.

PRIVATE

Indicates that the file is to be printed at a private CDCNET batch I/O station when the designated station operator is controlling the station.

DUAL_STATE

Indicates that the file is to be printed under control of the NOS or NOS/BE system that shares the mainframe.

QTF

Indicates that the file is to be forwarded to a remote RHF system for printing by that system.

NTF

Indicates that the file is to be forwarded to a remote NTF system for printing by that system.

If omitted, this parameter is ignored.

OUTPUT_PRIORITY or *OUTPUT_PRIORITIES* or *OP*

Specifies the default increments that is added to the initial priority (defined by the output class) of the output files to be selected. Options are:

Keyword	Increment
LOW	0
MEDIUM	1500
HIGH	3000

If omitted, this parameter is ignored.

OUTPUT_STATE or *OUTPUT_STATES* or *OS*

Specifies the state the selected output files must have. Options are:

ALL

Uses all of the following keyword values.

DEFERRED (D)

Output files not yet eligible to be printed.

QUEUED (Q)

Output files waiting to be printed.

INITIATED (I)

Output files that have been printed.

TERMINATED (T)

Output files that are terminating.

COMPLETED (C)

Output files that have finished printing and are waiting for their purge delay to expire.

If omitted, this parameter is ignored.

REMOTE_HOST_DIRECTIVE or
REMOTE_HOST_DIRECTIVES or *RHD*

Specifies the remote host directive strings of the output files to be selected. If omitted, this parameter is ignored.

ROUTING_BANNER or *ROUTING_BANNERS* or *RB*

Specifies the routing banner strings of the output files to be selected. If omitted, this parameter is ignored.

SITE_INFORMATION or *SI*

Specifies the site information strings of the output files to be selected. If omitted, this parameter is ignored.

STATION or *STATIONS* or *S*

Specifies the I/O station names (or the control facility names in the case of a private station or NTF remote system) of the output files to be selected. If specified, the

keyword AUTOMATIC causes the output files with a STATION attribute value of AUTOMATIC to be selected. If omitted, this parameter is ignored.

SYSTEM_JOB_NAME or *SYSTEM_JOB_NAMES* or *SJN*

Specifies the system job names of the jobs that created the output files to be selected. If omitted, this parameter is ignored.

USER_INFORMATION or *UI*

Specifies the user information strings of the jobs that created the output files to be selected. If omitted, this parameter is ignored.

VERTICAL_PRINT_DENSITY or *VERTICAL_PRINT_DENSITIES* or *VPD*

Specifies the vertical print densities of the output files to be selected. Options are:

SIX

Selects a printer to print at six lines-per-inch.

EIGHT

Selects a printer to print at eight lines-per-inch.

NONE

Vertical print density is not used to select a printer.

If omitted, this parameter is ignored.

VFU_LOAD_PROCEDURE or *VFU_LOAD_PROCEDURES* or *VLP*

Specifies the vfu load procedure files of the output files to be selected. If specified, the keyword value NONE causes the output files with the VFU_LOAD_PROCEDURE attribute value of NONE to be selected. If omitted, this parameter is ignored.

MAXIMUM_SELECTION or *MS*

Specifies the maximum number of output files that can be selected. If omitted, the number of output files meeting the criteria specified on this subcommand is used.

OUTPUT_SELECTION_LIST or *OSL*

Specifies the variable that will contain the system output file names of the selected output files. This variable has a LOCAL scope. If omitted, JMV$SELECTED_OUTPUT is used.

Remarks

- If no selection parameters are supplied, all output files for which the caller of this command is the login user or control user are returned.
- For more information, see the NOS/VE System Usage manual.

MANAGE _REMOTE _FILES 26

MANAGE_REMOTE_FILE Command

Purpose Delimits a set of commands to be executed on the specified remote system.

Format

```
MANAGE_REMOTE_FILE or
MANAGE_REMOTE_FILES or
MANRF or
MFLINK
    LOCATION=string or name
    FILE=file
    DATA_DECLARATION=keyword
    UNTIL=string
    SUBSTITUTION_MARK=string or keyword
    IGNORE_REMOTE_VALIDATION=boolean
    STATUS=status variable
```

Parameters

LOCATION or L

Specifies the name of the remote location to be accessed. This is a name associated with a remote system, such as a family name or a logical identifier. (Location names are determined by your network application administrator.)

This parameter must be a string or name value.

You cannot specify a variable name for this parameter. If you want to use a variable that has a name value, you can use the $NAME function instead.

This parameter is required.

FILE or F

Specifies the name of a file on the local NOS/VE system to be used as the input or output file during a file transfer. This parameter is required even when you are not performing a file transfer.

DATA_DECLARATION or *DD*

Specifies the data format of a file to be transferred.

If the remote location is another NOS/VE host, this parameter is ignored. The rules for copying NOS/VE files based on the local and remote file attributes apply. For a discussion of rules for copying NOS/VE files, see the NOS/VE System Usage manual.

If the remote location is a non-NOS/VE host, the following data descriptions are available. The meaning of each varies among the various remote host types. Refer to the Remote Host Facility Usage manual for system specific information.

- C6

  Use this format when you transfer files to hosts using a 6-bit code set. This format indicates the file contains character data from a character set with 64 or fewer character codes.

  The effect of this format is that each machine sees the file in its native character set. Thus, if you transfer the file from NOS/VE to NOS, NOS/VE sends the file in ASCII and NOS receives it in display code. Transfers to other systems result in full ASCII transfers as if DD=C8 was used.

- C8

  This format has the following meanings depending upon which system the file is being transferred to:

  - NOS

    Transfer results in a NOS 8/12 ASCII file. Use the NOS FCOPY command to convert the file to NOS 6/12 format.

  - NOS/BE

    Same as for NOS.

  - Any other ASCII system

    Transfer results in an ASCII file.

  - IBM/MVS

    Transfer results in an EBCDIC file.

- UU

  Use this format to transfer binary files to remote systems. Object and source libraries should be transferred using this format. Files transferred to NOS or NOS/BE will be padded unless they end on a 120-bit boundary (this is because NOS and NOS/BE store their files in 60-bit format). Similarly, files transferred from NOS or NOS/BE to NOS/VE and that have a file length that is an odd multiple of 60 bits will be padded to the next full byte (8-bit) length.

UNTIL or *U*

Specifies the string indicating the end of commands in the list. The string must appear on a separate line. If this parameter is omitted, a string of two asterisks (**) is assumed.

SUBSTITUTION_MARK or *SM*

Specifies a character used to delimit text to be substituted within the command text following the MANAGE_REMOTE_FILES command. Values can be any character or the keyword NONE. NONE specifies that no substitution mark is to be used. If this parameter is omitted, NONE is assumed.

IGNORE_REMOTE_VALIDATION or *IRV*

Specifies whether to use validation information provided on a previous CREATE_REMOTE_VALIDATION command. Options are:

TRUE

Ignores the validation information provided on a previous CREATE_REMOTE_VALIDATION command. You must provide new validation information for the remote system in the command text following the MANAGE_REMOTE_FILE command.

FALSE

Uses the validation information provided on a previous CREATE_REMOTE_VALIDATION command on the current command.

If omitted, FALSE is assumed.

Remarks

- You must provide validation information required by the remote system. As an alternative, you can issue a CREATE_REMOTE_VALIDATION command prior to using the MANAGE_REMOTE_FILES command. If this remote system is NOS/VE, the first command in the list of commands must be a LOGIN command.

- The names and parameters of commands accepted by each remote system type are described in the Remote Host Facility Usage manual.

  The MANAGE_REMOTE_FILES command passes the command text you supply to the remote system for execution. If the remote system is NOS/VE, the command text is a set of SCL commands to be executed as a batch job.

- You can include at most one remote command in the command text which causes an explicit file transfer. For remote NOS/VE systems, use the SEND_FILE or RECEIVE_FILE commands to explicitly transfer a file.

- For more information, see the NOS/VE System Usage manual.

RECEIVE_FILE MANRF Subcommand

Purpose

When used within the list of commands delimited by the MANAGE_REMOTE_FILES command, transfers a file from your local system to a remote system.

Format

RECEIVE_FILE or
RECF
FILE=file
STATUS=status variable

Parameters

FILE or **F**

Specifies the name of the file on the remote system that is to receive the file from your local system.

Remarks

- You can use the RECEIVE_FILE command only with the MANAGE_REMOTE_FILES command. (Refer to the MANAGE_REMOTE_FILES command.)
- Refer to the SEND_FILE command for information about transferring files from a remote system to your local system.
- For more information, see the NOS/VE System Usage manual.

SEND_FILE MANRF Subcommand

Purpose

When used within the list of commands delimited by the MANAGE_REMOTE_FILES command, sends a file from a remote system to your local system.

Format

SEND_FILE or
SENF
FILE=file
STATUS=status variable

Parameters

FILE or **F**

Specifies the name of the file on the remote system that is to be sent to your local system.

Remarks

- You can use the SEND_FILE command only with the MANAGE_REMOTE_FILES command. (Refer to the MANAGE_REMOTE_FILES command.)
- Refer to the RECEIVE_FILE command for information about transferring files from your local system to a remote system.
- For more information, see the NOS/VE System Usage manual.

MEASURE_PROGRAM_EXECUTION 27

MEASURE_PROGRAM_EXECUTION Command

Purpose Starts a program measurement utility session.

Format

MEASURE_PROGRAM_EXECUTION or
MEAPE
STATUS=status variable

Remarks

- The session ends when you enter the subcommand QUIT. The descriptions of each program measurement subcommand follow this description.
- For more information, see the NOS/VE Object Code Management manual.

Examples The following utility session specifies the program as the modules on file LGO, executes the program, and saves the program profile on file MY_FILE.

```
/measure_program_execution
MPE/set_program_description target_text=lgo
MPE/execute_instrumented_task
MPE/display_program_profile output=my_file
MPE/quit
/
```

CREATE_RESTRUCTURED_MODULE MEAPE Subcommand

Purpose Generates a restructuring procedure and executes the procedure to create a restructured load module on an object library. It can also save the restructuring procedure.

Format

CREATE_RESTRUCTURED_MODULE or
CRERM
RESTRUCTURED_MODULE=file
RESTRUCTURED_MODULE_NAME=any
RESTRUCTURING_COMMANDS=file
STATUS=status variable

Parameters **RESTRUCTURED_MODULE or RM**

File to which the object library containing the new load module is written. This parameter is required.

RESTRUCTURED_MODULE_NAME or RMN

Name given the new load module. If RESTRUCTURED_MODULE_NAME is omitted, the module name is the same as the file name.

RESTRUCTURING_COMMANDS or RC

File on which the restructuring procedure is saved. If RESTRUCTURING_COMMANDS is omitted, the restructuring procedure is discarded.

Remarks For more information, see the NOS/VE Object Code Management manual.

27

CREATE_RESTRUCTURING_COMMANDS MEAPE Subcommand

Purpose Generates and saves a restructuring procedure.

Format

CREATE_RESTRUCTURING_COMMANDS or CRERC
RESTRUCTURING_COMMANDS=file
RESTRUCTURED_MODULE=file
RESTRUCTURED_MODULE_NAME=any
STATUS=status variable

Parameters **RESTRUCTURING_COMMANDS or RC**

File to which the procedure is written. The procedure name is the same as the file name. This parameter is required.

RESTRUCTURED_MODULE or RM

Object library file to which the restructured module is written after the restructuring procedure is executed. This parameter is required.

RESTRUCTURED_MODULE_NAME or RMN

Name to be given the module created when the restructuring procedure is executed. If the RESTRUCTURED_MODULE_NAME parameter is omitted, the module name is the same as the file name.

Remarks

- The CREATE_RESTRUCTURING_COMMANDS subcommand uses the information accumulated in the connectivity matrix to generate the restructuring procedure.
- The CREATE_RESTRUCTURING_COMMANDS subcommand does not execute the restructuring procedure. To generate a restructured module, either enter a CREATE_RESTRUCTURED_MODULE subcommand during the session or execute the saved restructuring procedure.
- For more information, see the NOS/VE Object Code Management manual.

Examples

The following subcommand writes a restructuring procedure on file MODULE_RESTRUCTURE. If the procedure is executed, it creates a module named NEWLGO on object library file $USER.NEWLGO.

```
MPE/create_restructuring_commands restructuring_..
MPE../commands=module_restructure restructured_..
MPE../module=$user.newlgo
```

For an example of a restructuring procedure, refer to the BIND_MODULE subcommand description.

DISPLAY_PROGRAM_PROFILE MEAPE Subcommand

Purpose

Generates and displays a program profile. The program profile uses the execution time totals accumulated by previous EXECUTE_INSTRUMENTED_TASK subcommands.

Format

DISPLAY_PROGRAM_PROFILE or
DISPP
PROFILE_ORDER=keyword
PROGRAM_UNIT_CLASS=keyword
NUMBER=keyword or *integer*
OUTPUT=file
STATUS=status variable

Parameters *PROFILE_ORDER* or *PO*

Order in which the program profile is displayed. Options are:

TIME (T)

By percentage of the total execution time ordered greatest to least.

PROGRAM_UNIT (PU)

By program unit name ordered alphabetically.

MODULE_PROGRAM_UNIT (MPU)

By module name ordered alphabetically.

If PROFILE_ORDER is omitted, TIME is used.

PROGRAM_UNIT_CLASS or *PUC*

Class of program units whose statistics are displayed. Options are:

ALL

All program units measured, both local and remote.

LOCAL

Only program units that are part of the target text.

REMOTE

Only program units that are called by target text program units, but are not part of the target text. These program units provide the remote block statistics in the program profile.

If PROGRAM_UNIT_CLASS is omitted, ALL is used.

NUMBER or *N*

Number of program unit statistics displayed. The statistics are sorted as specified by the PROFILE_ORDER parameter and then displayed in order until the specified number of statistics have been displayed. If NUMBER is omitted, the entire program profile is displayed.

OUTPUT or *O*

File to which the display is written. This file can be positioned. If OUTPUT is omitted, file $OUTPUT is used.

Remarks For more information, see the NOS/VE Object Code Management manual.

EXECUTE_INSTRUMENTED_TASK MEAPE Subcommand

Purpose Executes and measures the performance of the last program specified by a SAVE_PROGRAM_MEASURES or SET_PROGRAM_DESCRIPTION subcommand.

Format

EXECUTE_INSTRUMENTED_TASK or
EXEIT
PARAMETER=string
NO_CONNECTIVITY_MATRIX=boolean
WORKING_SET_INTERVAL=integer
STATUS=status variable

Parameters *PARAMETER* or *P*

Parameter string passed to the program.

NO_CONNECTIVITY_MATRIX or *NCM*

Indicates whether a connectivity matrix is generated.

NOTE

Specify NO_CONNECTIVITY_MATRIX=TRUE if you do not intend to generate a restructuring procedure for the program. Omitting generation of a connectivity matrix saves time and system resources.

TRUE

No connectivity matrix is generated.

FALSE

A connectivity matrix is generated.

If NO_CONNECTIVITY_MATRIX is omitted, FALSE is assumed and a connectivity matrix is generated.

WORKING_SET_INTERVAL or *WSI*

Reserved.

Remarks

- The program is executed once for each EXECUTE_INSTRUMENTED_TASK subcommand you enter. You can specify a different parameter list for each execution. Cumulative statistics for all executions are kept.
- For more information, see the NOS/VE Object Code Management manual.

Examples The following sequence executes the modules on file LGO twice; cumulative statistics are kept for the program executions. The program profile is saved on file $USER.PROFILE_LIST.

```
/measure_program_execution
MPE/set_program_description target_text=lgo
MPE/execute_instrumented_task parameter='size=40' ..
MPE../no_connectivity_matrix=true
MPE/execute_instrumented_task parameter='size=400' ..
MPE../no_connectivity_matrix=true
MPE/display_program_profile output=$user.profile_list
MPE/quit
/
```

QUIT
MEAPE Subcommand

Purpose Ends a MEASURE_PROGRAM_EXECUTION utility session.

Format **QUIT** or
QUI

Parameters None.

Remarks For more information, see the NOS/VE Object Code Management manual.

RESTORE_PROGRAM_MEASURES
MEAPE Subcommand

Purpose Restores the program measurement environment using the information saved by a SAVE_PROGRAM_MEASURES subcommand.

Format

RESTORE_PROGRAM_MEASURES or
RESPM
MEASURES=file
STATUS=status variable

Parameters

MEASURES or **M**

File containing a saved program measurement environment. This parameter is required.

Remarks

- The RESTORE_PROGRAM_MEASURES subcommand always restores the program description. It also restores the execution time statistics and connectivity matrix if that information was saved on the file.
- For more information, see the NOS/VE Object Code Management manual.

Examples

The following sequence begins a program measurement session and restores the program measurement environment saved on file SAVED_MEASUREMENT.

```
/measure_program_execution
MPE/restore_program_measures measures=..
MPE../saved_measurement
MPE/
```

SAVE_PROGRAM_MEASURES MEAPE Subcommand

Purpose

Saves the current program measurement environment on a file.

Format

SAVE_PROGRAM_MEASURES or
SAVPM
MEASURES=file
AMOUNT=list of keyword
STATUS=status variable

Parameters

MEASURES or M

File on which the program measurement environment is saved. This parameter is required.

AMOUNT or A

Information to be saved. Options are:

ALL

Program description, connectivity matrix, and execution time totals.

CONNECTIVITY_MATRIX (CM)

Program description and connectivity matrix only.

EXECUTION_TIME_TOTALS (ETT)

Program description and execution time totals only.

If AMOUNT is omitted, ALL is used.

Remarks

- By default, the SAVE_PROGRAM_MEASURES subcommand saves the execution time totals and the connectivity matrix. If the session that uses the saved program measurement environment will not use the execution time totals or connectivity matrix, you can direct the subcommand not to save that information with the AMOUNT parameter.
- The SAVE_PROGRAM_MEASURES subcommand does not discard the program measurement statistics. The statistics are discarded when you specify another program description or end the session.
- To use the saved program environment in another session, enter a RESTORE_PROGRAM_MEASURES subcommand that specifies the file containing the saved program environment.
- The MEASURES file is written as a sequential data file. It is not intended to be listed; its only intended use is to resume a MEASURE_PROGRAM_EXECUTION session.
- For more information, see the NOS/VE Object Code Management manual.

Examples The following subcommand copies the program description and any accumulated statistics to file SAVED_MEASUREMENT.

```
MPE/save_program_measure measures=saved_measurement
```

SET_PROGRAM_DESCRIPTION MEAPE Subcommand

Purpose Specifies the program whose performance is to be measured.

Format

SET_PROGRAM_DESCRIPTION or
SETPD
TARGET_TEXT=file
FILE=list of file
LIBRARY=list of file
MODULE=list of any
STARTING_PROCEDURE=any
STACK_SIZE=integer
STATUS=status variable

Parameters

TARGET_TEXT or **TT**

Object file or object library containing the modules to be measured. This parameter is required.

FILE or *FILES* or *F*

Object list for the program. Each module in the specified object files and object libraries is unconditionally included in the program. The list must include the target text file. If FILE is omitted, the object list for the program consists of only the file specified on the TARGET_TEXT parameter.

LIBRARY or *LIBRARIES* or *L*

List of object libraries added to the program library list.

MODULE or *MODULES* or *M*

Module list.

You use a string value for a module whose name is not an SCL name.

Each module is unconditionally loaded from the object libraries in the program library list.

STARTING_PROCEDURE or *SP*

Name of the entry point where execution begins.

You use a string value for an entry point whose name is not an SCL name.

If STARTING_PROCEDURE is omitted, the last transfer symbol encountered during loading is used.

STACK_SIZE or *SS*

Upper size limit in bytes of the run-time stack used for procedure call linkages and local variables. If STACK_SIZE is omitted, a 2-million byte limit is used.

Remarks

- The program to be measured should be debugged and ready for use in a production environment. The program description specified on the subcommand should be the same program description used to execute the program in a production environment.
- When you execute the SET_PROGRAM_DESCRIPTION subcommand, any program description previously in effect and any program measurement statistics accumulated for that program are discarded.
- For more information, see the NOS/VE Object Code Management manual.

Examples

The following subcommand specifies that program modules are on files LGO and SUBLGO but only the module on file LGO is to be measured.

```
MPE/set_program_description target_text=lgo ..
MPE../file=(lgo,sublgo)
```

RECOVER_KEYED_FILE 28

RECOVER_KEYED_FILE Command

Purpose Begins a keyed-file recovery attempt.

Format

RECOVER_KEYED_FILE or
RECKF
FILE=file
PASSWORD=name
STATUS=status variable

Parameters

FILE or F

File path to the damaged keyed file to be recovered. This parameter is required.

If the damaged file does not currently exist, its cycle number cannot be determined by default. Therefore, the file path must explicitly specify the file cycle number so that the utility can reload the correct backup copy.

PASSWORD or *PW*

File password specified when BACKUP_PERMANENT_FILE wrote the backup copy of the file. A file password is optional, but, if a password exists for the file, it is required on this command. If no password exists for the file, NONE can be specified.

The file password in effect when the backup copy was written must be the same password in effect when the file was damaged. Otherwise, the backup copy cannot replace the damaged file.

Remarks

- The LOG_RESIDENCE attribute of the file specified on the command must match the LOG_RESIDENCE attribute of the backup copy to be reloaded. RECOVER_KEYED_FILE cannot use a backup copy that was written before the LOG_RESIDENCE attribute of the file was changed.

- If the file does not currently exist and the LOG_RESIDENCE of its backup copy is not the default log, you must enter a SET_FILE_ATTRIBUTE command for the file. The command must specify the same file cycle specified on the RECOVER_KEYED_FILE command and the same LOG_RESIDENCE as that of the backup copy to be used. (See the Example.)

- Similarly, if the file does not currently exist, but the file had a password when the backup copy was written, you must create the file with the same password. To do so, enter a CREATE_FILE command specifying the file path (including its cycle number) and the PASSWORD parameter.

- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples The following session attempts to restore a keyed file that no longer exists using its latest backup copy. When the latest backup copy was written, the file password was HUSH_HUSH and the LOG_RESIDENCE attribute was $USER.MY_LOG. Therefore, those values must be reestablished for the file cycle.

```
/recover_keyed_file, $user.keyed_file.1
reckf/create_file, $user.keyed_file.1, ..
reckf../password=hush_hush
reckf/set_file_attribute, $user.keyed_file.1, ..
reckf../log_residence=$user.my_log
reckf/recover_file_media
```

28

HELP RECKF Subcommand

Purpose Provides access to online information about the utility.

Format

HELP or
HEL
SUBJECT=string
MANUAL=file
STATUS=status variable

Parameters

SUBJECT or *S*

Topic to be found in the index of the online manual. The topic title must be enclosed in apostrophes ('topic').

If you omit the SUBJECT parameter, HELP displays a list of the available subcommands and prompts for display of a subcommand description in the online manual.

MANUAL or *M*

Online manual file to be read. If you omit the MANUAL parameter, the default is AFM. The working catalog is searched for the AFM file and then the $SYSTEM.MANUALS catalog.

Remarks

- If the SUBJECT parameter specifies a topic that is not in the manual index, a nonfatal error is returned notifying you that the topic could not be found.
- The default manual file, $SYSTEM.MANUALS.AFM, contains the online version of the NOS/VE Advanced File Management Usage manual, as provided with the NOS/VE system.
- If your terminal is defined for full-screen applications, the online manual is displayed in screen mode. Help is available for reading the online manual. To leave the online manual and return to the utility, use QUIT.

Examples

The following session shows the default display returned by the HELP subcommand.

```
/recover_keyed_file, $user.keyed_file.1
reckf/help
The following Recover_Keyed_File subcommands are available:
RECOVER_FILE_MEDIA
VOID_LOG_FOR_RESTORED_FILE
HELP
QUIT

For a description of a subcommand in the online manual, enter:
HELP subject = '<subcommand>'

To return from an online manual, enter: QUIT
reckf/quit
/
```

QUIT
RECKF Subcommand

Purpose Ends the RECOVER_KEYED_FILE session.

Format

QUIT or
QUI
STATUS=status variable

Remarks

- The QUIT command is required to end a session.
- A recovery attempt that returns a fatal error ends the session.

RECOVER_FILE_MEDIA
RECKF Subcommand

Purpose Reloads a backup of the file and then updates it using an update recovery log for the file.

Format

RECOVER_FILE_MEDIA or
RECFM
DAYS_SINCE_LAST_GOOD=integer
HOURS_SINCE_LAST_GOOD=integer
MINUTES_SINCE_LAST_GOOD=integer
FILE_CLASS=application
INITIAL_VOLUME=name
STATUS=status variable

Parameters

DAYS_SINCE_LAST_GOOD or *DSLG*

Number of days since the damaged file was intact (any integer not less than 0). It is used with the next two parameters to determine the backup copy to be reloaded.

If the first three parameters are omitted, the default value for each is 0, causing the latest backup copy to be reloaded.

HOURS_SINCE_LAST_GOOD or *HSLG*

Number of hours (added to the days specified by the first parameter) since the damaged file was intact (an integer from 0 through 23).

If the first three parameters are omitted, the latest backup copy is reloaded.

MINUTES_SINCE_LAST_GOOD or *MSLG*

Number of minutes (added to the days and hours specified by the first two parameters) since the damaged file was intact (an integer from 0 through 59).

If the first three parameters are omitted, the latest backup copy is reloaded.

FILE_CLASS or *FC*

Specifies the class of the file to be assigned. Refer to the REQUEST_MASS_STORAGE command in the NOS/VE System Performance and Maintenance manual, Volume 2, for class assignments and a complete description of this parameter.

INITIAL_VOLUME or *IV*

Name specifying the volume serial number (VSN) of the mass storage volume to which the file is to be assigned. The name is specified as a string of from 1 through 6 characters. Refer to the REQUEST_MASS_STORAGE command in the NOS/VE System Performance and Maintenance manual, Volume 2, for a complete description of this parameter.

Remarks

- This subcommand is effective only if both a backup copy and an update recovery log are available for the file.
- An update recovery log is maintained for the file only if its LOGGING_OPTIONS attribute includes the option ENABLE_MEDIA_RECOVERY.
- The subcommand can only reload backup copies created by the BACKUP_PERMANENT_FILE utility because those backup copies are recorded in the update recovery log for the file. (The ENABLE_MEDIA_RECOVERY logging option must be set before the backup.)
- For a backup copy to be used, the file password (if any), the LOG_RESIDENCE attribute, and the LOGGING_OPTIONS attribute for the file must not have changed since the backup copy was written.

- The FILE_CLASS and INITIAL_VOLUME parameters are described in detail as parameters of the REQUEST_MASS_STORAGE command in the NOS/VE System Performance and Maintenance manual, Volume 2.
- Once a keyed file is recovered using RECOVER_FILE_MEDIA, it must be backed up (using the BACKUP_PERMANENT_FILE utility) before it can be updated.
- The subcommand issues progress messages as it proceeds. Be sure to read the messages as they appear.
- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples The following session recovers the file using the last backup copy.

```
/recover_keyed_file, $user.my_keyed_file
reckf/recover_file_media
/
```

VOID_LOG_FOR_RESTORED_FILE RECKF Subcommand

Purpose Discards the update recovery log associated with a file that has been restored using the RESTORE_PERMANENT_FILE utility.

Format **VOID_LOG_FOR_RESTORED_FILE** or **VOILFRF**

STATUS=status variable

Remarks

- This subcommand is provided for situations in which an older version of the file is restored using the RESTORE_PERMANENT_FILE utility, and the user, content with this version, does not want to try to recover lost updates from the log.

- Updates cannot be recorded on a log associated with a restored file because the updates on the log do not correspond to the restored version of the file. (The restored file is an older version.) As a result, this subcommand is used to discard all past logged updates for the restored file.

- After the update recovery log is discarded, a backup copy of the file must be created by the BACKUP_PERMANENT_FILE utility if subsequent updates are to be recorded on the log.

- For more information, see the NOS/VE Advanced File Management Usage manual.

RELEASE_MASS_STORAGE 29

RELEASE_MASS_STORAGE Command

Purpose Starts the RELEASE_MASS_STORAGE utility session and specifies the file to receive the report on file cycles that are selected during the utility session.

Format

RELEASE_MASS_STORAGE or
RELMS
LIST=file
STATUS=status variable

Parameters *LIST* or *L*

Specifies the name of the file that receives a list of the file cycles that are released from mass storage during the utility session. The default is $LIST.

Remarks

- The SET_RELEASE_OPTIONS subcommand specifies whether the utility session releases the selected file cycles, reports on which file cycles were selected, or performs both options.
- For more information, see the NOS/VE File Archiving manual.

Examples To start the utility session and save the summary report in file RESULTS, enter:

```
/release_mass_storage list=results
relms/
```

CHANGE_FILE_SIZE RELMS Subcommand

Purpose Specifies the file size restrictions for release candidates.

Format

CHANGE_FILE_SIZE or
CHAFS
SIZE=range of integer
STATUS=status variable

Parameters **SIZE or S**

Specifies the minimum file size in bytes, a range in file size in bytes, or all file sizes with the keyword ALL. Specifying the keyword ALL indicates that file size is not a factor when selecting files to release. The default is ALL.

Remarks

- If you enter multiple CHANGE_FILE_SIZE subcommands, the most recent entry takes precedence.
- If you omit this subcommand, file size is not a consideration when selecting file cycles for release.
- For more information, see the NOS/VE File Archiving manual.

Examples To release the mass storage copies of archived file cycles that are 10 megabytes or larger and are in subcatalog TEST_DATA, enter:

```
relms/change_file_size 10000000
relms/display_file_size
INCLUDING CYCLES OF LENGTH AT LEAST:     10000000
INCLUDING CYCLES OF LENGTH AT MOST:    4398046511103
relms/release_catalog $user.test_data
relms/quit
```

To specify a file size range of 250,000 bytes to 6,000,000 bytes, enter:

```
relms/change_file_size 250000..6000000
relms/display_file_size
INCLUDING CYCLES OF LENGTH AT LEAST:     250000
INCLUDING CYCLES OF LENGTH AT MOST:     6000000
```

29

CHANGE_WEIGHTS RELMS Subcommand

Purpose Specifies the multipliers used to influence the priority assigned to a file cycle in the release candidate list.

Format

CHANGE_WEIGHTS or
CHANGE_WEIGHT or
CHAW
AGE_MULTIPLIER=integer
SIZE_MULTIPLIER=integer
STATUS=status variable

Parameters

AGE_MULTIPLIER or *AM*

Specifies the multiplier for the file cycle age. A file cycle age is the number of minutes since the file cycle was last accessed. The default age multiplier is 1.

SIZE_MULTIPLIER or *SM*

Specifies the multipliers for the file cycle size. The size is the length of the file cycle in bytes. The default size multiplier is 1.

Remarks

- Before the system releases file cycle data, it prioritizes the release candidates using the following algorithm:

  priority = (age * age_multiplier) + (size * size_multiplier) where *age* is the number of minutes since the file cycle was modified, and *size* is the length of the file in bytes.

- For more information, see the NOS/VE File Archiving manual.

Examples

To double the weight assigned to a file's size, enter:

```
relms/change_weights size_multiplier=2
```

To have files prioritized by only the file's size, enter:

```
relms/change_weights age_multiplier=0
```

To have files prioritized so that the smallest file has the highest priority and age does not matter, enter:

```
relms/change_weights age_multiplier=0 ..
relms../size_multiplier=-1
```

DISPLAY_COPIES_NEEDED RELMS Subcommand

Purpose Displays the minimum number of current archive copies that must exist before a file cycle can be released.

Format

DISPLAY_COPIES_NEEDED or
DISCN
STATUS=status variable

Remarks For more information, see the NOS/VE File Archiving manual.

Examples To display the minimum number of current archive copies that must exist before a file cycle can be released, enter:

```
relms/display_copies_needed
THE NUMBER OF COPIES REQUESTED ARE:
 TAPE                                =   1
```

In this example, only one archive copy must be made by the tape processor before a file cycle can be released from mass storage.

DISPLAY_FILE_SIZE RELMS Subcommand

Purpose Displays the file size criteria used to select release candidates.

Format

DISPLAY_FILE_SIZE or
DISFS
STATUS=status variable

Remarks

- Use the CHANGE_FILE_SIZE subcommand to set the file size criteria.
- For more information, see the NOS/VE File Archiving manual.

Examples To display the current file size restrictions for releasing mass storage space, enter:

```
relms/display_file_size
INCLUDING CYCLES OF LENGTH AT LEAST:    250000
INCLUDING CYCLES OF LENGTH AT MOST:    6000000
```

This shows that only archived files with a length between 250,000 and 6,000,000 bytes can be released from mass storage.

DISPLAY_RELEASE_OPTIONS RELMS Subcommand

Purpose Displays information on whether the RELEASE_MASS_STORAGE utility session releases archived file cycles, writes a report that lists the selected archived file cycles, or performs both options.

Format

DISPLAY_RELEASE_OPTIONS or
DISRO
STATUS=status variable

Remarks

- Use the SET_RELEASE_OPTIONS subcommand to set the processing option.
- For more information, see the NOS/VE File Archiving manual.

Examples To display the current release processing options, enter:

```
relms/display_release_options
RELEASE_PROCESSING_OPTION:  ALL
```

The option ALL specifies that the utility session will release mass storage space and generate a report that lists the archived file cycles that were released.

DISPLAY_WEIGHTS RELMS Subcommand

Purpose Displays the multipliers used to influence the priority assigned to a file cycle in the release candidate list.

Format

DISPLAY_WEIGHTS or
DISW
STATUS=status variable

Remarks

- The factors used to determine weights are the age multiplier and the size multiplier.
- To set the weights, use the CHANGE_WEIGHTS subcommand.
- For more information, see the NOS/VE File Archiving manual.

Examples To display the weights assigned to the current processors, enter:

```
relms/display_weights
PRIORITY, FILE AGE (MINUTES) MULTIPLIED BY:     1
PRIORITY, FILE SIZE (BYTES) MULTIPLIED BY:      1
```

EXCLUDE_CATALOG RELMS Subcommand

Purpose Excludes a catalog from subsequent releasing or reporting. A catalog needs to be excluded only if it is a subcatalog of a catalog that is being released.

Format

EXCLUDE_CATALOG or
EXCC
CATALOG=file
STATUS=status variable

Parameters

CATALOG or **C**

Specifies the catalog to exclude from subsequent releasing or reporting. This parameter is required.

Remarks

- This subcommand takes precedence over all INCLUDE subcommand entries.
- For more information, see the NOS/VE File Archiving manual.

Examples This example excludes subcatalog CATALOG_1 from the releasing of master catalog TKWS87:

```
/release_mass_storage
relms/exclude_catalog catalog=.tkws87.catalog_1
relms/release_catalog catalog=.tkws87
relms/quit
```

EXCLUDE_FILE RELMS Subcommand

Purpose Prevents an archived file or cycle from being released from mass storage.

Format

EXCLUDE_FILE or
EXCF
FILE=file
STATUS=status variable

Parameters

FILE or F

Specifies the archived file or cycle to keep on mass storage. This parameter is required.

Remarks

- This subcommand takes precedence over all INCLUDE subcommands.
- For more information, see the NOS/VE File Archiving manual.

Examples This example releases all archived file cycles in master catalog TKWS87, except for file CATALOG_1.REGISTER:

```
/release_mass_storage
relms/exclude_file file=.tkws87.catalog_1.register
relms/release_catalog catalog=.tkws87
relms/quit
```

EXCLUDE_HIGHEST_CYCLES RELMS Subcommand

Purpose Excludes a specified number of high (highest numbered) file cycles from subsequent releasing of mass storage space.

Format

EXCLUDE_HIGHEST_CYCLES or
EXCLUDE_HIGHEST_CYCLE or
EXCHC
NUMBER_OF_CYCLES=keyword or *integer*
STATUS=status variable

Parameters *NUMBER_OF_CYCLES* or *NOC*

Specifies the number of high cycles to exclude. The value must be an integer in the range from 0 to 999 or the keyword ALL. The keyword ALL specifies that all file cycles are excluded. The default is 3.

Remarks

- This subcommand takes precedence over all INCLUDE subcommands.
- For more information, see the NOS/VE File Archiving manual.

Examples This example releases all archived file cycles, except the highest numbered cycle, in the TKWS87 master catalog:

```
/release_mass_storage
relms/exclude_highest_cycles number_of_cycles=1
relms/release_catalog catalog=.tkws87
relms/quit
```

INCLUDE_CYCLES RELMS Subcommand

Purpose Selects archived file cycles for subsequent release from mass storage based on the creation date and time, the last access date and time, the last modification date and time, or the expiration date of the cycle.

Format

INCLUDE_CYCLES or
INCLUDE_CYCLE or
INCC
SELECTION_CRITERIA=keyword
MONTH=keyword or *integer*
DAY=integer
YEAR=integer
HOUR=integer
MINUTE=integer
SECOND=integer
MILLISECOND=integer
STATUS=status variable

Parameters

SELECTION_CRITERIA or SC

Specifies the archived file cycles that are candidates for release from mass storage space, or should appear in the reports. File cycles are selected based on a cutoff date and time specified by the MONTH, DAY, YEAR, MINUTE, HOUR, SECOND, and MILLISECOND parameters. This parameter is required. You must specify one of the following keywords:

ACCESSED_BEFORE

Selects cycles that were accessed before the specified date and time.

ACCESSED_AFTER

Selects cycles that were accessed after the specified date and time.

CREATED_BEFORE

Selects cycles created before the specified date and time.

CREATED_AFTER

Selects cycles created after the specified date and time.

EXPIRED_BEFORE

Selects cycles that expire before the specified date. The default is the current date.

EXPIRED_AFTER

Selects cycles that expire after the specified date.

MODIFIED_BEFORE

Selects cycles that were modified before the specified date and time.

MODIFIED_AFTER

Selects cycles that were modified after the specified date and time.

ALL

Removes the effect of any previous INCLUDE_CYCLES subcommand. Do not specify values for the parameters MONTH, DAY, YEAR, HOUR, MINUTE, SECOND, and MILLISECOND when using this value.

MONTH or *M*

Specifies the month of the cutoff date applied by the SELECTION_CRITERIA parameter. This value must be an integer from 1 to 12 (corresponding to the months of January to December), or the name of the month. This parameter is required unless SELECTION_CRITERIA specifies EXPIRED_BEFORE, in which case the default is the current month.

DAY or *D*

Specifies the day of the cutoff date applied by the SELECTION_CRITERIA parameter. This value must be an integer from 1 to 31. This parameter is required unless SELECTION_CRITERIA specifies EXPIRED_BEFORE, in which case the default is the current day.

YEAR or *Y*

Specifies the year of the cutoff date applied by the SELECTION_CRITERIA parameter. This parameter must be an integer from 1900 to 1999. This parameter is required unless SELECTION_CRITERIA specifies EXPIRED_BEFORE, in which case the default is the current year.

HOUR or *HR*

Specifies the hour of the cutoff time applied by the SELECTION_CRITERIA parameter. This value must be an integer from 0 to 23 corresponding to the hour on a 24-hour clock. The default is 0.

MINUTE or *MIN*

Specifies the minute of the cutoff time applied by the SELECTION_CRITERIA parameter. This value must be an integer in the range from 0 to 59. The default is 0.

SECOND or *SEC*

Specifies the second of the cutoff time applied by the SELECTION_CRITERIA parameter. This parameter must be an integer in the range from 0 to 59. The default is 0.

MILLISECOND or *MSEC*

Specifies the millisecond of the cutoff time applied by the SELECTION_CRITERIA parameter. This parameter must be an integer from 0 to 999. The default is 0.

Remarks

- By entering the INCLUDE_CYCLES subcommand twice with different values for the SELECTION_CRITERIA parameter, you can create a time window the file cycles must meet in order to be included in the releasing of mass storage. The following values for the SELECTION_CRITERIA parameter can be used together:

  ACCESSED_AFTER and ACCESSED_BEFORE
  CREATED_AFTER and CREATED_BEFORE
  EXPIRED_AFTER and EXPIRED_BEFORE
  MODIFIED_AFTER and MODIFIED_BEFORE

- To release mass storage space for archived file cycles that have not been accessed since the cutoff date and time, enter the INCLUDE_CYCLES subcommand with SELECTION_CRITERIA specifying ACCESSED_BEFORE.

- The EXCLUDE_CATALOG, EXCLUDE_FILE, and EXCLUDE_HIGHEST_CYCLES subcommands take precedence over the INCLUDE_FILES subcommand.

- For more information, see the NOS/VE File Archiving manual.

Examples

To release all archived file cycles that were modified on or after June 2, 1988, enter:

```
/release_mass_storage
relms/include_cycles ..
relms../selection_criteria=modified_after ..
relms../month=6 day=2 year=1988
relms/release_all_files
relms/quit
```

To release all archived files that were modified on or after June 1 and before June 6 of 1988, enter:

```
/release_mass_storage
relms/include_cycles selection_criteria=..
relms../modified_after ..
relms../month=june day=1 year=1988
relms/include_cycles selection_criteria=..
relms../modified_before ..
relms../month=june day=6 year=1988
relms/release_all_files
relms/quit
```

INCLUDE_VOLUMES RELMS Subcommand

Purpose Specifies to release archived file cycles from the specified mass storage disk volumes.

Format

INCLUDE_VOLUMES or
INCLUDE_VOLUME or
INCV
RECORDED_VSNS=list of: keyword or name
CYCLE_SELECTION=keyword
STATUS=status variable

Parameters

RECORDED_VSNS or **RECORDED_VSN** or **RVSN**

Specifies the recorded VSN of the disk volumes to include. This value can be a VSN or the keyword ALL, which means the archived file cycles on all disk volumes are release candidates. This parameter is required.

CYCLE_SELECTION or *CS*

Specifies which cycles on a disk volume to include. You can specify the following keywords; the default is MULTIPLE_VOLUMES:

INITIAL_VOLUME (IV)

Selects cycles with the beginning-of-information (BOI) on the volume specified by RECORDED_VSN. Cycles with the BOI on another volume are skipped.

MULTIPLE_VOLUMES (MV)

Selects all cycles residing either partially or completely on the volume.

Remarks

- The system ignores values for the CYCLE_SELECTION parameter when RECORDED_VSN specifies ALL.
- For more information, see the NOS/VE File Archiving manual.

Examples This example releases the archived file cycles for user TKWS87 that reside on the disk volume VOL033:

```
/release_mass_storage
relms/include_volume recorded_vsn=VOL033 ..
relms../cycle_selection=multiple_volumes
relms/release_catalog catalog=.tkws87
relms/quit
```

QUIT
RELMS Subcommand

Purpose Ends the utility session.

Format **QUIT** or
QUI

Parameters None.

Remarks For more information, see the NOS/VE File Archiving manual.

RELEASE_ALL_FILES
RELMS Subcommand

Purpose Either releases the mass storage copies of all archived file cycles or lists the archived file cycles. This option is determined by the SET_RELEASE_OPTIONS subcommand.

Format **RELEASE_ALL_FILES** or
RELAF

Parameters None.

Remarks

- To use this subcommand, you must have system administration capabilities, or be at the system console.
- Archived file cycles in the $SYSTEM catalog cannot be released.
- The RELEASE_ALL_FILES subcommand skips file cycles that are attached to a job.
- Previous INCLUDE and EXCLUDE subcommands restrict the file cycles that are selected.
- A previous CHANGE_FILE_SIZE subcommand determines the size requirements for a release candidate.
- A previous CHANGE_WEIGHTS subcommand determines the priority for file cycle candidates based on file age and size.
- For more information, see the NOS/VE File Archiving manual.

Examples

To release space on disk volume DISK02 and DISK03 for all file cycles that have been archived, enter:

```
/release_mass_storage
relms/include_volumes rvsn=(disk02, disk03)
relms/release_all_files
relms/quit
```

RELEASE_CATALOG RELMS Subcommand

Purpose

Either releases the mass storage copies of the archived file cycles in the specified catalog or family, or lists the archived file cycles in the specified catalog or family. This option is determined by the SET_RELEASE_OPTIONS subcommand.

Format

RELEASE_CATALOG or
RELC
CATALOG=file
STATUS=status variable

Parameters CATALOG or C

Specifies the catalog or family from which the archived file cycle data is released. This parameter is required.

Remarks

- Archived file cycles in the $SYSTEM catalog cannot be released.
- The release of mass storage and/or report starts with the file cycles in the specified catalog and continues with the file cycles in all catalogs subordinate to the specified catalog.
- The RELEASE_CATALOG subcommand skips file cycles that are attached to a job.
- Previous INCLUDE and EXCLUDE subcommands restrict the file cycles that are selected.
- A previous CHANGE_FILE_SIZE subcommand determines the size requirements for a release candidate.
- A previous CHANGE_WEIGHTS subcommand determines the priority for file cycle candidates based on file age and size.
- For more information, see the NOS/VE File Archiving manual.

Examples To release the mass storage space for archived file cycles in subcatalog TEST_DATA, enter:

```
/release_mass_storage list=..
relms../release_catalog_test_data
relms/release_catalog catalog=$user.test_data
relms/quit
```

29

RELEASE_FILE
RELMS Subcommand

Purpose Either releases the mass storage copy of the specified archived file, or lists the specified archived file. This option is determined by the SET_RELEASE_OPTIONS subcommand.

Format

RELEASE_FILE or
RELF
FILE=file
PASSWORD=keyword or *name*
STATUS=status variable

Parameters

FILE or **F**

Specifies the permanent file or permanent file cycle to release. Specifying a permanent file causes the releasing of all of its archived copies. This parameter is required.

PASSWORD or *PW*

Specifies either the password for the permanent file or the keyword NONE (if the permanent file does not have a password). The default is the keyword NONE.

Remarks

- Archived file cycles in the $SYSTEM catalog cannot be released.
- The RELEASE_FILE subcommand skips file cycles that are attached to a job.
- Previous INCLUDE and EXCLUDE subcommands restrict the file cycles that are selected.
- A previous CHANGE_FILE_SIZE subcommand determines the size requirements for a release candidate.
- A previous CHANGE_WEIGHTS subcommand determines the priority for file cycle candidates based on file age and size.
- For more information, see the NOS/VE File Archiving manual.

Examples

To release the mass storage space for the archived file $USER.TEST_DATA.CASE_100, enter:

```
/release_mass_storage list=..
../released_test_data_case_100
rel ms/release_file file=$user.test_data.case_100
rel ms/quit
```

SET_LIST_OPTIONS RELMS Subcommand

Purpose Specifies the information that is written in the list file for the utility session. The list file is specified by the LIST parameter of the RELEASE_MASS_STORAGE command.

Format

SET_LIST_OPTIONS or
SET_LIST_OPTION or
SETLO
FILE_DISPLAY_OPTIONS=list of keyword
CYCLE_DISPLAY_OPTIONS=list of keyword
DISPLAY_EXCLUDED_ITEMS=boolean
STATUS=status variable

Parameters

FILE_DISPLAY_OPTIONS or *FILE_DISPLAY_OPTION* or *FDO*

Selects the type of file information to record in the list file. You can specify the following keywords; the default is NONE:

ACCOUNT (A)

Records the account name.

PROJECT (P)

Records the project name.

NONE

Records only the file name.

ALL

Records the account and project name.

CYCLE_DISPLAY_OPTIONS or
CYCLE_DISPLAY_OPTION or *CDO*

Selects the type of cycle information to record in the list file. The cycle number and whether the cycles as excluded are always recorded, unless you specify NONE. You can specify the following keywords; the default is MODIFICATION_DATE_TIME and SIZE:

ACCESS_COUNT (AC)

Records the number of times the cycle has been accessed.

ACCESS_DATE_TIME (ADT)

Records the date and time the cycle was last accessed.

CREATION_DATE_TIME (CDT)

Records the date and time the cycle was created.

EXPIRATION_DATE (ED)

Records the expiration date of the cycle.

GLOBAL_FILE_NAME (GFN)

Records the internally generated global file name. This name is not released.

MODIFICATION_DATE_TIME (MDT)

Records the date and time the cycle was last modified.

RECORDED_VSN (RVSN)

Records all mass storage volumes on which the cycle resides.

SIZE (S)

Records the size of the cycle in bytes.

NONE

Records only the cycle number.

ALL

Selects all of the display options.

DISPLAY_EXCLUDED_ITEMS or
DISPLAY_EXCLUDED_ITEM or *DEI*

Specifies whether excluded catalogs, files, and cycles are included in the list file. The default is FALSE.

TRUE

Writes all excluded catalogs, files, and cycles in the list file.

FALSE

Does not write excluded catalogs, files, and cycles in the list file.

Remarks For more information, see the NOS/VE File Archiving manual.

SET_RELEASE_OPTIONS RELMS Subcommand

Purpose Specifies whether the RELEASE_MASS_STORAGE utility session is to release archived file cycles, to write a report listing the selected archived file cycles, or to perform both options.

Format

SET_RELEASE_OPTIONS or
SET_RELEASE_OPTION or
SETRO
PROCESSING_OPTION=keyword
STATUS=status variable

Parameters *PROCESSING_OPTION* or *PO*

Specifies whether to release the selected file cycles from mass storage, to write a report specifying which file cycles were selected, or to perform both options. If you specify ALL or REPORT, the report is written on the list file specified by the LIST parameter of the RELEASE_MASS_STORAGE command. The default is ALL. The keyword options are:

ALL

Release the selected file cycles from mass storage and write a report.

RELEASE

Release the selected file cycles from mass storage.

REPORT

Report on the archive file cycles that were selected during the utility session.

Remarks

- If you omit this subcommand, the utility session releases the mass storage space and writes a list of the file cycles that were released.
- For more information, see the NOS/VE File Archiving manual.

Examples To produce a report of the archived file cycles in subcatalog TEST_DATA that are candidates for release, enter:

```
/release_mass_storage list=
../files_selected_from_test_data
relms/set_release_options processing_option=report
relms/release_catalog catalog=$user.test_data
relms/quit
```

RESTORE_FOREIGN_FILES 30

RESTORE _FOREIGN _FILES Command

Purpose This command invokes a command utility that is capable of migrating NOS files from tape to NOS/VE.

Format

RESTORE _FOREIGN _FILES or
RESFF
LIST=file
STATUS=status variable

Parameters *LIST* or *L* or *OUTPUT* or *O*

This is the default file to contain displays written during the utility session. Display-generating subcommands are able to override this specification. The default file is $OUTPUT.

Remarks For more information, see the Migration From NOS to NOS/VE manual.

CREATE _TAPE _CATALOG RESFF Subcommand

Purpose This command reads a foreign backup tape and creates a NOS/VE file containing a tape catalog of the content of one or more tapes. This tape catalog is used to select files for restoration from tape to NOS/VE.

Format

CREATE _TAPE _CATALOG or
CRETC
BACKUP_FILE=file
NEW_TAPE _CATALOG=file
BACKUP_FILE _SYSTEM=keyword
STATUS=status variable

Parameters **BACKUP_FILE or BF**

This is the tape file. It can be standard labelled or unlabelled.

NEW_TAPE_CATALOG or *NTC* or *TAPE_CATALOG* or *TC*

The NOS/VE file to contain the tape catalog of the backup file contents. In the case of an unlabelled tape all files will be included in the tape catalog. In the case of a labelled tape, only one member of a multi-file set will be included. The default file name is FOREIGN_TAPE_CATALOG.

BACKUP_FILE_SYSTEM or *BFS*

This specifies the system that was used to write the archive tapes from which files will be restored. NONE indicates a non-archive tape. If this parameter is not specified, the utility will determine the correct value by reading the tape. An abnormal status will be issued where recognition fails. The following values can be used:

NONE

NOS (N)

PFDUMP (P)

RECLAIM (R)

DISPLAY_TAPE_CATALOG_ENTRY RESFF Subcommand

Purpose This command displays the entries in the tape catalog.

Format

DISPLAY_TAPE_CATALOG_ENTRY or
DISPLAY_TAPE_CATALOG_ENTRIES or
DISTCE
WHERE_CASE=boolean
DISPLAY_OPTION=list of keyword
LIST=file
OLD_TAPE_CATALOG=file
STATUS=status variable

Parameters *WHERE_CASE* or *WC*

This specifies the catalog entries to display. The value is an NOS/VE boolean expression that can reference any of the variables that pertain to the backup file system for the tape catalog. The default displays only selected entries. Note: String comparisons are case sensitive.

DISPLAY_OPTION or *DO*

This selects what variables, or fields, of each entry are to be displayed. The order of specification determines the order in the display. Each field displayed has a sort key to determine the ordering of display lines. Ascending or descending order is applied depending on the field. The ALL option will display all variables that will fit on a line of the display. In this case, the ordering of fields on a line is based on estimated importance and field size. The list heading contains the variable names being displayed, in long or short format depending on available space. See LIST description for more information. Any options that are specified, but not supported by the type of tape catalog, will be diagnosed. If more options are specified than will fit on a display line, the condition will be diagnosed. The default is ALL.

ALL (A)

FULL (F)

CREATION_DATE (CD)

CYCLE_NUMBER (CN)

DATA_CONVERSION (DC)

DUMP_NUMBER (DN)

ENTRY_NUMBER (EN)

EXTERNAL_VSN (EV)

FILE_NUMBER (FN)

FOREIGN_FILE_NAME (FFN)

MODIFICATION_DATE (MD)

NATIVE_CATALOG_NAME (NCN)

NATIVE_FILE_NAME (NFN)

OWNER_ID (OI)

RECORD_ID (RI)

RECORD_NUMBER (RN)

RECORD_TYPE (RT)

SELECTED (S)

TEXT_FILE_TERMINATION (TFT)

TEXT_RECORD_TERMINATION (TRT)

USER_INDEX (UI)

LIST or *L* or *OUTPUT* or *O*

This specifies the file to receive the display output. The default is determined by the LIST parameter specified when invoking the RESTORE_FOREIGN_FILES utility. Two listing widths are supported depending on the PAGE_WIDTH file attribute of the list file. If the PAGE_WIDTH is less than 132 characters, the narrow width of 80 characters is used, else 132 is used.

OLD_TAPE_CATALOG or *OTC* or *TAPE_CATALOG* or *TC*

This specifies the foreign tape catalog to be displayed. The default file is FOREIGN_TAPE_CATALOG.

EDIT_TAPE_CATALOG RESFF Subcommand

Purpose This command makes a tape catalog available for editing by a subutility. It is by means of this editing process that subset members of a tape catalog can be selected for migration. The tape catalog can be saved for use later in case the same files need to be accessed from the tape again.

Format **EDIT_TAPE_CATALOG** or
EDITC
OLD_TAPE_CATALOG=file
NEW_TAPE_CATALOG=file
STATUS=status variable

Parameters *OLD_TAPE_CATALOG* or *OTC* or *TAPE_CATALOG* or *TC*

The file that contains the tape catalog to be edited. The default file is FOREIGN_TAPE_CATALOG.

NEW_TAPE_CATALOG or *NTC*

The file to contain the tape catalog that results from the editing session. If this parameter is omitted, the resulting tape catalog will overwrite the tape catalog to be edited.

QUIT
RESFF Subcommand

Purpose This command terminates the RESTORE_FOREIGN_FILES utility session. It has no parameters.

Format **QUIT** or
QUI

Parameters None.

RESTORE_SELECTED_FILES
RESFF Subcommand

Purpose This command creates a tape request (if the BACKUP_FILE is not specified) and restores the entries to the native file names and native catalogs specified in the selected tape catalog entries.

Format **RESTORE_SELECTED_FILES** or
RESTORE_SELECTED_FILE or
RESSF
BACKUP_FILE=file
OLD_TAPE_CATALOG=file
STATUS=status variable

Parameters *BACKUP_FILE* or *BF*

By default, the backup file is automatically requested and returned during command execution. If you expect to use this tape several times during a session, you can use the CREATE_170_REQUEST command and specify the file name for the tape. In this case the tape will not be returned when the command terminates. Specify the CREATE_170_REQUEST command and, if this is a labelled tape, use the FILE_SET_POSITION parameter to specify the set member.

OLD_TAPE_CATALOG or *OTC* or *TAPE_CATALOG* or *TC*

The file containing the tape catalog to use in restoring the files from tape. The default file name is FOREIGN_TAPE_CATALOG.

RESTORE _LOG 31

RESTORE _LOG Command

Purpose Begins a RESTORE _LOG utility session.

Format

RESTORE _LOG or
RESL
LOG _RESIDENCE = file
STATUS = status variable

Parameters **LOG _RESIDENCE** or **LR**

Catalog path containing the files comprising the log to be restored. This parameter is required.

Remarks

- Immediately after entering the RESTORE _LOG session, you should use the VALIDATE _LOG or RESTORE _REPOSITORIES subcommands to determine the type and extent of log damage, if any.
- For more information, see the NOS/VE Advanced File Management Usage manual.

DELETE _LOG _CONTROL _FILE RESL Subcommand

Purpose Deletes the log control file.

Format

DELETE _LOG _CONTROL _FILE or
DELLCF
STATUS = status variable

Remarks

- The log control file should be deleted only if it is damaged or if you want to force the log control file to be restored from the backup file. Damage to the log control file can be detected by the VALIDATE _LOG, RESTORE _REPOSITORIES, or RESTORE _LOG _CONTROL _FILE subcommands.
- For more information, see the NOS/VE Advanced File Management Usage manual.

31

DELETE_REPOSITORIES RESL Subcommand

Purpose Deletes log repositories.

Format

DELETE_REPOSITORIES or
DELETE_REPOSITORY or
DELR
REPOSITORIES=list of range of: keyword or **integer**
STATUS=status variable

Parameters

REPOSITORIES or **REPOSITORY** or **R**

Specifies which repositories in the log are to be deleted. This parameter is required.

List of integer

Specifies the repositories to be deleted. Values can be a list of repository numbers specified in the repository name. Repositories have names in the format AAF$REPOSITORY_n where n is the integer value specified; that is, AAF$REPOSITORY_1, starting at one for the first repository, and incremented sequentially and contiguously. The last repository is specified as AAF$REPOSITORY_0. You can specify as many values as there are repositories to be deleted. If more than one value is specified, the values must be enclosed in parentheses and separated by commas or spaces.

ALL or A

All repositories in the log are deleted.

Remarks

- Repositories should be deleted only if they are damaged or if you want to force the repositories to be restored from the backup files. Damage to repositories can be detected by the VALIDATE_LOG or RESTORE_REPOSITORIES subcommands.
- For more information, see the NOS/VE Advanced File Management Usage manual.

ENABLE_LOG
RESL Subcommand

Purpose Enables a disabled log; that is, makes the log available for general use.

Format

ENABLE_LOG or
ENAL
STATUS=status variable

Remarks

- If the log is disabled and it is usable; that is, the log is undamaged, ENABLE_LOG enables it. This makes the log available for general use.
- If the log is disabled but not usable, an error is displayed and the log remains disabled. Damage can be detected on the log control file and/or the repositories.
- A log must be enabled and usable before you can use it to recover keyed files.
- For more information, see the NOS/VE Advanced File Management Usage manual.

HELP
RESL Subcommand

Purpose Provides access to online information about the utility.

Format

HELP or
HEL
SUBJECT=string
MANUAL=file
STATUS=status variable

Parameters *SUBJECT* or *S*

Topic to be found in the index of the online manual. The topic must be enclosed in apostrophes ('topic').

If you omit the SUBJECT parameter, HELP displays a list of the available subcommands and prompts for display of a subcommand description in the online manual.

MANUAL or *M*

Online manual whose index is searched.

AFM

The AFM online manual index is searched.

File

File name of the online manual whose index is searched.

If MANUAL is omitted, the default is AFM. The working catalog is searched for the file and then the $SYSTEM.MANUALS is searched.

Remarks

- If the SUBJECT parameter specifies a topic that is not in the manual index, a nonfatal error is returned notifying you that the topic could not be found.
- The default manual file, $SYSTEM.MANUALS.AFM, contains the online version of the NOS/VE Advanced File Management Usage manu.
- If your terminal is defined for full-screen applications, online manuals are displayed in screen mode. Help on reading online manuals is available in the online manual. To leave the online manual and return to the utility, use QUIT.
- For more information, see the NOS/VE Advanced File Management Usage manual.

Examples The following session shows the default display returned by the HELP subcommand.

```
/restore_log
resl/help

The following RESTORE_LOG subcommands are available:

VALIDATE_LOG
RESTORE_REPOSITORIES
RESTORE_LOG_CONTROL_FILE
DELETE_REPOSITORIES
DELETE_LOG_CONTROL_FILE
ENABLE_LOG
HELP
QUIT

For the description of a subcommand in the online
manual, enter:  HELP subject = '<subcommand>'

To return from an online manual, enter:  QUIT
resl/quit
/
```

QUIT
RESL Subcommand

Purpose Ends the RESTORE_LOG session.

Format

QUIT or
QUI
STATUS=status variable

Remarks

- The QUIT command is required to end a session.
- For more information, see the NOS/VE Advanced File Management Usage manual.

31

RESTORE_LOG_CONTROL_FILE RESL Subcommand

Purpose Restores the log control file from the specified log backup file.

Format

RESTORE_LOG_CONTROL_FILE or
RESLCF
MEDIA=keyword
BACKUP_FILE=file
EXTERNAL_VSN=list of string
RECORDED_VSN=list of string
TYPE=keyword
STATUS=status variable

Parameters

MEDIA or **M**

Device class of the log backup file to be restored. This parameter is required.

MAGNETIC_TAPE_DEVICE or MTD

Indicates that the log backup file is stored on a labeled tape. (In this case, the BACKUP_FILE parameter is not used.)

MASS_STORAGE_DEVICE or MSD

Indicates that the log backup file specified by the BACKUP_FILE parameter is stored on disk. (In this case, the RECORDED_VSN, EXTERNAL_VSN, and TYPE parameters are not used.)

BACKUP_FILE or *BF*

The file path name of one of the backup files in the log (previously established by the CONFIGURE_LOG_BACKUP subcommand of the ADMINISTER_RECOVERY_LOG utility) to be used for restoring the log control file. This parameter must be specified if MEDIA is set to MASS_STORAGE_DEVICE.

EXTERNAL_VSN or *EVSN*

List of external VSNs identifying the tape volumes that compose the log backup file. The VSNs are specified as strings of from 1 through 6 characters enclosed in apostrophes.

RECORDED_VSN or *RVSN*

List of recorded VSNs of the tape volumes that compose the log backup file. The recorded VSN is in the ANSI VOL1 label on the volume. The VSNs are specified as strings of from 1 through 6 characters enclosed in apostrophes. This parameter must be specified if MEDIA is set to MAGNETIC_TAPE_DEVICE.

TYPE or *T*

Tape density of the tape drive on which the log backup file was written.

MT9$800

Indicates 800 cpi written by a nine-track tape drive.

MT9$1600

Indicates 1600 cpi written by a nine-track tape drive.

MT9$6250

Indicates 6250 cpi written by a nine-track tape drive.

MT18$38000

Indicates 38000 cpi written by a 16-track tape drive.

The default value is MT18$38000.

Remarks

- In general, the backup file that was written to most recently is the best one to specify first as the log backup file. If RESTORE_LOG_CONTROL_FILE fails, try again specifying the next most recent backup file, and so on.

- The log control file can be restored only if the log was configured for log backups (see the CONFIGURE_LOG_BACKUP subcommand of the ADMINISTER_RECOVERY_LOG utility). A copy of the log control file exists at the front of each log backup file, having been written there as part of the ongoing process of backing up the log.

- If the log control file is not already disabled, RESTORE_LOG_CONTROL_FILE immediately disables it. This is to ensure the log is not used while it is being restored. The log can be enabled using ENABLE_LOG (described later in this chapter).

- RESTORE_LOG_CONTROL_FILE restores a log control file only if it detects damage to the log control file. Damage to the log control file can also be detected by the RESTORE_REPOSITORIES or VALIDATE_LOG subcommands.

- Once a damaged log control file is restored, the log is no longer available for logging entries. The log is available only for recovering keyed files. To begin logging entries again, you must switch to a different log, or you must delete the log whose log control file has been restored, then recreate it.

- For more information, see the NOS/VE Advanced File Management Usage manual.

RESTORE_REPOSITORIES RESL Subcommand

Purpose Restores damaged repository log files from the log backup files.

Format **RESTORE_REPOSITORIES** or **RESR**
STATUS=status variable

Remarks

- Older repositories can be restored only if the log was configured for automatic backups (see CONFIGURE_LOG_BACKUPS of the ADMINISTER_RECOVERY_LOG utility). If the active repository is to be replaced, backups are not required.

- If the log is not already disabled, RESTORE_REPOSITORIES immediately disables it; this ensures that the log is not used while it is being restored. Once the log is restored, it can be enabled using ENABLE_LOG.

- Initially, RESTORE_REPOSITORIES determines the usability of the log; that is, the type and extent of log damage, if any.

- Once the log is restored, if recovery information is lost (for example, the active repository is lost, which had not yet been backed up), or if the log control file has been restored, the log is available only for recovery operations. To begin recording log entries again, you must switch to a different log, or you must delete the log, then recreate it.

- For more information, see the NOS/VE Advanced File Management Usage manual.

VALIDATE_LOG RESL Subcommand

Purpose Determines the usability of the log; that is, the type and extent of log damage, if any.

Format

VALIDATE_LOG or
VALL
STATUS=status variable

Remarks

- If damage to the log is detected and if the log is not already disabled, VALIDATE_LOG immediately disables it. This is to ensure that the log is not used while it is being restored. Once the log is restored, it can be enabled using ENABLE_LOG. If no damage to the log is detected, the log is not disabled.

- For more information, see the NOS/VE Advanced File Management Usage manual.

RESTORE_PERMANENT_FILES 32

RESTORE _PERMANENT_FILES Command

Purpose Initiates the utility that restores permanent files and catalogs from backup copies created by the BACKUP_PERMANENT_FILE utility. The restore operations are directed by RESTORE _PERMANENT_FILE subcommands.

Format

RESTORE _PERMANENT_FILES or
RESTORE _PERMANENT_FILE or
RESPF
LIST=file
STATUS=status variable

Parameters *LIST* or *L*

Identifies the file to which a summary of the results of the restore utility are written and, optionally, specifies how the file is to be positioned prior to use. Omission causes $LIST to be used.

Remarks

- The content of the list file can be specified using the SET_LIST_OPTION subcommand prior to using a RESTORE _PERMANENT_FILE subcommand. If the SET_LIST_OPTION subcommand is omitted, the modification date and time and size of the file are displayed for each permanent file cycle.
- For more information, see the NOS/VE System Usage manual.

Examples The following subcommand initiates a RESTORE _PERMANENT_FILE subcommand utility session. The subcommand specifies that the report listing be written to file RESTORE _LISTING.

```
/restore_permanent_files list=restore_listing
```

Following entry of this subcommand, RESTORE _PERMANENT_FILE subcommands can be entered in response to the following prompt:

```
PUR/
```

$BACKUP_FILE RESPF Function

Purpose Returns a string containing information on a backup file produced by the BACKUP_PERMANENT_FILE utility. Because this function causes the file to be rewound, only the first item of information found on the file can be queried and returned to you. When the string value is returned, all letters within the string are converted to uppercase. This function is valid only within the RESTORE_PERMANENT_FILE utility.

Format

$BACKUP_FILE or
$BF
(PARAMETER_1: file
PARAMETER_2: keyword)

Parameters

PARAMETER_1

Specifies the name of the backup file to be queried. This parameter is required.

PARAMETER_2

Specifies the particular attribute that is being queried. The following are valid keywords:

IDENTIFIER (I)

Returns a string containing the path name of the first name on the backup file.

IDENTIFIER_TYPE (IT)

Returns a string containing a name that indicates the type of the first item on the backup file. One of the following names is returned.

SET, CATALOG, FILE, CYCLE

If you do not specify a keyword, IDENTIFIER is assumed.

Remarks

- This function is especially useful when attempting to restore from a backup file for which the destination is known but the name of the file or catalog is unknown.
- The $BACKUP_FILE function always returns a string. The $FNAME function is included in the RESTORE_CATALOG command to convert this string to a file name. Once the string has been converted to a file name, you can use the file name in any subsequent RESTORE_FILE or RESTORE_CATALOG subcommands.
- For more information, see the NOS/VE System Usage manual.

Examples

For the following example, assume that you receive a backup tape produced by the BACKUP_CATALOG command and you wish to restore the catalog to your own $USER.MY_CATALOG. To do this, enter the following commands:

```
/restore_permanent_files l=list_file
PUR/restore_catalog $fname($backup_file(backup_file..
PUR../,identifier)) backup_file=backup_file ..
PUR../new_catalog_name=$user.my_catalog
PUR/quit
```

DISPLAY_BACKUP_FILE RESPF Subcommand

Purpose

Displays the contents of a backup file.

Format

DISPLAY_BACKUP_FILE or
DISBF
BACKUP_FILE=file
DISPLAY_OPTION=keyword
NUMBER=keyword or *integer*
STATUS=status variable

Parameters **BACKUP_FILE or BF**

Specifies the file that contains the backup copies of the files and catalogs previously backed up by a BACKUP_PERMANENT_FILE utility session.

DISPLAY_OPTION or DO

Specifies the level of information to be displayed. Options are:

CATALOG_INFO

Displays information on archived files.

IDENTIFIER (I)

Displays the name and type (file or catalog) of each entry on the backup file.

DESCRIPTOR (D)

Displays the following information:

- Record headers maintained on the backup file.
- Version of the backup utility that produced the backup file.
- Date and time the backup file was written.
- Backup utility subcommand that produced the backup file.
- Cycle number of each file cycle.
- Usage count of each file cycle.
- Creation date and time of each file cycle.
- Last access date and time of each file cycle.
- Date and time of the last modification of each file cycle.
- Expiration date of each file cycle.
- Size of each file cycle.

READ_DATA (RD)

Displays the information described for the DESCRIPTOR parameter and also attempts to read all data for each cycle on the backup file. The listing reports whether or not the data is read without error. No attempt is made to verify the data with the original file backed up.

If omitted, IDENTIFIER is assumed.

NUMBER or *N*

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 if the keyword value ALL is specified, all entries on the backup file are displayed.

Remarks For more information, see the NOS/VE System Usage manual.

INCLUDE_CYCLES RESPF Subcommand

Purpose Includes cycles in subsequent restore operations based on the creation date and time, last access date and time, last modification date and time, or expiration date of the cycle.

Format

INCLUDE_CYCLES or
INCLUDE_CYCLE or
INCC
SELECTION_CRITERIA=keyword
AFTER=date_time
BEFORE=date_time
STATUS=status variable

Parameters **SELECTION_CRITERIA** or **SC**

Specifies the selection criteria to be used in determining which cycles will be restored on subsequent restore operations. Choose one of the following:

ACCESSED (A)

Select files based on the date and time they were last accessed.

CREATED (C)

Select files based on the date and time they were created.

EXPIRED (E)

Select files based on their expiration dates and times.

MODIFIED (M)

Select files based on the date and time they were last modified.

IGNORE_DATE_TIME (IDT)

Do not select files based on a date and time. This option turns off any criteria that may have been selected in previous INCLUDE_CYCLES commands.

This parameter is required.

AFTER or *A*

Specifies the date and time after which the SELECTION_CRITERIA operation must have occurred in order for a file to be included in subsequent restore operations. If omitted, 1980-01-01.00:00:00.000 is used.

BEFORE or *B*

Specifies the date and time before which the SELECTION_CRITERIA operation must have occurred in order for a file to be included in subsequent restore operations. If omitted, $NOW is used.

Remarks

- The values specified on this command take precedence over any previous calls to INCLUDE_CYCLES.
- For more information, see the NOS/VE System Usage manual.

QUIT
RESPF Subcommand

Purpose Ends a RESTORE_PERMANENT_FILES utility session.

Format

```
QUIT or
QUI
```

Parameters None.

Remarks For more information, see the NOS/VE System Usage manual.

RESTORE_ALL_FILES RESPF Subcommand

Purpose Enables a system operator to restore all catalogs and all permanent files for a NOS/VE system (those written to the backup file with the BACKUP_ALL_FILES subcommand). Other users can restore all catalogs which they own and all files and cycles for which they have cycle permission.

Format

RESTORE_ALL_FILES or
RESAF
BACKUP_FILE=**file**
STATUS=status variable

Parameters

BACKUP_FILE or **BF**

Specifies the file that contains the backup copies of the files and catalogs to be restored. This parameter is required.

Remarks

- Backup copies of catalogs and files that do not already exist in the permanent file system are restored.
- Catalogs and files that already exist are not altered.
- The file specified by the BACKUP_FILE parameter is initially positioned to beginning-of-information.
- To restore permanent files when partial backups have been taken, the RESTORE_ALL_FILES subcommand 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 last partial backup. File cycle data that is not contained on the last partial back is restored using the RESTORE_EXCLUDED_FILE_CYCLES subcommand.
- For more information, see the NOS/VE System Usage manual.

Examples The following job restores all files in the system that were previously backed up with a BACKUP_ALL_FILES subcommand:

```
/job
job/request_magnetic_tape file=pf_tape_file ..
job../evsn='pfb001' type=mt9$6250
job/restore_permanent_files
job/restore_all_files backup_file=pf_tape_file
job/quit
job/jobend
```


RESTORE_CATALOG RESPF Subcommand

Purpose Restores a catalog that does not currently exist as a catalog.

Format

RESTORE_CATALOG or
RESC
CATALOG=file
BACKUP_FILE=file
NEW_CATALOG_NAME=file
STATUS=status variable

Parameters

CATALOG or **C**

Specifies the catalog that is to be restored from the backup file. This parameter is required.

BACKUP_FILE or **BF**

Specifies the file that contains the backup copy of the catalog and its associated files and subcatalogs. This parameter is required.

NEW_CATALOG_NAME or *NCN*

Specifies the catalog into which the files and subcatalogs on the backup file are restored. Omission causes the name as it exists on the backup file to be used.

Remarks

- Backup copies of files and subcatalogs are restored.
- You must be the owner of the catalog.
- This command cannot be used to restore your master catalog.
- The catalog being restored must not currently exist.
- The file specified by the BACKUP_FILE parameter must have been created by the BACKUP_PERMANENT_FILE utility.
- The backup file is initially positioned at beginning-of-information.
- For more information, see the NOS/VE System Usage manual.

Examples The following example restores the master catalog to a new subcatalog in the master catalog:

```
/restore_permanent_files list=restore_listing
PUR/restore_catalog catalog=$user new_catalog_name=..
PUR../$user.catalog_2 backup_file=backed_up_files
PUR/quit
```

RESTORE_EXCLUDED_FILE_CYCLES RESPF Subcommand

Purpose Restores cycles to files that currently exist in the permanent file system but do not have data defined for them.

Format

RESTORE_EXCLUDED_FILE_CYCLES or
RESTORE_EXCLUDED_FILE_CYCLE or
RESEFC
FILE=file
CATALOG=file
BACKUP_FILE=file
NEW_NAME=file
RESTORE_OPTIONS=list of keyword
STATUS=status variable

Parameters

FILE or *F*

The FILE parameter specifies the file or cycle for which data is to be restored (as identified on the backup file). If no cycle number is specified, data for all cycles of the file is restored. If specified, a cycle number must be a specific cycle (not $HIGH or $LOW).

CATALOG or *C*

The CATALOG parameter specifies the catalog for which data is to be restored (as identified on the backup file). Data for all cycles in the catalog is restored.

BACKUP_FILE or **BF**

Specifies the file containing the backup information. This file is positioned at the beginning-of-information. This parameter is required.

NEW_NAME or *NCN* or *NEW_CATALOG_NAME* or *NEW_FILE_NAME* or *NN* or *NFN*

Specifies a new name for the catalog, file, or cycle for which the data is being restored. This parameter can be used if the name on the backup file is different than that in the current permanent file system. Omission causes 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, $HIGH is used.

RESTORE_OPTIONS or *RESTORE_OPTION* or *RO*

Reserved for site personnel.

Remarks

- This subcommand is used to restore cycle data when partial backups have been performed. If the permanent file system is backed up by a full backup followed by daily partial backups, then the last partial backup is restored with the RESTORE_ALL_FILES subcommand. All other backups are restored in reverse order using this subcommand.
- The modification date on the backup file must match the modification date in the current permanent file catalog, unless otherwise specified by a SET_RESTORE_OPTIONS subcommand.
- If a cycle already has data defined for it, the cycle is not altered.

- You may specify either the file or catalog parameter, but not both. Omission of both parameters causes all data to be restored, in which case the NEW_NAME parameter cannot be used.
- For more information, see the NOS/VE System Usage manual.

Examples The following example restores files from a previous partial dump and a previous full dump.

```
/restore_permanent_files
PUR/restore_all_files bf=partial_dump
PUR/restore_excluded_file_cycles bf=full_dump
```

The following example restores all cycles of a file from a partial and full dump.

```
PUR/restore_file $user.data_file_1 bf=partial_dump
PUR/resefc file=$user.data_file_1 bf=full_dump
```

RESTORE_EXISTING_CATALOG RESPF Subcommand

Purpose Restores the contents of a currently existing catalog.

Format

RESTORE_EXISTING_CATALOG or
RESEC
CATALOG=**file**
BACKUP_FILE=**file**
NEW_CATALOG_NAME=*file*
STATUS=*status variable*

Parameters

CATALOG or **C**

Specifies the catalog that is to be restored from the backup file. This parameter is required.

BACKUP_FILE or **BF**

Specifies the file that contains the backup copy of the catalog and its associated files and subcatalogs. The file is initially positioned at beginning-of-information. This parameter is required.

NEW_CATALOG_NAME or *NCN*

Specifies the existing catalog into which the files and subcatalogs on the backup file are restored. Omission causes the name as it exists on the backup file to be used.

Remarks

- Backup copies of files and subcatalogs that do not already exist in the specified catalog are restored.
- Any cycle that already exists is not altered.
- Cycle permission is required to restore any file cycle within an existing catalog.
- You must be the owner of the catalog to restore any subcatalogs.
- The file specified by the BACKUP_FILE parameter must have been created by the BACKUP_PERMANENT_FILE utility.
- The backup file is initially positioned at beginning-of-information.
- For more information, see the NOS/VE System Usage manual.

Examples . The following commands restore the master catalog that was backed up with the BACKUP_CATALOG subcommand:

```
/restore_permanent_files list=restore_list
PUR/restore_existing_catalog ..
PUR../catalog=$user backup_file=backed_up_files
PUR/quit
```

RESTORE_EXISTING_FILE RESPF Subcommand

Purpose Restores the file cycles of an existing file.

Format

RESTORE_EXISTING_FILE or
RESEF
FILE=file
BACKUP_FILE=file
PASSWORD=keyword or *name*
NEW_FILE_NAME=file
STATUS=status variable

Parameters

FILE or **F**

Specifies the file whose file cycles are to be restored from the backup file. If a cycle reference is included the cycle reference is ignored. This parameter is required.

BACKUP_FILE or **BF**

Specifies the file that contains the backup copy of the file. This parameter is required.

PASSWORD or *PW*

Parameter Attributes: SECURE

Specifies the file password. This parameter must match the password of the existing file. Omission or specifying the keyword NONE causes no password to be used.

NEW_FILE_NAME or *NFN*

Specifies the existing file to be restored. Omission causes the name as it exists on the backup file to be used.

Remarks

- All file cycles that exist on the backup file but do not exist as a permanent file are restored.
- Cycles that currently exist as permanent files are not altered.
- You must have CYCLE permission to restore an existing file.
- The file specified by the BACKUP_FILE parameter must have been created during by BACKUP_PERMANENT_FILE utility.
- The backup file is initially positioned at beginning-of-information.
- For more information, see the NOS/VE System Usage manual.

Examples The following example restores cycle number 87 of file DATA_FILE_0 in subcatalog CATALOG_1 of the master catalog that was previously backed up:

```
/delete_file $user.catalog_1.data_file_0.87 ..
../pw=new_data_0_pw
/respf
PUR/restore_existing_file ..
PUR../$user.catalog_1.data_file_0 ..
PUR../bf=copy_of_file pw=new_data_0_pw
PUR/quit
```


RESTORE_FILE RESPF Subcommand

Purpose Restores the file cycles of a file that does not currently exist as a permanent file.

Format

RESTORE_FILE or
RESF
FILE=file
BACKUP_FILE=file
PASSWORD=keyword or *name*
NEW_FILE_NAME=file
STATUS=status variable

Parameters

FILE or F

Specifies the file whose file cycles are to be restored from the backup file. This parameter is required.

BACKUP_FILE or BF

Specifies the file that contains the backup copy of the file. This parameter is required.

PASSWORD or PW

Parameter Attributes: SECURE

Specifies the file password. It must match the existing file password. This parameter is used only if a specific cycle of an existing file is being restored. Omission or specifying the keyword NONE causes no password to be used.

NEW_FILE_NAME or *NFN*

Specifies a new name for the file being restored. Omission causes the name as it exists on the backup file to be used.

Remarks

- If the file name includes a cycle reference, only that cycle is restored (at least one file cycle must already exist).
- If a cycle reference is omitted, all file cycles are restored (the file must not already exist).
- If a cycle reference is included on the FILE parameter, it must be a specific cycle number; the keywords $HIGH and $LOW cannot be used.
- If a cycle reference is not specified on the NEW_FILE_NAME parameter, $NEXT is used.
- If a cycle reference is specified on the NEW_FILE_NAME parameter, the file specified with the FILE parameter must also include a cycle reference.
- You must have CYCLE permission to the file in order to restore all file cycles or an additional file cycle.
- The file specified by the BACKUP_FILE parameter must have been created by the BACKUP_PERMANENT_FILE utility.
- The backup file is initially positioned at beginning-of-information.
- For more information, see the NOS/VE System Usage manual.

Examples

The following subcommands restore cycle number 87 of file DATA_FILE_0 in subcatalog CATALOG_1. The file is restored as cycle number 1 of file DATA_FILE_2 in CATALOG_2 of the master catalog.

```
/respf
PUR/restore_file $user.catalog_1.data_file_0.87 ..
PUR../bf=copy_of_file pw=new_data_0_pw ..
PUR../nfn=$user.catalog_2.data_file_2
PUR/quit
```

SET_LIST_OPTIONS RESPF Subcommand

Purpose Specifies the information that is written to the list file by subsequent subcommands.

Format

SET_LIST_OPTIONS or
SET_LIST_OPTION or
SETLO
FILE_DISPLAY_OPTIONS=list of keyword
CYCLE_DISPLAY_OPTIONS=list of keyword
DISPLAY_EXCLUDED_ITEMS=boolean
STATUS=status variable

Parameters

FILE_DISPLAY_OPTIONS or *FILE_DISPLAY_OPTION* or *FDO*

Selects the data to be displayed with the file name. Options are:

ACCOUNT (A)

Displays the account name.

PROJECT (P)

Displays the project name.

NONE

Displays only the file name.

ALL

Displays the account and project name.

Omission causes NONE to be used.

CYCLE_DISPLAY_OPTIONS or
CYCLE_DISPLAY_OPTION or *CDO*

Selects the data to be displayed for each cycle backed up. The cycle number and whether the cycle was excluded is also displayed. Options are:

ALL

Selects all of the following.

ACCESS_COUNT (AC)

Displays the number of accesses to the cycle.

ACCESS_DATE_TIME (ADT)

Displays the date and time the cycle was last accessed.

ALTERNATE_FILE_MEDIA_DESCRIPTOR (AFMD)

Displays archive information.

CREATION_DATE_TIME (CDT)

Displays the date and time the cycle was created.

EXPIRATION_DATE (ED)

Displays the expiration date of the cycle.

GLOBAL_FILE_NAME (GFN)

Displays the internally generated global file name. This name is neither backed up nor restored.

MODIFICATION_DATE_TIME (MDT)

Displays the date and time the cycle was last modified.

NONE

Displays the cycle number.

RECORDED_VSN (RVSN)

Displays all mass storage volumes on which the cycle resides.

SIZE (S)

Displays the size of the cycle in bytes.

Omission causes (MODIFICATION_DATE_TIME, SIZE) to be used.

DISPLAY_EXCLUDED_ITEMS or
DISPLAY_EXCLUDED_ITEM or *DEI*

Specifies whether excluded catalogs, files, and cycles are displayed on the list file.

TRUE

The identification of all excluded catalogs, files, and cycles is displayed. This is the default.

FALSE

Excluded items are not displayed.

Remarks For more information, see the NOS/VE System Usage manual.

SET_RESTORE_OPTIONS RESPF Subcommand

Purpose Specifies the options to be used in subsequent restore operations.

Format

SET_RESTORE_OPTIONS or
SET_RESTORE_OPTION or
SETRO
REQUIRE_MATCHING_MODIFICATION=keyword or *boolean*
ALLOCATION_SIZE=keyword or *integer*
FILE_CLASS=keyword or *name*
INITIAL_VOLUME=keyword or *name*
UPDATE_CYCLE_STATISTICS=keyword or *boolean*
VOLUME_OVERFLOW_ALLOWED=keyword or *boolean*
RESTORE_ARCHIVE_INFORMATION=keyword or *boolean*
STATUS=status variable

Parameters *REQUIRE_MATCHING_MODIFICATION* or *RMM*

Specifies whether or not the RESTORE_EXCLUDED_FILE_CYCLES subcommand will restore a cycle whose modification date and time recorded in the backup file does not match the file's catalog information. Values can be:

TRUE

A file cycle is restored only if the last modification date and time in the catalog matches the one in the backup file.

FALSE

A file cycle is restored regardless of the last modification.

$UNSPECIFIED

Acts as if no SET_RESTORE_OPTIONS were issued for this parameter.

If this parameter is omitted, TRUE is assumed.

ALLOCATION_SIZE or *AS*

Reserved for site personnel.

FILE_CLASS or *FC*

Reserved.

INITIAL_VOLUME or *IV*

Reserved for site personnel.

UPDATE_CYCLE_STATISTICS or *UCS*

Reserved.

VOLUME_OVERFLOW_ALLOWED or *VOA*

Reserved for site personnel.

RESTORE_ARCHIVE_INFORMATION or *RAI*

Reserved for site personnel.

Remarks

- Once you enter this subcommand, the options selected remain in effect for the rest of the session or until you enter this subcommand again.
- For more information, see the NOS/VE System Usage manual.

SOURCE_CODE_UTILITY 33

SOURCE_CODE_UTILITY Command

Purpose Begins an SCU command utility session.

Format

SOURCE_CODE_UTILITY or
SCU or
SOUCU
STATUS=status variable

Remarks

- Entering a CREATE_LIBRARY or USE_LIBRARY subcommand initializes the working library for the SCU command utility session. If neither subcommand is issued, file SOURCE_LIBRARY is used for the base and result libraries. If file SOURCE_LIBRARY does not exist, it is created.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following sequence begins an SCU session and initializes the working library from file OLDPL in your working catalog, assumed not to be $LOCAL. The base file, OLDPL, is a source file whose file structure is a library. Entering the QUIT subcommand causes the working library to be written on the next cycle of file OLDPL.

```
/source_code_utility
sc/use_library b=oldpl r=oldpl.$next
sc/quit
```

The next example does not use the USE_LIBRARY subcommand, but rather initializes the working library from file SOURCE_LIBRARY in your working catalog.

```
/source_code_utility
sc/create_deck d=deck1 m=version1
sc/quit
```

ADD_LIBRARY SCU Subcommand

Purpose Adds decks from one or more source libraries to the working library.

Format

ADD_LIBRARY or
ADD_LIBRARIES or
ADDL
SOURCE_LIBRARY=**list of file**
LIST=*file*
DISPLAY_OPTIONS=*keyword*
STATUS=*status variable*

Parameters

SOURCE_LIBRARY or **SOURCE_LIBRARIES** or **SL**

List of one or more source library files. This parameter is required.

LIST or *L*

Listing file. You can specify a file position as part of the file name. SCU lists the source library origin of each deck in the working library. Within an SCU session, if you omit LIST, the listing file is the file specified on the SET_LIST_OPTIONS subcommand. Otherwise, the default is file $LIST.

DISPLAY_OPTIONS or *DO*

Specifies the level of information listed. Currently, both keyword values produce the same listing.

BRIEF or B
FULL or F

If DISPLAY_OPTIONS is omitted, BRIEF is used. ALL is an alias for FULL.

Remarks

- ADD_LIBRARIES only adds decks that are not already in the working library. It reads the deck list for each source library in the order you specify the libraries on the command. When it reads a deck name that is not currently in the working library, it adds the deck to the library. When it reads a deck name that is already in the working library, it sends a message describing the duplication, but it does not add the deck to the working library.

- If a modification is in more than one source library modification list and the creation times do not match, ADD_LIBRARY reports an error and does not add any decks to the working library.
- If no decks could be merged because an exception occurred in each deck, an error status is returned and ADD_LIBRARY makes no change to the library.
- Decks, features, groups, and modifications are ordered alphabetically on the ADD_LIBRARIES result library.
- Key characters in source libraries that are added to the working library must match the key character in the working library. If the key characters do not match, SCU generates an error message.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following command adds the decks from the source library on file $USER.NEWLIB to the working library. The contents of the working library are then displayed.

```
sc/add_library sl=$user.newlib l=output
 DECKA                    :NVE.PAT.BASE
 DECKB                    :NVE.PAT.BASE
 DECKC                    :NVE.PAT.NEWLIB
 DECKD                    :NVE.PAT.BASE
```

CHANGE_DECK
SCU Subcommand

Purpose Changes the content of one or more deck header fields.

Format

CHANGE_DECK or
CHANGE_DECKS or
CHAD
DECK=keyword or *list of: name* or *range of name*
AUTHOR=string
CLEAR_ORIGINAL_INTERLOCK=boolean
CLEAR_SUB_INTERLOCK=boolean
DECK_DESCRIPTION=list of string
PROCESSOR=string
GROUP=list of name
DELETE_GROUP=list of name

CHARACTER=keyword or *string*
TAB_COLUMN=list of integer
DELETE_COLUMN=list of: integer or *range of integer*
WIDTH=integer
LINE_IDENTIFIER=keyword
EXPAND=boolean
STATUS=status variable

Parameters *DECK* or *DECKS* or *D*

Decks whose headers are changed. You can specify a list of one or more names, a list of one or more ranges, or the keyword ALL. ALL specifies all decks in the library. The default is the name of the most recently used deck.

AUTHOR or *A*

New author. If AUTHOR is omitted, the author field is not changed.

CLEAR_ORIGINAL_INTERLOCK or *COI*

Indicates whether the original interlock for an extracted deck should be cleared. Options are:

TRUE

Clears the original interlock field of the extracted deck by erasing the name and time stamp that were recorded in this deck.

FALSE

Leaves the original interlock field of the extracted deck unchanged.

If CLEAR_ORIGINAL_INTERLOCK is omitted, FALSE is used.

CLEAR_SUB_INTERLOCK or *CI* or
CLEAR_INTERLOCK or *CSI*

Indicates whether the subinterlock field of the original deck should be cleared. Options are:

TRUE

Clears the subinterlock field of the original deck.

FALSE

Leaves the subinterlock field of the original deck unchanged.

If CLEAR_SUB_INTERLOCK or CLEAR_INTERLOCK is omitted, FALSE is used.

NOTE

You must have authority 4 for the file to clear a deck subinterlock or original interlock field.

DECK_DESCRIPTION or *DD*

List of strings containing the new deck description. If DECK_DESCRIPTION is omitted, the description field is not changed.

PROCESSOR or *P*

New processor. If PROCESSOR is omitted, the processor field is not changed.

GROUP or *GROUPS* or *G*

Additional groups to which the deck is to belong. The subcommand deletes any groups specified on the DELETE_GROUP parameter before adding groups to the group list. If GROUP is omitted, the deck is not associated with additional groups.

DELETE_GROUP or *DELETE_GROUPS* or *DG*

Groups to which the deck should no longer belong. The subcommand deletes groups specified before adding any groups specified on the GROUP parameter. If DELETE_GROUP is omitted, the deck continues to belong to the same groups it did previously.

CHARACTER or *C*

Either a 1-character string containing the new default tab character or the keyword NONE to disable tabbing. If CHARACTER is omitted, the tabbing status and default tab character are not changed.

TAB_COLUMN or *TAB_COLUMNS* or *TC*

List of from 1 to 256 additional default tab columns. SCU deletes the tab columns on the DELETE_COLUMN parameter before it adds the new tab columns. If TAB_COLUMN is omitted, no new tab columns are added.

DELETE_COLUMN or *DELETE_COLUMNS* or *DC*

List of default tab columns or tab column ranges to be removed. SCU deletes the specified tab columns before it adds the tab columns on the TAB_COLUMN parameter. If DELETE_COLUMN is omitted, no tab columns are removed.

WIDTH or *W*

New default line width. If WIDTH is omitted, the default line width is not changed.

LINE_IDENTIFIER or *LI*

New default line identifier placement. Options are:

RIGHT (R)

Place line identifiers to the right of the text.

LEFT (L)

Place line identifiers to the left of the text.

NONE

No line identifiers are placed on output lines.

If LINE_IDENTIFIER is omitted, the default line identifier placement is not changed.

EXPAND or *E*

New expand attribute value. Options are:

TRUE

An EXPAND_DECK subcommand expands the deck. (The deck can also be expanded by a COPY or COPYC directive.)

FALSE

An EXPAND_DECK subcommand does not expand the deck; it skips the deck and continues processing at the next deck. Only a COPY or COPYC directive can expand the deck.

If EXPAND is omitted, the expand attribute is not changed.

Remarks

- The DECK parameter specifies each deck to which the changes should apply. The other parameters (except STATUS) specify the deck header fields to be changed.
- To display a deck header, enter a DISPLAY_DECK subcommand. You can reference individual deck header attributes with the SCU function $DECK_ATTRIBUTES.
- If you have access authority 4 for the file, you can enter a CHANGE_DECK subcommand to clear a subinterlock that was set when a user extracted a deck from the library.
- You can remove unused groups from a library explicitly using the DELETE_GROUP subcommand or implicitly using the EXTRACT_SOURCE_LIBRARY subcommand. Specifying DECKS=ALL and INTERLOCK=NONE on the EXTRACT_SOURCE_LIBRARY subcommand copies all decks to a new RESULT file, saving only groups, modifications, and features belonging to those decks.
- Changes to a deck header are not part of any modification. When you include or exclude modifications, you must make any associated deck header changes separately by entering the CHANGE_DECK subcommand.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand adds default tab column 35 and deletes default tab column 30 for DECK1.

```
sc/change_deck d=deck1 tc=35 dc=30
```

The following subcommand clears the subinterlock fields of all deck headers in the working library if you have access authority 4 for the file.

```
sc/change_deck d=all csi=true
```

CHANGE_DECK_NAME SCU Subcommand

Purpose Substitutes new names for existing deck names.

Format

CHANGE_DECK_NAME or
CHANGE_DECK_NAMES or
CHADN
NAME_LIST=file
LIST=file
CHANGE_DECK_REFERENCES=boolean
MODIFICATION=name
STATUS=status variable

Parameters

NAME_LIST or **NL**

Name substitution file. This parameter is required.

LIST or *L*

Listing file. You can specify a file position as part of the file name. If LIST is omitted, the listing file is the file specified on the SET_LIST_OPTIONS subcommand. Otherwise, the default is file $LIST.

CHANGE_DECK_REFERENCES or *CDR*

Indicates whether the command substitutes deck names on COPY and COPYC directives. Options are:

TRUE

COPY and COPYC names are substituted.

FALSE

COPY and COPYC names are not substituted.

If CHANGE_DECK_REFERENCES is omitted, FALSE is used.

MODIFICATION or *M*

Modification to which the changed lines belong. If MODIFICATION is omitted, SCU$ALTER is used.

Remarks

- A deck name can occur in two places within a source library: within its deck header, and on COPY and COPYC directives in the source text. To list the COPY and COPYC references to the deck, enter a DISPLAY_DECK_REFERENCES command.

- You store the name substitutions on a separate file and specify the file on the NAME_LIST parameter. Each name substitution is specified as a line containing an SCL parameter list. The parameter list must have the following parameters:

  OLD_NAME (ON)

  Existing name.

  NEW_NAME (NN)

  Substituted name. NEW_NAME must be different from ALL.

- For more information, see the NOS/VE Source Code Management manual.

Examples

The following subcommand changes deck names as specified in file NEW_DECK_NAMES. The changed lines belong to modification RENAME.

```
sc/change_deck_names nl=new_deck_names m=rename ..
sc../cdr=true
```

The contents of file NEW_DECK_NAMES are:

```
my_deck,deck465
```

The command replaces each occurrence of the deck name MY_DECK with the new name DECK465. Because the command specifies that the

CHANGE_DECK_REFERENCES parameter is TRUE, it replaces the deck name both in the deck header and on COPY and COPYC directives throughout the library.

CHANGE_DECK_REFERENCES SCU Subcommand

Purpose Changes the deck names of COPY and COPYC directives that are located in the specified decks.

Format

CHANGE_DECK_REFERENCES or
CHADR
DECK=keyword or *list of: name* or *range of name*
MODIFICATION=name
NAME_LIST=file
LIST=file
STATUS=status variable

Parameters

DECK or *DECKS* or *D*

Decks in which substitutions are performed. The keyword ALL specifies all decks in the library. If DECK is omitted, ALL is used.

MODIFICATION or *M*

Modification to which the changed lines belong. If MODIFICATION is omitted, SCU$ALTER is used.

NAME_LIST or **NL**

Name substitution file. This parameter is required.

LIST or *L*

Listing file. You can specify a file position as part of the file name. If LIST is omitted, the listing file is the file specified on the SET_LIST_OPTIONS subcommand. Otherwise, the default is file $LIST.

Remarks

- The CHANGE_DECK_REFERENCES subcommand only changes deck names on COPY and COPYC directives, not in deck headers. To change a deck name in its deck header, enter the CHANGE_DECK_NAMES command.

- You use CHANGE_DECK_REFERENCES to replace references to one deck with references to another deck. To list the COPY and COPYC references to a deck, enter a DISPLAY_DECK_REFERENCES command.

- You store the name substitutions on a separate file and specify the file on the NAME_LIST parameter. Each name substitution is specified as a line containing an SCL parameter list. The parameter list must have the following parameters:

  OLD_NAME (ON)

  Existing name.

  NEW_NAME (NN)

  Substituted name. NEW_NAME should be different from ALL.

- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand changes references as specified in file NEW_NAMES. The changes belong to modification RENAME.

```
sc/change_deck_references nl=new_names m=rename
```

The following lists the contents of file NEW_NAMES.

```
deck44,deck45
```

The command changes each COPY or COPYC reference to DECK44 so that it references DECK45.

CHANGE_LIBRARY SCU Subcommand

Purpose Changes the content of one or more fields in the working library header.

Format

CHANGE_LIBRARY or
CHAL
LIBRARY=name
LIBRARY_DESCRIPTION=list of string
VERSION=string
LAST_USED_DECK=name
LAST_USED_MODIFICATION=name
STATUS=status variable

Parameters

LIBRARY or *L*

New library name. If LIBRARY is omitted, the library name is not changed.

LIBRARY_DESCRIPTION or *LD*

Strings used to describe the source code that is maintained on this library. If LIBRARY_DESCRIPTION is omitted, the description field is not changed.

33

VERSION or *V*

New library version. If VERSION is omitted, the version field is not changed.

LAST_USED_DECK or *LUD*

Default deck name that is stored in the library header. The deck name is used as the default value for the deck parameter on most subcommands. Specifying NONE clears the last used deck name. If a name is explicitly stated for a DECK parameter on an SCU subcommand, LAST_USED_DECK is automatically changed.

LAST_USED_MODIFICATION or *LUM*

Default modification name that is stored in the library header. The modification name is used as the default value for the modification parameter on most subcommands. Specifying NONE clears the last used modification name. If a name is explicitly stated for a MODIFICATION parameter on an SCU subcommand, LAST_USED_MODIFICATION is automatically changed to that name.

Remarks

- To display the contents of the library header, enter a DISPLAY_LIBRARY command.
- You can reference individual library header attributes with the SCU function $LIBRARY_ATTRIBUTES.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following command changes the content of the library version field.

```
sc/change_library v='Version 1.1'
```

CHANGE_MODIFICATION SCU Subcommand

Purpose Changes information in one or more modification descriptions.

Format

CHANGE_MODIFICATION or
CHANGE_MODIFICATIONS or
CHAM
MODIFICATION=keyword or *list of: name* or *range of name*
FEATURE=keyword or *name*
AUTHOR=string
MODIFICATION_DESCRIPTION=list of string
STATE=integer
STATUS=status variable

Parameters

MODIFICATION or *MODIFICATIONS* or *M*

Modification descriptions to be changed. You can specify a list of one or more names (from 1 to 9 characters each), a list of one or more ranges, or the keyword ALL. ALL specifies all modifications in the library. If MODIFICATION is omitted, the information for the description of the last used modification is changed.

FEATURE or *F*

New feature name or keyword NONE. Specifying NONE clears the current feature association. If FEATURE is omitted, the feature field is not changed.

AUTHOR or *A*

New author. If AUTHOR is omitted, the author field is not changed.

MODIFICATION_DESCRIPTION or *MD*

Strings used to describe the modifications. If MODIFICATION_DESCRIPTION is omitted, the description field is not changed.

STATE or *S*

New modification state. The following are the states and their descriptions.

State	Description
0	Experimental
1	Developmental
2	Stable
3	Verified
4	Released

If STATE is omitted, the state is not changed.

NOTE

You cannot raise the modification state above your authority for the file.

Remarks

- The CHANGE_MODIFICATIONS subcommand can only change the headers of modifications within the modification list of the working library.
- To raise the value in the state field of the modification header, your authority for the library file must be the same or greater than the new state. For example, to raise the state to 2, your authority must be 2, 3, or 4.

  You can only lower a state to 0. To lower the state to 0, your authority for the library file must be the same or greater than the current state. For example, to lower a modification that is currently in state 2, your authority must be 2, 3, or 4.

- To display a modification header, enter a DISPLAY_MODIFICATION command. You can reference individual modification header attributes with the SCU function $MODIFICATION_ATTRIBUTES.

- You can remove unused features from a library explicitly using the DELETE_FEATURE subcommand or implicitly using the EXTRACT_SOURCE_LIBRARY subcommand. Specifying DECKS=ALL and INTERLOCK=NONE on the EXTRACT_SOURCE_LIBRARY subcommand copies all decks to a new RESULT file, saving only groups, modifications, and features that belong to those decks.

- The feature should not be named ALL or NONE.

- For more information, see the NOS/VE Source Code Management manual.

Examples The following command clears the feature associations of all modifications in the working library.

```
sc/change_modification m=all f=none
```

The following command raises the state of MOD_4 to state 1 (developmental). You must have at least authority 1 for the file to raise the modification state to 1.

```
sc/change_modification m=mod_4 s=1
```

COMBINE_LIBRARY SCU Subcommand

Purpose Combines the decks from one or more source libraries with those in the working library.

Format

COMBINE_LIBRARY or
COMBINE_LIBRARIES or
COML
SOURCE_LIBRARY=list of file
LIST=file
DISPLAY_OPTIONS=keyword
ENFORCE_INTERLOCKS=boolean
STATUS=status variable

Parameters

SOURCE_LIBRARY or SOURCE_LIBRARIES or SL

List of one or more source library names. This parameter is required.

LIST or *L*

Listing file. You can specify a file position as part of the file name. SCU lists the source library origin of each deck in the working library. If LIST is omitted, the listing file is the file specified on the SET_LIST_OPTIONS subcommand. Otherwise, the default is file $LIST.

DISPLAY_OPTIONS or *DO*

Specifies the information listed. Currently, both of the following keywords produce the same listing.

BRIEF or B
FULL or F

If DISPLAY_OPTIONS is omitted, BRIEF is used. ALL is an alias for FULL.

ENFORCE_INTERLOCKS or *EI*

Indicates whether the original interlock field of a source library deck must match the subinterlock field of the working library deck it is to replace. Options are:

TRUE

Interlocks must match.

FALSE

Interlocks need not match.

If ENFORCE_INTERLOCKS is omitted, FALSE is used.

Remarks

- COMBINE_LIBRARY reads the source library deck lists in the order you specify the libraries on the command.
- After reading a deck name, COMBINE_LIBRARY determines if the deck name is already in the working library deck list. If the name is not in the list, it adds the deck to the working library. If the name is already in the list, it replaces the deck in the working

library with the deck from the source library. The combining process is continued until each successive source library in the list has been combined with the working library.

- If no decks could be merged because an exception occurred in each deck, an error status is returned and no change is made to the library.

  If the creation times of modifications that occur on both libraries do not match, COMBINE_LIBRARY issues an error and does not alter the working library.

- COMBINE_LIBRARY lists the source library origin of each deck in the working library on the listing file.

- Decks, features, groups, and modifications are ordered alphabetically on the COMBINE_LIBRARY result library.

- You can enter a COMBINE_LIBRARY subcommand to merge decks from an extracted library with the decks in the library from which it was extracted to form a new library. It adds new decks and replaces existing decks.

- If you set interlocks when you extract the library, entering COMBINE_LIBRARY enforces the interlock if you specify that the interlocks should be enforced. COMBINE_LIBRARY checks whether the original interlock value in the extracted deck header matches the subinterlock value in the working library header. If the values match, the working library deck is replaced with the extracted deck. Otherwise, it issues a warning message, does not replace the working library, and then attempts to combine any remaining decks in the list.

- Key characters in source libraries that are added to the working library must match the key character in the working library. If the key characters do not match, SCU generates an error message.

- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand combines the decks in the source library NEWLIB with the decks in the working library.

```
sc/combine_library sl=newlib l=output
 DECKA                    :NVE.PAT.BASE
 DECKB                    :NVE.PAT.BASE
 DECKC                    :NVE.PAT.NEWLIB
 DECKD                    :NVE.PAT.BASE
 DECKE                    :NVE.PAT.NEWLIB
```

CREATE_DECK
SCU Subcommand

Purpose Creates one or more decks.

Format

CREATE_DECK or
CREATE_DECKS or
CRED
DECK=list of name
MODIFICATION=name
SOURCE=list of file
AUTHOR=string
DECK_DESCRIPTION=list of string
PROCESSOR=string
GROUP=list of name
CHARACTER=keyword or *string*
TAB_COLUMN=list of integer
WIDTH=integer
LINE_IDENTIFIER=keyword
EXPAND=boolean
DECK_DIRECTIVES_INCLUDED=boolean
MULTI_PARTITION=boolean
SAME_AS=name
STATUS=status variable

Parameters *DECK* or *DECKS* or *D*

List of one or more deck names. Each name must be unique to the library. If DECK is omitted, you must specify the SOURCE parameter and DECK_DIRECTIVES_INCLUDED=TRUE.

MODIFICATION or *M*

Modification name (1 to 9 characters). The modification must be in state 0 (zero). The default is the last used modification.

SOURCE or *SOURCES* or *S*

List of one or more files containing the source text for the decks. You can specify a file position as part of the file name. The SOURCE parameter is required when you specify DECK_DIRECTIVES_INCLUDED=TRUE.

AUTHOR or *A*

Optional author identification.

DECK_DESCRIPTION or *DD*

List of strings containing the optional deck description. If DECK_DESCRIPTION is omitted, a description is not saved.

PROCESSOR or *P*

Optional identification of the processor to which the deck text is input.

GROUP or *GROUPS* or *G*

Optional list of groups to which the deck is to belong. If any of the group names are not in the group list, SCU adds the names to the list.

CHARACTER or *C*

Either a 1-character string containing the tab character or the keyword NONE to disable tabbing. If CHARACTER is omitted, tabbing is disabled.

TAB_COLUMN or *TAB_COLUMNS* or *TC*

Optional list of 1 through 256 default tab columns. The column numbers range from 1 through 256.

WIDTH or *W*

Default line width. If WIDTH is omitted or specified as 0 (zero), deck lines can be up to 256 characters and the lines are not padded with trailing blanks when the deck is expanded.

LINE_IDENTIFIER or *LI*

Default line identifier placement.

RIGHT (R)

Identifiers are placed to the right of the text.

LEFT (L)

Identifiers are placed to the left of the text.

NONE

No line identifiers are placed on output lines.

If LINE_IDENTIFIER is omitted, NONE is used.

EXPAND or *E*

Specifies the expand attribute for the decks created. Applicable only if the subcommand names decks on its DECK parameter, not if DECK directives name the decks. (A DECK directive specifies the expand attribute for its deck.)

TRUE

An EXPAND_DECK subcommand expands the deck. (COPY and COPYC directives can also expand the deck.)

FALSE

An EXPAND_DECK subcommand skips the deck and continues its processing with the next specified deck. (Only COPY and COPYC directives can expand the deck.)

If EXPAND is omitted, TRUE is used.

DECK_DIRECTIVES_INCLUDED or *DDI*

Indicates whether the deck names are specified on DECK directives embedded in the source text or as the DECK parameter of this subcommand.

TRUE

The deck names are on DECK directives in the source text on the source file. CREATE_DECK only reads text from the first source file specified when DECK directives are included.

FALSE

The deck names shown in the DECK parameter.

If DECK_DIRECTIVES_INCLUDED is omitted, FALSE is used and the DECK parameter must be specified.

MULTI_PARTITION or *MP*

Indicates whether the deck text can be more than one partition of data.

TRUE

The subcommand can copy more than one partition of data to each deck.

FALSE

The subcommand can copy only one partition of data to each deck.

If MULTI_PARTITION is omitted, FALSE is used.

SAME_AS or *SA*

Optional deck name. If a name is specified, the subcommand copies deck header fields not specified on the CREATE_DECK subcommand from the deck header of this deck. If SAME_AS is omitted, unspecified header fields are left blank.

Remarks

- CREATE_DECK provides a header for each deck. The minimum content of the deck header is the deck name and the creation modification. You can specify additional values for deck header fields with parameters on the subcommand. You can also specify the SAME_AS parameter to copy deck header fields from another deck header; CREATE_DECK only copies those deck header fields not explicitly specified.

- Each deck created is given a name (from 1 through 31 characters). By default, the subcommand uses the deck names specified on the DECK parameter. However, if you specify DECK_DIRECTIVES_INCLUDED=TRUE on the subcommand, it uses the deck names specified on DECK directives in the source text. You can specify the expand attribute for a deck on its DECK directive.

- The subcommand can specify the creation modification for the deck. A modification name is from 1 through 9 characters, and it can be an existing modification within the library or a new modification. Any source text that the subcommand copies to a deck belongs to the creation modification. The default is the last used modification.

- To copy source text to the newly created decks, you must specify the SOURCE parameter. If you specify the SOURCE parameter and the DECK parameter, you must specify a file name for each deck name on the DECK parameter. The subcommand copies text to each deck from its corresponding file on the SOURCE parameter; that is, it copies the text from the first file to the first deck created, the text from the second file to the second deck created, and so forth. If you specify the file $NULL for a deck, the subcommand copies no text and the deck remains empty.

- By default, the subcommand copies only the first partition of text from a source text file. To copy more than one partition of text, specify MULTI_PARTITION=TRUE on the subcommand. This indicates that if the subcommand reads an end-of-partition delimiter when copying text, it converts the delimiter to a WEOP text-embedded directive and continues copying text.

- If you specify DECK_DIRECTIVES_INCLUDED=TRUE and omit the DECK parameter, the subcommand creates a deck header for each DECK directive it reads on the source text file.

- If you specify DECK_DIRECTIVES_INCLUDED=TRUE and errors are encountered in the source file, CREATE_DECK attempts to skip ahead to the next DECK directive. The working library will contain the decks that were processed without errors.

- The subcommand places the created decks within the library so that the alphabetic sequence of names in the deck list is maintained.

- The maximum number of lines in one deck is 16,777,214.
- For more information, see the NOS/VE Source Code Management manual.

Examples

The following subcommand creates two decks. First, it creates a deck named DECK2 and copies one partition of text to the deck from file FILE2. It then creates a deck named DECK3 and copies one partition of text to the deck from file FILE3. The deck headers contain the same information as the DECK1 header, except for their description fields.

```
sc/create_deck d=(deck2,deck3) m=original ..
sc../s=(file2,file3) sa=deck1 ..
sc../dd='Second version of INIT_ARRAY'
```

The following subcommand creates decks using the text on file FILE4. SCU generates a deck header for each DECK directive embedded in the file text. The deck headers are the same as the DECK1 header, except for the name and expand attribute fields. The DECK directive specifies the deck name and expand attribute.

```
sc/create_deck m=original s=file4 sa=deck1 ddi=true
```


CREATE_LIBRARY SCU Subcommand

Purpose

Creates an empty source library and specifies the result library to be used.

Format

CREATE_LIBRARY or
CREL
RESULT=file
LIBRARY=name
LIBRARY_DESCRIPTION=list of string
KEY=string
VERSION=string
STATUS=status variable

Parameters

RESULT or *R*

Name of the file to be used as the result library file. If RESULT is omitted, the file SOURCE_LIBRARY in your working catalog is used as the result library file.

LIBRARY or *L*

Library name. If LIBRARY is omitted, the name specified by the RESULT parameter is used as the library name.

LIBRARY_DESCRIPTION or *LD*

String or strings that describe the source code maintained on this library. If LIBRARY_DESCRIPTION is omitted, the null string is used.

KEY or *K*

One-character string containing the key character. The key character is the first character of a text-embedded directive. If KEY is omitted, * is used.

VERSION or *V*

String used to describe the version of the library. If VERSION is omitted, the null string is used.

Remarks

- Using the CREATE_LIBRARY subcommand, you can specify a key character other than the default character *. The key character is the character SCU recognizes as the prefix for all text-embedded directives in the library.
- CREATE_LIBRARY creates a source library containing only a library header, which you can display with the DISPLAY_LIBRARY subcommand. To change library header information, enter a CHANGE_LIBRARY subcommand. To reference a library header field, use the SCU function $LIBRARY_ATTRIBUTES.
- The library created by this subcommand becomes the base library and cannot be changed unless you enter the END_LIBRARY subcommand, then specify another library with either the CREATE_LIBRARY or USE_LIBRARY subcommand.

- During an SCU session, if neither a CREATE_LIBRARY nor a USE_LIBRARY subcommand is issued before other subcommands, file SOURCE_LIBRARY in your working catalog is used for the base and result libraries.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following sequence creates an empty source library named SOURCE_LIBRARY. The key character for the library is *.

```
/source_code_utility
sc/create_library
sc/quit
```

CREATE_MODIFICATION SCU Subcommand

Purpose Creates one or more modifications in the library modification list.

Format

CREATE_MODIFICATION or
CREATE_MODIFICATIONS or
CREM
MODIFICATION=list of name
FEATURE=name
AUTHOR=string
MODIFICATION_DESCRIPTION=list of string
STATUS=status variable

Parameters

MODIFICATION or **MODIFICATIONS** or **M**

List of one or more modification names (from 1 through 9 characters each). This parameter is required.

FEATURE or *F*

Optional name of the feature to which the modification belongs. If the feature name is not in the feature list, SCU adds the name to the list.

AUTHOR or *A*

Optional modification author.

MODIFICATION_DESCRIPTION or *MD*

Optional list of strings containing the modification description.

Remarks

- A modification created by a CREATE_MODIFICATION subcommand contains only the modification header; no lines belong to the modification. The modification is defined for specification on subsequent commands.
- Modifications are placed on the library in alphabetical order.
- If CREATE_MODIFICATIONS creates more than one header, the headers are identical except for their names.
- To display the modifications defined within the working library, enter a DISPLAY_MODIFICATION_LIST command. To determine within an expression whether a modification exists, use the SCU function $MODIFICATION.
- FEATURE name should not be ALL or NONE.
- For more information, see the NOS/VE Source Code Management manual.

Examples

The following subcommand creates a description for modification MOD_4 for feature SYNTAX_CHECK. The author of the modification is K. Riley. The text in the SCL variables LINE1 and LINE2 is the modification description.

```
sc/line1='This is a very long title for ..
sc../a modification to show that'
sc/line2='a list of strings may be used for ..
sc../the description.'
sc/create_modification m=mod_4 f=syntax_check ..
sc../a='K. Riley' md=(line1,line2)
```

$BASE
SCU Function

Purpose Returns the base library file.

Format **$BASE**

Parameters None.

Remarks For more information, see the NOS/VE Source Code Management manual.

Examples The following command displays the current value of the base file.

```
/source_code_utility
sc/use_library b=$user.fortran_lib
sc/display_value v=$base
:NVE.PAT.FORTRAN_LIB
```


$DECK
SCU Function

Purpose Returns a boolean value indicating whether the specified deck is in the working library.

Format **$DECK**
(DECK: name)

Parameters **DECK**

Name of the deck to be found. This parameter is required.

Remarks For more information, see the NOS/VE Source Code Management manual.

Examples The following command assigns a boolean value to the SCL variable DECK_EXISTS, depending on whether DECK1 is in the working library.

```
sc/deck_exists=$deck(deck1)
```

$DECK_ATTRIBUTES
SCU Function

Purpose Returns the content of an SCU deck header. The value is returned as a record.

Format

$DECK_ATTRIBUTES
(DECK: name)

Parameters *DECK*

Name of the deck for which the header content is returned. If DECK is omitted, the last used deck name is assumed.

Remarks

- The attributes have the following types:

```
ACTIVE_LINE_COUNT: integer
AUTHOR: string
CHARACTER: string
CREATION_DATE_TIME: date_time
DECK_DESCRIPTION: list of string
EXPAND: boolean
GROUP: list of name
INACTIVE_LINE_COUNT: integer
LINE_IDENTIFIER: name
MODIFICATION: list of name
MODIFICATION_DATE_TIME: date_time
NAME: name
ORIGINAL_INTERLOCK: record
   USER_NAME: name
   DATE_TIME: date_time
PROCESSOR: string
SUB_INTERLOCK: record
   USER_NAME: name
   DATE_TIME: date_time
TAB_COLUMNS: list of integer
WIDTH: integer
```

- To use the contents of the header returned, it is best to create a variable implicitly, for example, DA1=$DECK_ATTRIBUTES(DECK1).

- If you use the $DECK_ATTRIBUTES function to assign attributes to a variable in an iterative process, you must delete and re-create the variable for each iteration. The existing variable cannot be re-assigned the attributes of a different deck.
- If the deck has not been modified, the creation values are returned in the MODIFICATION_DATE_TIME field.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following command displays the modification headers for those modifications that apply to deck FTN_DECK.

```
sc/display_modifications ..
sc../m=$deck_attributes(ftn_deck).modification
```


$DECK_NAME_LIST Selection Criteria Function

Purpose Returns a list of deck names currently selected to be written to the result file.

Format **$DECK_NAME_LIST**

Parameters None.

Remarks

- The order of names is the same as on the working library.
- When used in selection criteria subcommand processing, $DECK_NAME_LIST reflects the current deck list to be written to the compile, result, or source file being produced.
- The function $DECK_NAME_LIST returns a list of names for ease in processing with the FOR EACH/ FOREND command.
- For more information, see the NOS/VE Source Code Management manual.

Examples This example shows the data structure of $DECK_NAME_LIST as an empty list.

```
/scu
sc/use_library ..
sc../b=$system.cybil.osf$program_interface r=$null
sc/extract_deck d=none sc=command
scc/display_value v=$deck_name_list ..
scc../do=data_structure
display option: DATA_STRUCTURE

"EMPTY LIST"
```

This example displays names of decks common to CYBIL and terminal definition groups:

```
sc/extract_deck d=none sc=$command s=$local.source1
scc/include_group g=(cybil,tug$define_terminal) ..
scc../c=all
scc/display_value v=$deck_name_list ..
scc../do=data_structure
1: "NAME"  TUC$CURSOR_NUMBER_OF_DIGITS
2: "NAME"  TUT$APPLICATION_NAME
  :
34: "NAME"  TUT$SUBTABLE_POINTERS
```

This example performs operations on each deck that is currently selected.

```
FOR EACH deck in $DECK_NAME_LIST DO
:
FOREND
```

$DECK_NAME_LIST SCU Function

Purpose Returns a list of deck names on the working library.

Format $DECK_NAME_LIST

Parameters None.

Remarks

- The names are listed alphabetically.
- When used in selection criteria subcommand processing, $DECK_NAME_LIST returns the names of currently selected decks.
- The function $DECK_NAME_LIST returns a list of names for ease in processing with the FOR EACH/ FOREND command.
- For more information, see the NOS/VE Source Code Management manual.

Examples This example performs operations on each deck on the working library.

```
FOR EACH deck in $DECK_NAME_LIST DO
  :
FOREND
```


DELETE_DECK SCU Subcommand

Purpose Deletes one or more decks from the working library.

Format

DELETE_DECK or
DELETE_DECKS or
DELD
DECK=list of: name or **range of name**
STATUS=status variable

Parameters

DECK or **DECKS** or **D**

Decks to be deleted. This parameter is required.

Remarks

- You cannot delete a deck if the creation modification of the deck is in a state greater than your authority for the file.
- The DELETE_DECK subcommand removes the deck name from the deck list of the working library (as opposed to being deactivated like the EDIT_DECK DELETE_LINE subcommand).

- When you specify a range of decks, DELETE_DECK deletes each deck in the deck list, beginning with the first deck specified through the last deck specified. Before specifying a range of decks to be deleted, you should display the deck list with a DISPLAY_DECK_LIST subcommand to determine the decks included in the range.

- If a deck to be deleted has a conflicting subinterlock set, SCU sends a warning message, observing that another user extracted the deck using an EXTRACT_SOURCE_LIBRARY command. The deck is deleted. SCU then attempts to delete any remaining decks.

- For more information, see the NOS/VE Source Code Management manual.

Examples The following command deletes deck DECKA and decks DECKC through DECKF.

```
sc/delete_decks d=(decka,deckc..deckf)
```

DELETE_FEATURE SCU Subcommand

Purpose Deletes one or more features from a source library.

Format

DELETE_FEATURE or
DELETE_FEATURES or
DELF
FEATURE=list of: name or **range of name**
STATUS=status variable

Parameters

FEATURE or **FEATURES** or **F**

Feature to be deleted. This parameter is required.

Remarks

- If modifications are still associated with the feature, a warning message is issued and the feature is retained. Modifications can be disassociated with a feature using the CHANGE_MODIFICATION subcommand or deleted using the DELETE_MODIFICATION subcommand.

- For more information, see the NOS/VE Source Code Management manual.

Examples In the following example, the CHANGE_MODIFICATION subcommand changes modifications that were associated with the feature TRIAL to the feature FILE_LIMITS. The DELETE_FEATURE subcommand then deletes the now unused feature name TRIAL from the library feature list.

```
sc/change_modification ..
sc../m=$feature_member_names(trial) f=file_limits
sc/delete_feature f=trial
```

DELETE_GROUP SCU Subcommand

Purpose Deletes one or more groups from a source library.

Format

DELETE_GROUP or
DELETE_GROUPS or
DELG
GROUP=**list of: name** or **range of name**
STATUS=status variable

Parameters

GROUP or **GROUPS** or **G**

Group to be deleted. This parameter is required.

Remarks

- If decks still belong to the group, a warning message is issued and the group is retained. Decks can be moved from a group using the CHANGE_DECK subcommand or deleted using the DELETE_DECK subcommand.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following example assumes a group name for decks on a source library was misspelled. The CHANGE_DECK subcommand substitutes the correct group name for the incorrect name and the DELETE_GROUP subcommand deletes the incorrect group name from the library group list.

```
sc/change_deck d=$group_member_names(dybil) ..
sc../dg=dybil g=cybil
sc/delete_group g=dybil
```

DELETE_MODIFICATION SCU Subcommand

Purpose Deletes one or more modifications. Deleting a modification reverses all text changes that were introduced by the modification. All insertions are deleted, all replacements are removed, and all deletions are reactivated.

Format

DELETE_MODIFICATION or
DELETE_MODIFICATIONS or
DELM
MODIFICATION=list of: name or **range of name**
DECK=keyword or *list of: name* or *range of name*
STATUS=status variable

Parameters

MODIFICATION or **MODIFICATIONS** or **M**

Modifications to be deleted. This parameter is required.

DECK or *DECKS* or *D*

Either one or more deck names or the keyword ALL. ALL specifies all decks in the working library. If DECK is specified, SCU deletes only the modification changes within the specified decks. If DECK is omitted, ALL is used.

Remarks

- You cannot delete the creation modification of a deck directly: you must first delete each deck for which the modification is the creation modification. You can then delete the modification from the modification list.
- You cannot delete a modification whose state is greater than your authority for the file.
- If a deck affected by a deleted modification has its subinterlock set, SCU sends a warning message, stating that a user has extracted the deck with an EXTRACT_SOURCE_LIBRARY command. The modification is deleted. SCU then attempts deletion of modification changes on any remaining decks in the deck list.
- You can use this subcommand to create a new library without the modification. To temporarily reverse modification changes when expanding text, use the selection criteria subcommand EXCLUDE_MODIFICATION.

- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand deletes modification MOD5.

```
sc/delete_modification m=mod5
```

DISPLAY_DECK SCU Subcommand

Purpose Displays one or more deck headers.

Format

DISPLAY_DECK or
DISPLAY_DECKS or
DISD
DECK=keyword or *list of: name* or *range of name*
OUTPUT=file
DISPLAY_OPTIONS=keyword
TEXT=keyword
STATUS=status variable

Parameters *DECK* or *DECKS* or *D*

Decks whose headers are to be displayed. You can specify a list of one or more deck names, a list of one or more deck ranges, or the keyword ALL. ALL specifies all decks in the working library. If DECK is omitted, the last used deck is displayed.

OUTPUT or *O*

File on which the display is written. You can specify a file position as part of the file name. If OUTPUT is omitted, file $OUTPUT is used.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Specifies the information listed. Options are:

BRIEF (B)

Lists only deck header information.

FULL (F)

Lists deck header information, modifications to which deck lines belong, and the groups to which the deck belongs. ALL is an alias for FULL.

If DISPLAY_OPTIONS is omitted, BRIEF is used.

TEXT or *T*

Specifies deck text to be displayed. Options are:

INACTIVE (I)

Active and inactive lines.

ACTIVE (A)

Active lines only.

NONE

Deck text is not displayed.

If TEXT is omitted, NONE is used.

Remarks

- You can display deck text with the DISPLAY_DECK subcommand. You can display either the active lines or both the active and inactive lines. Inactive lines are lines that have been deleted; only active lines appear in expanded deck text.
- The DISPLAY_DECK subcommand is valid within an editing session started by an EDIT_DECK subcommand. It is also valid within a selection criteria file if prefixed with the slant character (/DISPLAY_DECK).
- For more information, see the NOS/VE Source Code Management manual.

Examples

The following subcommand displays the deck header of deck DECK1. The subcommand specifies full information level (DO=F) so the modifications in the deck and the groups to which the deck belong are also displayed. The subcommand also specifies a listing of both the inactive and active lines in the deck (T=I).

```
sc/display_deck d=deck1 do=f t=i

  Deck Information

DECK: DECK1
EXPAND: FALSE
AUTHOR: M.J.Perreten
PROCESSOR: Fortran
ORIGINAL_INTERLOCK:
SUB_INTERLOCK:
WIDTH: 80
LINE IDENTIFIER: none
TAB ACTIVE: TRUE
CHARACTER: #
TAB_COLUMNS:   5,   7,   9,  11,  13  15,  17
CREATION_DATE - TIME: 12/02/81 - 10:41:51
MODIFICATION_DATE -TIME: 03/24/82 - 13:37:19
DECK_DESCRIPTION: First example deck
COUNTS
 MODS:2     GROUPS:1   LINES ACTIVE:6   INACTIVE:1

 MODS AND SEQUENCE NUMBERS
ORIGINAL  6
FIRST_MOD 1

 GROUP LIST
LOOPS
  Active(A)/Inactive(I) text lines for deck DECK1

A ORIGINAL       1
A ORIGINAL       2        DO 10 I=1,100
I ORIGINAL       3        10 I= I+1
                        I FIRST_MOD
A FIRST_MOD      1     10 I= I+1
A ORIGINAL       4
A ORIGINAL       5 *COPYC COMMON1
A ORIGINAL       6
```

Each line of the text listing contains a letter indicating whether the line is active or inactive (A or I), the line identifier, and the line text. If the line is inactive, the succeeding line names the modification that deactivated the line.

DISPLAY_DECK_LIST SCU Subcommand

Purpose Lists the decks found in the working library in alphabetical order by deck name.

Format

DISPLAY_DECK_LIST or
DISDL
ALTERNATE_BASE=list of file
OUTPUT=file
DISPLAY_OPTIONS=keyword
STATUS=status variable

Parameters

ALTERNATE_BASE or *ALTERNATE_BASES* or *AB*

Optional list of one or more source libraries whose deck lists are combined with the working library deck list. If ALTERNATE_BASE is omitted, the decks on the current working library will be displayed.

OUTPUT or *O*

File on which the display is written. You can specify a file position as part of the file name. If OUTPUT is omitted, file $OUTPUT is used.

DISPLAY_OPTIONS or *DO*

Specifies the information listed. Currently, both of the following keywords produce the same listing.

BRIEF or B
FULL or F

If DISPLAY_OPTIONS is omitted, BRIEF is used. ALL is an alias for FULL.

Remarks

- If you specify one or more alternate base libraries, DISPLAY_DECK_LIST combines their deck lists with the working library deck list for the duration of the subcommand. You can use this option to display the deck list that would be used if you specified the alternate base libraries on an EXPAND_DECKS or EXTRACT_DECKS subcommand.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand displays a combined deck list of the decks on source library MY_LIB and the working library.

```
sc/display_deck_list ab=my_lib
FORTRAN_TEXT      FORTRAN_TEXT_II
MY_TEXT
```

The listing does not indicate which source library contains the deck.

DISPLAY_DECK_REFERENCES SCU Subcommand

Purpose Displays a cross-reference listing for one or more decks. A reference to a deck is a COPY or COPYC directive that names the deck.

Format **DISPLAY_DECK_REFERENCES** or **DISDR**

DECK=keyword or *list of: name* or *range of name*
EXTERNAL_DECK=keyword or *list of name*
OUTPUT=file
DECK_RESIDENCE=keyword
REFERENCE_DIRECTION=keyword
REFERENCE_TYPE=keyword
STATUS=status variable

Parameters *DECK* or *DECKS* or *D*

Decks to be cross-referenced. You can specify a list of names, a list of ranges, or the keyword ALL or NONE. ALL specifies all decks in the working library. If DECK is omitted, the name of the last deck is used. If you specify NONE, you prevent the last deck from being cross-referenced.

EXTERNAL_DECK or *EXTERNAL_DECKS* or *ED*

Decks to be cross-referenced that are not on the working library. You can specify a list of names or the keyword ALL. ALL specifies all decks not in the working library that are referenced by decks in the working library. If EXTERNAL_DECK is omitted, you must specify the DECK parameter.

OUTPUT or *O*

File on which the cross-reference is written. You can specify a file position as part of the file name. If OUTPUT is omitted, file $OUTPUT is used.

DECK_RESIDENCE or *DR*

Specifies the references to list. Options are:

EXTERNAL

List only references to decks not in the working library.

INTERNAL

List only references to decks in the working library.

ALL

List references to decks both in the working library and not in the working library.

If DECK_RESIDENCE is omitted, ALL is used.

REFERENCE_DIRECTION or *RD*

Specifies the direction the references are traced. Options are:

TO

References to the decks.

FROM

References from the decks.

ALL

References to and from the decks.

If REFERENCE_DIRECTION is omitted, TO is used.

REFERENCE_TYPE or *RT*

Specifies the reference type to be listed. Options are:

DIRECT

Lists only direct references.

INDIRECT

Lists only indirect references.

ALL

Lists both direct and indirect references.

If REFERENCE_TYPE is omitted, ALL is used.

Remarks

- The REFERENCE_TYPE parameter indicates whether DISPLAY_DECK_REFERENCES lists direct references or indirect references or both.
- Direct references involve only two decks; indirect references involve three or more decks. For example, if DECKA contains a COPY directive that copies DECKB, DECKA directly references DECKB. If DECKB contains a COPY directive that copies DECKC, DECKA indirectly references DECKC.
- The DECK_RESIDENCE parameter indicates whether this subcommand lists references to decks within the working library, decks not in the working library, or both.
- This subcommand is valid within an editing session started by an EDIT_DECK subcommand. It is also valid within a selection criteria file if prefixed with the slant character (/DISPLAY_DECK_REFERENCES).
- For more information, see the NOS/VE Source Code Management manual.

Examples

The following subcommand produces a cross-reference for deck SUB1 on the working library. It traces direct and indirect references both to and from the deck, including references to decks not resident on the working library.

```
sc/display_deck_references d=sub1 rd=all
  References FROM deck
  (e = external deck, i = indirect reference)

SUB1                                  references
 e  SUB2

  References TO internal deck
  (i = indirect reference)

SUB1                                  is referenced by
   PROGRAM1
```

DISPLAY_FEATURE SCU Subcommand

Purpose Displays the modifications belonging to a feature.

Format

DISPLAY_FEATURE or
DISF
FEATURE=name
OUTPUT=file
DISPLAY_OPTIONS=keyword
STATUS=status variable

Parameters

FEATURE or **F**

Feature name. This parameter is required.

OUTPUT or *O*

File on which the display is written. You can specify a file position as part of the file name. If OUTPUT is omitted, file $OUTPUT is used.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Specifies the information displayed. Options are:

BRIEF (B)

Lists only the modification names.

FULL (F)

Lists the modification names and the modification descriptions. ALL is an alias for FULL.

If DISPLAY_OPTIONS is omitted, BRIEF is used.

Remarks

- You can change the feature to which a modification belongs with the CHANGE_MODIFICATION subcommand.
- The DISPLAY_FEATURE subcommand is valid within an editing session started by an EDIT_DECK subcommand. It is also valid within a selection criteria file if prefixed with the slant character (/DISPLAY_FEATURE).
- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand displays the names and modification descriptions for all modifications belonging to the feature NEW_PROMPTS.

```
sc/display_feature f=new_prompts do=f
Descriptions of modifications associated with the feature
    NEW_PROMPTS

MODIFICATION: PROMPT_1
STATE: 0
FEATURE: NEW_PROMPTS
AUTHOR: Jane Doe
CREATION_DATE - TIME:      10/31/83 - 08.24.54
MODIFICATION_DATE - TIME: 10/31/83 - 08.24.54
MODIFICATION_DESCRIPTION: This adds a prompt for parameter
                          NEW_DECK.
MODIFICATION: PROMPT_2
STATE: 0
FEATURE: NEW_PROMPTS
AUTHOR: Jane Doe
CREATION_DATE - TIME:      11/05/83 - 13.29.04
MODIFICATION_DATE - TIME: 11/06/83 - 09.46.15
MODIFICATION_DESCRIPTION: This adds a prompt for parameter
                          OLD_DECK.

 Number of modifications associated with this feature: 2
```


DISPLAY_FEATURE _LIST SCU Subcommand

Purpose Lists the features in the source library.

Format

DISPLAY_FEATURE _LIST or
DISFL
OUTPUT=file
DISPLAY_OPTIONS=keyword
STATUS=status variable

Parameters *OUTPUT* or *O*

File on which the display is written. You can specify a file position as part of the file name. If OUTPUT is omitted, file $OUTPUT is used.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Specifies the information listed. Options are:

BRIEF (B)

Lists only the feature names.

FULL (F)

Lists the feature names and the names of the modifications that belong to each feature. ALL is an alias for FULL.

If DISPLAY_OPTIONS is omitted, BRIEF is used.

Remarks

- Features are listed alphabetically.
- To add a feature, create a modification that belongs to the feature. If the feature list contains an unused feature name, you can enter an EXTRACT_SOURCE_LIBRARY command to remove all unused feature names from the result library. The feature list of the new library includes only those features with which modifications in the new library are associated and which have not been explicitly excluded by selection criteria commands.
- The DISPLAY_FEATURE_LIST subcommand is valid within an editing session started by an EDIT_DECK subcommand. It is also valid within a selection criteria file if prefixed with the slant character (/DISPLAY_FEATURE_LIST).
- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand lists the features in the working library.

```
sc/display_feature_list
NEW_PROMPTS                    NEW_RESPONSE
```

DISPLAY_GROUP SCU Subcommand

Purpose Lists the decks belonging to a group.

Format

DISPLAY_GROUP or
DISG
GROUP=name
ALTERNATE_BASE=list of file
OUTPUT=file
DISPLAY_OPTIONS=keyword
STATUS=status variable

Parameters

GROUP or G

Group name. This parameter is required.

ALTERNATE_BASE or *ALTERNATE_BASES* or *AB*

Optional list of one or more additional source libraries from which decks are listed if they belong to the group.

OUTPUT or *O*

File on which output is written. You can specify a file position as part of the file name. If OUTPUT is omitted, file $OUTPUT is used.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Specifies the information listed. Options are:

BRIEF (B)

Lists only the deck names.

FULL (F)

Lists the deck names and the information in each deck header. ALL is an alias for FULL.

If DISPLAY_OPTIONS is omitted, BRIEF is used.

Remarks

- If you specify one or more alternate base libraries, DISPLAY_GROUP combines their group and deck lists with the working library group and deck lists for the duration of the subcommand.
- You can change the group to which a deck belongs with the CHANGE_DECK subcommand.
- The DISPLAY_GROUP subcommand is valid within an editing session started by an EDIT_DECK subcommand.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand lists the decks in the group SECTION1.

```
sc/display_group g=section1
Decks associated with group SECTION1

FORTRAN_TEXT                FORTRAN_TEXT_II
FORTRAN_TEXT_III
```

DISPLAY_GROUP_LIST SCU Subcommand

Purpose Lists the groups in the library.

Format

DISPLAY_GROUP_LIST or
DISGL
ALTERNATE_BASE=list of file
OUTPUT=file
DISPLAY_OPTIONS=keyword
STATUS=status variable

Parameters

ALTERNATE_BASE or *ALTERNATE_BASES* or *AB*

Optional list of one or more libraries whose groups are listed with those of the base library.

OUTPUT or *O*

File on which the display is written. You can specify a file position as part of the file name. If OUTPUT is omitted, file $OUTPUT is used.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Specifies the information listed. Options are:

BRIEF (B)

Lists only the group names.

FULL (F)

Lists the group names and the decks in each group. ALL is an alias for FULL.

If DISPLAY_OPTIONS is omitted, BRIEF is used.

Remarks

- Groups are listed alphabetically.
- If you specify one or more alternate base libraries, DISPLAY_GROUP_LIST combines their group and deck lists with the working library group and deck lists for the duration of the subcommand.
- To add a group, create a deck that belongs to the group by including the group name on the CREATE_DECK subcommand. If the group list contains an unused group name, you can enter an EXTRACT_SOURCE_LIBRARY command to remove all unused group names from the result library. The group list of the new library includes only those groups to which decks in the new library belong and which have not been explicitly excluded by selection criteria commands.
- The DISPLAY_GROUP_LIST subcommand is valid within an editing session started by an EDIT_DECK subcommand.
- For more information, see the NOS/VE Source Code Management manual.

Examples

The following subcommand lists the groups on the working library and on library MY_LIB.

```
sc/display_group_list ab=my_lib
SECTION1                              SECTION2
SECTION3
```

DISPLAY_LIBRARY SCU Subcommand

Purpose

Displays the library header of the working library.

Format

DISPLAY_LIBRARY or
DISL
OUTPUT=file
DISPLAY_OPTIONS=keyword
STATUS=status variable

Parameters *OUTPUT* or *O*

File on which the display is written. You can specify a file position as part of the file name. If OUTPUT is omitted, file $OUTPUT is used.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Specifies the information listed. Options are:

BRIEF (B)

Lists only library header information.

FULL (F)

Lists library header information and the names of the decks, groups, modifications, and features in the working library. ALL is an alias for FULL.

If DISPLAY_OPTIONS is omitted, BRIEF is used.

Remarks

- Besides the library header fields, DISPLAY_LIBRARY can also display the deck list, group list, modification list, and feature list of the working library.
- You can change the content of fields in the working library header with a CHANGE_LIBRARY subcommand. To reference a field in the library header, use the SCU function $LIBRARY_ATTRIBUTES.
- The DISPLAY_LIBRARY subcommand is valid within an editing session started by an EDIT_DECK subcommand. It is also valid within a selection criteria file if prefixed with the slant character (/DISPLAY_LIBRARY).
- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand displays the contents of the working library header.

```
sc/display_library
LIBRARY: SOURCE_CODE_UTILITY
VERSION: BUILD_12609
SCU_VERSION: 86133
LIBRARY_FORMAT_VERSION: V1.1
CHANGE_COUNTER: 394
LIBRARY_DESCRIPTION:  This library contains the source for
SOURCE_CODE_UTILITY (SCU) and associated SCL procedures.
CREATION_DATE - TIME:      07/31/81 - 13:15:43
MODIFICATION_DATE - TIME: 06/10/86 - 22:33:17
KEY: *
LAST_USED_DECK: SCP$GET_DEFAULT_RESOURCES
LAST_USED_MODIFICATION: SCB6134
COUNTS
 DECKS: 1237     MODS: 719     GROUPS: 41   FEATURES: 246
```

DISPLAY_MODIFICATION SCU Subcommand

Purpose Displays one or more modification headers.

Format

DISPLAY_MODIFICATION or
DISPLAY_MODIFICATIONS or
DISM

MODIFICATION=keyword or *list of: name* or *range of name*
DECK=keyword or *list of: name* or *range of name*
OUTPUT=file
DISPLAY_OPTIONS=keyword
STATUS=status variable

Parameters

MODIFICATION or *MODIFICATIONS* or *M*

Modifications to be displayed. You can specify a list of one or more names, a list of one or more ranges, or the keyword ALL. ALL specifies all modification descriptions in the working library. If MODIFICATION is omitted, the last used modification is displayed.

DECK or *D*

Indicates whether the displayed information should apply to only the specified deck or to all decks. ALL specifies all decks in the working library. If DECK is omitted, ALL is used.

OUTPUT or *O*

File on which the display is written. You can specify a file position as part of the file name. If OUTPUT is omitted, file $OUTPUT is used.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Specifies the information displayed. Options are:

BRIEF (B)

Displays the modification header only.

FULL (F)

Displays the modification header and the sequence of editing commands and inserted text that would produce the modification changes. ALL is an alias for FULL.

If DISPLAY_OPTIONS is omitted, BRIEF is used.

Remarks

- The DISPLAY_MODIFICATION subcommand is valid within an editor session started by an EDIT_DECK subcommand. It is also valid within a selection criteria file if prefixed with the slant character (/DISPLAY_MODIFICATION).
- For more information, see the NOS/VE Source Code Management manual.

Examples

The following subcommand displays the modification MOD_4 description and changes.

```
sc/display_modification m=mod_4 do=f
MODIFICATION: MOD_4
STATE: 0
FEATURE:
AUTHOR: Sam Spade
CREATION_DATE - TIME:     02/23/83 - 13:09:26
MODIFICATION_DATE - TIME: 02/24/83 - 08:14:01
MODIFICATION_DESCRIPTION: Fourth example modification
Text lines altered by modification MOD_4
SELECT_DECK X
INSERT_LINES P=BEFORE IL=FIRST UNTIL='///END\\\'
      do 10 i=1,10
   10 i = i+1
///END\\\
INSERT_LINES P=AFTER IL=MOD_3.2 UNTIL='///END\\\'
```

```
  100 i = i+100
///END\\\
SELECT_DECK Y
INSERT_LINES P=BEFORE IL=FIRST UNTIL='///END\\\'
*copyc z
///END\\\
```

DISPLAY_MODIFICATION_LIST SCU Subcommand

Purpose Lists all modifications in the working library.

Format **DISPLAY_MODIFICATION_LIST** or **DISML**

OUTPUT=file
DISPLAY_OPTIONS=keyword
STATUS=status variable

Parameters *OUTPUT* or *O*

File on which the display is written. You can specify a file position as part of the file name. If OUTPUT is omitted, file $OUTPUT is used.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Specifies the information listed.

ALPHABETIC (A)

Modifications are in alphabetical order.

CHRONOLOGICAL (C)

Modifications are ordered by date and time with the oldest modification first.

If DISPLAY_OPTIONS is omitted, ALPHABETIC is used.

Remarks

- To add a modification to the list, enter a CREATE_MODIFICATION subcommand. To remove a modification from the list, enter a DELETE_MODIFICATION subcommand.
- The DISPLAY_MODIFICATION_LIST subcommand is valid within an editing session started by an EDIT_DECK subcommand. It is also valid within a selection criteria file if prefixed with the slant character (/DISPLAY_MODIFICATION_LIST).

- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand lists all modifications in the working library.

```
sc/display_modification_list
MOD_1      MOD_2      MOD_3      MOD_4
```

EDIT_DECK
SCU Subcommand

Purpose Begins an editing session within an SCU session.

Format

EDIT_DECK or
EDID or
EDIT_LIBRARY or
EDIL
DECK=keyword or *name*
MODIFICATION=name
INPUT=file
OUTPUT=file
PROLOG=file
DISPLAY_UNPRINTABLE_CHARACTERS=boolean
STATUS=status variable

Parameters *DECK* or *D*

Deck to be edited first.

NOTE

If the deck does not exist, it is created. If you have never entered a deck name on a DECK parameter, this parameter is required.

If DECK is omitted, the editing session begins with the last deck used.

To begin the editing session without selecting a deck, specify NONE on the DECK parameter.

MODIFICATION or *M*

Modification to which changes made during the editing session belong. For you to edit a deck using an existing modification, the modification must be in its initial state, state 0. If the modification does not already exist, it is created.

If MODIFICATION is omitted, the last modification is used. If you have never created a modification, this parameter is required.

INPUT or *I*

File from which commands are read. If INPUT is omitted, $COMMAND is used.

OUTPUT or *O*

File to which the display is written. If OUTPUT is omitted, file $OUTPUT is used. ($OUTPUT is usually connected to the terminal.)

PROLOG or *P*

File the system executes when you start an editing session. If PROLOG is omitted, file $USER.SCU_EDITOR_PROLOG is used. You can establish a different default prolog file by using the CREATE_DEFAULT_VARIABLE command to set the variable ESD$EDIF_PROLOG to the file you want to be your default prolog. For more information on the CREATE_DEFAULT_VARIABLE command, refer to the NOS/VE System Usage manual.

DISPLAY_UNPRINTABLE_CHARACTERS or *DUC*

Specifies whether unprintable ASCII characters in the range 0 to 31 and 127 are replaced by mnemonics in the file. Options are:

TRUE

Unprintable characters are replaced by mnemonics, preceded by a less than symbol and followed by a greater than symbol, according to the ASCII character set.

FALSE

Unprintable characters are replaced by a single space and a warning message is issued if they are encountered. If the file is written when you exit the editing session, the mapping to spaces is written to the file.

If TRUE is specified, the mnemonics are replaced by the ASCII characters when the file is replaced. If DISPLAY_UNPRINTABLE_ CHARACTERS is omitted, FALSE is used.

Remarks

- You can specify the deck to be edited with the DECK parameter. If you specify NONE on the DECK parameter, you must enter a deck selection subcommand before entering subcommands to change text.
- Commands that follow the EDIT_DECK subcommand on the same line are processed as editor subcommands.
- This subcommand adds an entry containing the EDIT_FILE utility subcommands to the NOS/VE subcommand list; the name of the entry is SCU_EDIT.
- If the interaction style you selected is SCREEN, the session occurs in full screen mode. The command CHANGE_INTERACTION_STYLE selects interaction modes.
- All editing subcommands and the deck selection subcommands that are available within the EDIT_FILE utility are described in the NOS/VE File Editor manual.
- The EDIT_FILE utility uses the tab columns specified in the deck header.
- Once you have started an editing session with an EDIT_DECK subcommand, you can then use an EDIT_FILE subcommand to edit a file.
- To discard decks that were created unintentionally, enter:

```
end_deck write_deck=false
```

- Once you have entered the SCU EDIT_DECK subcommand, you can enter the EDIT_DECK subcommand to edit other decks. This subcommand has only a DECK parameter.
- To change modifications, you must stop editing and enter the EDIT_DECK SCU subcommand specifying a different modification.
- The mnemonics that appear when DISPLAY_UNPRINTABLE_CHARACTERS=TRUE will be enclosed in less than and greater than symbols. For example, the mnemonic for the ASCII character 0 is NUL. This mnemonic appears on the terminal screen as follows: <NUL>
- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand begins an editing session in line mode. All text changes belong to the new modification MOD_1.

```
sc/edit_deck m=mod_1
sce/
```

The following is the header written on the output file if the EDIT_DECK subcommand is entered in batch mode.

```
EDITOR                                          08:39:10    PAGE 1
1986-07-09    NOS/VE SOURCE CODE UTILITY V1.1 86163
BASE=:nve.pat.my_library.3
Begin editing deck DECK10
```

Including the LOCATE_TEXT (L) editor subcommand after the following EDIT_DECK subcommand causes the editor to start with the cursor positioned at the first occurrence of an XDCL procedure in deck PROC1.

```
sc/edit_deck d=proc1 m=x; l 'PROCEDURE [XDCL]'
```

END_LIBRARY SCU Subcommand

Purpose Ends the interaction with the current working library. Another library can then be specified as the working library.

Format

END_LIBRARY or
ENDL
WRITE_LIBRARY=boolean
STATUS=status variable

Parameters

WRITE_LIBRARY or *WL*

Specifies whether the working library should be written to the result file. The result file is specified in the CREATE_LIBRARY or USE_LIBRARY subcommand. If no result file was specified and you indicate that the working library should be written to the result file, then the library is written to file SOURCE_LIBRARY. If WRITE_LIBRARY is omitted, TRUE is used.

Remarks

- After entering the END_LIBRARY subcommand, you can work on another library by specifying either the USE_LIBRARY or CREATE_LIBRARY subcommand.
- For more information, see the NOS/VE Source Code Management manual.

Examples

The following example ends the association with the current working library. The library is written if changes have been detected by the $LIBRARY_MODIFIED function. Another library is then accessed by the USE_LIBRARY subcommand.

```
sc/end_library wl=$library_modified
sc/use_library b=my_library r=new_library
```

$ERRORS_FILE
SCU Function

Purpose

Returns the file to which intermediate diagnostic messages are written.

Format

$ERRORS_FILE

Parameters

None.

Remarks

For more information, see the NOS/VE Source Code Management manual.

Examples The following command displays the current value of the file to which intermediate diagnostic messages are written.

```
/source_code_utility
sc/set_list_options e=$user.my_error_file
sc/display_value v=$errors_file
:NVE.PAT.MY_ERROR_FILE
```

EXCLUDE_DECK
Selection Criteria Subcommand

Purpose Explicitly excludes one or more decks.

Format

EXCLUDE_DECK or
EXCLUDE_DECKS or
EXCD
DECK=list of: name or **range of name**
STATUS=status variable

Parameters

DECK or **DECKS** or **D**

Decks to be excluded. This parameter is required.

Remarks For more information, see the NOS/VE Source Code Management manual.

Examples The following sequence extracts modules from base library $USER.MY_LIBRARY using selection criteria commands. The extracted modules are then written to $USER.PART_OF_MY_LIBRARY.

```
/extract_source_library b=$user.my_library ..
../r=$user.part_of_my_library i=none sc=command
scc/include_group g=group1
scc/exclude_deck d=unwanted
scc/quit
```

The command sequence extracts all decks belonging to group GROUP1 except deck UNWANTED. When selection criteria entry has ended, the result is written on $USER.PART_OF_MY_LIBRARY.

EXCLUDE_FEATURE
Selection Criteria Subcommand

Purpose Explicitly excludes modifications belonging to one or more features.

Format

EXCLUDE_FEATURE or
EXCLUDE_FEATURES or
EXCF
FEATURE=**list of name**
STATE=*integer*
STATUS=*status variable*

Parameters

FEATURE or **FEATURES** or **F**

Features to be excluded. This parameter is required.

STATE or *S*

Maximum state (from 0 through 4) of modifications excluded. All modifications whose state is less than or equal to this value are excluded. If STATE is omitted, all modifications belonging to the feature are excluded.

Remarks

- This command is not valid for an EXTRACT_SOURCE_LIBRARY subcommand that sets an interlock.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following sequence extracts new source library $USER.MY_RESULT from the library on file $USER.MY_LIBRARY.

```
/extract_source_library decks=all ..
../base=$user.my_library ..
../result=$user.my_result ..
../interlock=none selection_criteria=command
scc/exclude_feature new_prompts
scc/quit
/
```

The sequence extracts all decks from the source library. However, it omits all lines of text belonging to modifications associated with the feature NEW_PROMPTS. It omits the feature NEW_PROMPTS

from the feature list of the new library and the modifications associated with NEW_PROMPTS from the modification list.

EXCLUDE_GROUP
Selection Criteria Subcommand

Purpose Explicitly excludes the decks belonging to one or more groups.

Format

EXCLUDE_GROUP or
EXCLUDE_GROUPS or
EXCG
GROUP=list of name
COMBINATION=keyword
STATUS=status variable

Parameters

GROUP or **GROUPS** or **G**

Groups to be excluded. This parameter is required.

COMBINATION or *C*

Indicates whether the decks excluded must belong to one or all specified groups. Options are:

ANY

Excluded decks must belong to at least one of the specified groups.

ALL

Excluded decks must belong to all the specified groups.

If COMBINATION is omitted, ANY is used.

Remarks For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand sequence expands all decks on the working library except those belonging to group SECTION_1.

```
sc/expand_deck decks=all selection_criteria=command
scc/exclude_group group=section_1
scc/quit
```

EXCLUDE_LIBRARY
Selection Criteria Subcommand

Purpose Excludes decks found on one or more alternate base libraries. Although the command prevents you from selecting decks from specified libraries, COPY and COPYC directives processed by an EXPAND_DECK subcommand can still copy decks from the specified libraries.

Format

EXCLUDE_LIBRARY or
EXCLUDE_LIBRARIES or
EXCL
ALTERNATE_BASE=list of file
STATUS=status variable

Parameters

ALTERNATE_BASE or **ALTERNATE_BASES** or **AB**

Source library files whose decks are excluded. The files must be a subset of the libraries specified on the ALTERNATE_BASE parameter of the subcommand. This parameter is required.

Remarks

- The EXCLUDE_LIBRARIES subcommand allows you to specify source libraries on the ALTERNATE_BASE parameter of the EXPAND_DECK subcommand that are to be used only for decks copied by COPY and COPYC directives. No other decks on the excluded library are expanded.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand sequence expands all decks on the working library. Decks are copied from the library on file COMMON_LIBRARY if referenced by COPY or COPYC directives in the text.

```
sc/expand_decks decks=all alternate_base=..
sc../common_library selection_criteria=command
scc/exclude_library alternate_base=common_library
scc/quit
```

EXCLUDE_MODIFICATION
Selection Criteria Subcommand

Purpose Explicitly excludes one or more modifications.

Format

EXCLUDE_MODIFICATION or
EXCLUDE_MODIFICATIONS or
EXCM
MODIFICATION=list of name
STATUS=status variable

Parameters

MODIFICATION or MODIFICATIONS or M

Modifications to be excluded. This parameter is required.

Remarks

- This subcommand is not valid for an EXTRACT_SOURCE_LIBRARY subcommand that sets an interlock.
- If several modifications of the same line exist, it is possible for an expanded deck to contain two versions of the same line if the modification deactivating the original line is excluded from the expanded deck.

  For example, assume Line 1 Version 1 is introduced by modification A. Modification B deactivates and replaces that line with Line 1 Version 2. Then modification C deactivates and replaces Line 1 Version 2 with Line 1 Version 3. If the deck is expanded with modification B excluded, both the Line 1 Version 1 and Line 1 Version 3 will appear in the compile file because Line 1 Version 1 is no longer activated.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand sequence expands all text in decks DECK1 through DECK3 except those lines belonging to modifications MOD2 and MOD4.

```
sc/expand_decks decks=(deck1..deck3) ..
sc../selection_criteria=command
scc/exclude_modification (mod2,mod4)
scc/quit
```

EXCLUDE_STATE
Selection Criteria Subcommand

Purpose Explicitly excludes all modifications whose state is not greater than that specified.

Format

EXCLUDE_STATE or
EXCS
STATE=integer
STATUS=status variable

Parameters

STATE or **S**

Maximum state (from 0 through 3) of the modifications excluded. This parameter is required.

Remarks

- This command is not valid for an EXTRACT_SOURCE_LIBRARY subcommand that sets an interlock.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand sequence extracts all text in deck DECK1 except those lines belonging to modifications with a 0 (zero) or 1 state.

```
sc/extract_decks deck=deck1 ..
sc../selection_criteria=command
scc/exclude_state 1
scc/quit
```

EXPAND_DECK
SCU Subcommand

Purpose Expands one or more decks. When the SCU expands a deck, it processes directives embedded in the source text and copies the expanded text to a separate compile file.

Format

EXPAND_DECK or
EXPAND_DECKS or
EXPD
DECK=keyword or *list of: name* or *range of name*
COMPILE=file
DEBUG_AIDS=keyword
OUTPUT_SOURCE_MAP=file
SELECTION_CRITERIA=file

WIDTH=integer
LINE_IDENTIFIER=keyword
ALTERNATE_BASE=list of file
LIST=file
EXPANSION_DEPTH=integer
DISPLAY_OPTIONS=keyword
ORDER=keyword
STATUS=status variable

Parameters

DECK or *DECKS* or *D*

Decks to be expanded. You can specify a list of one or more names, a list of one or more ranges, or the keyword ALL. ALL specifies all decks in the working library and in any alternate base libraries specified on the ALTERNATE_BASE parameter. If DECK is omitted, the last deck used is expanded. To prevent the last used deck from being expanded, specify NONE on the DECK parameter. In that case, SCU determines the decks expanded by the subcommands entered via the selection criteria file.

COMPILE or *C*

File on which the expanded text is written. You can specify a file position as part of the file name. If COMPILE is omitted, file COMPILE is used.

DEBUG_AIDS or *DA*

If this parameter is set to DT, screen debugging information is written to the file named by the OUTPUT_SOURCE_MAP parameter. If DEBUG_AIDS is set to NONE or is omitted, no debugging information is produced.

OUTPUT_SOURCE_MAP or *OSM*

Names a file to receive screen debugging information specified by the DEBUG_AIDS parameter. If the file is not named, the screen debugging information is written to a file named OUTPUT_SOURCE_MAP.

SELECTION_CRITERIA or *SC*

File from which selection criteria commands are read. You can specify a file position as part of the file name. To enter selection criteria commands interactively, specify

COMMAND. If SELECTION_CRITERIA is omitted, no selection criteria processing is performed and the DECK parameter specifies which decks will be expanded.

WIDTH or *W*

Length of the expanded lines excluding line identifiers. If WIDTH is omitted, SCU uses the default line width from the header of each deck.

LINE_IDENTIFIER or *LI*

Line identifier placement. Options are:

RIGHT (R)

Line identifiers are placed to the right of the text.

LEFT (L)

Line identifiers are placed to the left of the text.

NONE

No line identifiers are placed on output lines.

If LINE_IDENTIFIER is omitted, SCU uses the default line identifier placement from the header of each deck.

ALTERNATE_BASE or *ALTERNATE_BASES* or *AB*

Optional list of one or more additional libraries to be searched for decks.

LIST or *L*

Listing file. You can specify a file position as part of the file name. Within an SCU session, if LIST is omitted, the listing file is the file specified on the SET_LIST_OPTIONS subcommand. Otherwise, the default is file $LIST.

EXPANSION_DEPTH or *ED*

Number of levels of COPY and COPYC directives to process. COPY and COPYC directives beyond the maximum expansion depth are expanded as text. If EXPANSION_DEPTH is omitted, COPY and COPYC directives are processed whenever they are encountered.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Indicates whether the listing includes the library for each deck from which the deck was expanded. Options are:

BRIEF (B)

Does not list the decks or their library origins.

FULL (F)

Lists the library origin when more than one library is used. ALL is an alias for FULL.

If DISPLAY_OPTIONS is omitted, BRIEF is used.

ORDER or *O*

Indicates whether the decks are expanded in the order specified or in alphabetical order. Options are:

COMMAND (C)

Decks are expanded in the order specified on the DECK parameter and by selection criteria commands.

LIBRARY (L)

Decks are expanded in alphabetical order.

If ORDER is omitted, LIBRARY is used.

Remarks

- For each deck specified by the DECK parameter, the EXPAND_DECK subcommand checks the expand attribute to determine if it expands the deck. If the expand attribute is TRUE, it expands the deck. If the expand attribute is FALSE, it skips the deck and continues processing with the next specified deck.
- To expand a text file, use the EXPAND_FILE subcommand and the EXPAND_SOURCE_FILE command.
- In order for OUTPUT_SOURCE_MAP to correctly reflect the origin of the text of each deck, the deck must either be unmodified or have been written to a result library. If a deck is encountered whose only current source is on the working library and the result library is currently scheduled for an actual file, then the currently scheduled result library is logged in the

output source map as the origin and an error status is issued. A WRITE_LIBRARY subcommand must be entered to copy all decks from the working library to an actual file.

If $NULL was specified as the result library, an error status is issued and the attempt aborts. A WRITE_LIBRARY subcommand must be entered, naming the result library. Then the EXPAND_DECK subcommand can be reissued.

- You can specify the decks to be expanded by name on the DECK parameter or by selection criteria commands in the selection criteria file or both. SCU begins with the decks specified on the DECK parameter and then adds and removes decks as specified by selection criteria commands. It omits any decks whose expand attribute is FALSE.

- You can specify alternate base libraries with the ALTERNATE_BASE parameter. SCU begins searching for a deck in the working library. If the deck is not found, SCU searches the ALTERNATE_BASE libraries in the order that they appear in the specified list.

- The EXPANSION_DEPTH parameter can limit the levels of nested directives processed. If SCU reads a directive at a level beyond the maximum level processed, it expands the directive as text.

- The LINE_IDENTIFIER, WIDTH, and ORDER parameters affect how the expanded text is written on the compile file. The LINE_IDENTIFIER and WIDTH parameters can override the default values in the deck headers. The ORDER parameter allows you to specify the order that SCU writes the decks on the file. If LINE_IDENTIFIER is explicitly stated in the EXPAND_DECK command, then the file attribute STATEMENT_IDENTIFIER is set. If LINE_IDENTIFIER is not explicitly stated, the system assumes that the file contents of the decks are not homogeneous and does not set STATEMENT_IDENTIFIER.

- The line width can be specified by the WIDTH parameter. If the line width for a deck is 0 (zero), EXPAND_DECKS writes each line as it is stored in the deck (no trailing blanks or truncation); a blank line, therefore, is written as a zero-length V record. If the line width for a deck is nonzero, EXPAND_DECKS writes each line using that width. Lines shorter than the width are padded with trailing blanks; lines longer than the width are truncated.

- SCU issues a warning message for those decks that cannot be expanded.

- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand expands the text of deck FORTRAN_TEXT and writes the expanded text on file FORTRAN_INPUT.

```
sc/expand_deck d=fortran_text c=fortran_input ..
sc../do=full ab=ftnlib l=output
*=Deck was copied
 FORTRAN_TEXT
*FTN_IO                    :NVE.PAT.FTNLIB
*FTN_FORM                  :NVE.PAT.FTNLIB
```


EXPAND_FILE SCU Subcommand

Purpose Expands a text file. When the system expands a file, it processes the directives embedded in the source text and copies the expanded text to a separate compile file.

Format

EXPAND_FILE or
EXPF
FILE=file
COMPILE=file
DEBUG_AIDS=keyword
INPUT_SOURCE_MAP=file
OUTPUT_SOURCE_MAP=file
SELECTION_CRITERIA=file
WIDTH=integer
LINE_IDENTIFIER=keyword

ALTERNATE_BASE=list of file
LIST=file
EXPANSION_DEPTH=integer
DISPLAY_OPTIONS=keyword
STATUS=status variable

Parameters

FILE or F

File to be expanded. This parameter is required.

COMPILE or *C*

File on which the expanded text is written. You can specify a file position as part of the file name. If COMPILE is omitted, file COMPILE is used.

DEBUG_AIDS or *DA*

If this parameter is set to DT, screen debugging information is written to the file named by the OUTPUT_SOURCE_MAP parameter. If DEBUG_AIDS is set to NONE or is omitted, no debugging information is produced.

INPUT_SOURCE_MAP or *ISM*

Names a file from which screen debugging information is copied for the file specified by the FILE parameter. The content of the input source map is the output source map that was generated when the content of the FILE was produced. If INPUT_SOURCE_MAP is omitted, the screen debugging information describes lines read from FILE as having that origin.

OUTPUT_SOURCE_MAP or *OSM*

Names a file to receive screen debugging information specified by the DEBUG_AIDS parameter. If OUTPUT_SOURCE_MAP is omitted, the screen debugging information is written to a file named OUTPUT_SOURCE_MAP.

SELECTION_CRITERIA or *SC*

File from which selection criteria subcommands are read. You can specify a file position as part of the file name. To enter selection criteria subcommands interactively, specify COMMAND. If SELECTION_CRITERIA is omitted, no selection criteria processing is performed.

WIDTH or *W*

Length of the expanded lines, excluding line identifiers. If WIDTH is omitted, SCU uses 0 (zero) for the default line width. A line width of 0 (zero) means that lines can be up to 256 characters (with no trailing blanks) when the file is expanded.

LINE_IDENTIFIER or *LI*

Line identifier placement.

RIGHT (R)

Line identifiers are placed to the right of the text.

LEFT (L)

Line identifiers are placed to the left of the text.

NONE

No line identifiers are placed on output lines.

If LINE_IDENTIFIER is omitted, NONE is used.

ALTERNATE_BASE or *ALTERNATE_BASES* or *AB*

Optional list of one or more additional libraries to be searched for decks.

LIST or *L*

Listing file. You can specify a file position as part of the file name. Within an SCU session, if LIST is omitted, the listing file is the file specified on the SET_LIST_OPTIONS subcommand. Otherwise, the default is file $LIST.

EXPANSION_DEPTH or *ED*

Number of levels of COPY and COPYC directives to process. COPY and COPYC directives beyond the maximum expansion depth are expanded as text. If EXPANSION_DEPTH is omitted, COPY and COPYC directives are processed whenever they are encountered.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Indicates whether the listing includes the library for each deck from which the deck was expanded.

BRIEF (B)

Does not list the decks or their library origins.

FULL (F)

Lists the library origin when more than one library is used. ALL is an alias for FULL.

If DISPLAY_OPTIONS is omitted, BRIEF is used.

Remarks

- To expand a deck, use the EXPAND_DECK subcommand.
- To expand a file while not in an SCU session, use the EXPAND_SOURCE_FILE command.
- You can specify alternate base libraries with the ALTERNATE_BASE parameter. When SCU processes a COPY or COPYC directive, it first searches the deck list of the working library for the deck specified on the directive and then it searches the deck lists of the alternate base libraries in the order the libraries are listed on the ALTERNATE_BASE parameter.
- The EXPANSION_DEPTH parameter can limit the levels of nested directives processed. If SCU reads a directive at a level beyond the maximum level processed, it expands it as text.
- The LINE_IDENTIFIER, WIDTH, and ORDER parameters affect how the expanded text is written on the compile file.
- The line width can be specified by the WIDTH parameter. If the line width for a file or deck is 0 (zero), EXPAND_FILE writes each line as it is stored in the file or deck (no trailing blanks or truncation); a blank line, therefore, is written as a zero-length V record. If the line width for a file or a deck is nonzero, EXPAND_FILE writes each line using that width. Lines shorter than the width are padded with trailing blanks; lines longer than the width are truncated.

- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand expands the text of file NEW_TEXT and writes the expanded text on file COMPILE. The unique name given to the temporary deck created from file NEW_TEXT is $82 .. 17.

```
sc/expand_file f=new_text do=full l=output
*=Deck was copied
 $821497P3S0002D19860305T110817 Working Library
```

EXTRACT_DECK SCU Subcommand

Purpose Extracts one or more decks. Extracting a deck copies the deck text to another file without processing directives embedded in the text. No delimiter is written between extracted decks.

Format

EXTRACT_DECK or
EXTRACT_DECKS or
EXTD
DECK=keyword or *list of: name* or *range of name*
SOURCE=file
SELECTION_CRITERIA=file
WIDTH=integer
LINE_IDENTIFIER=keyword
ALTERNATE_BASE=list of file
LIST=file
DISPLAY_OPTIONS=keyword
ORDER=keyword
EXPAND=keyword or *boolean*
DECK_DIRECTIVES_INCLUDED=boolean
STATUS=status variable

Parameters *DECK* or *DECKS* or *D*

Decks to be extracted. You can specify a list of one or more names, a list of one or more ranges, or the keyword ALL. ALL specifies all decks in the working library and in any alternate base libraries specified on the ALTERNATE_BASE parameter. If DECK is omitted, the last used deck is extracted. To prevent the last used deck from being extracted, specify NONE on the DECK

parameter. In that case, SCU determines the decks extracted by the subcommands entered via the selection criteria file.

SOURCE or *S*

File on which the extracted text is written. You can specify a file position as part of the file name. If SOURCE is omitted, file SOURCE is used.

SELECTION_CRITERIA or *SC*

File from which selection criteria commands are read. You can specify a file position as part of the file name. If SELECTION_CRITERIA is omitted, no selection criteria processing is performed, and the decks extracted are determined by the DECK parameter.

WIDTH or *W*

Length of the extracted lines, excluding line identifiers. If WIDTH is omitted, the default line width for each deck is used.

LINE_IDENTIFIER or *LI*

Line identifier placement. Options are:

RIGHT (R)

Line identifiers are placed to the right of the text.

LEFT (L)

Line identifiers are placed to the left of the text.

NONE

No line identifiers are placed on output lines.

If LINE_IDENTIFIER is omitted, the default line identifier placement for each deck is used.

ALTERNATE_BASE or *ALTERNATE_BASES* or *AB*

Optional list of one or more additional libraries to be searched for decks.

LIST or *L*

Listing file. You can specify a file position as part of the file name. Within an SCU session, if LIST is omitted, the listing file is the file specified on the SET_LIST_OPTIONS subcommand. Otherwise, the default is file $LIST.

DISPLAY_OPTIONS or *DISPLAY_OPTION* or *DO*

Indicates whether the listing includes the library for each deck from which the deck was extracted. Options are:

BRIEF (B)

Does not list the decks or their library origins.

FULL (F)

Lists the library origin when more than one library is used. ALL is an alias for FULL.

If DISPLAY_OPTIONS is omitted, BRIEF is used.

ORDER or *O*

Indicates whether the decks are extracted in the order specified or in alphabetical order. Options are:

COMMAND (C)

Decks are extracted in the order specified on the subcommand.

LIBRARY (L)

Decks are extracted in alphabetical order.

If ORDER is omitted, LIBRARY is used.

EXPAND or *E*

Indicates the required expand attribute for each deck extracted. Options are:

TRUE

Expand attribute must be TRUE.

FALSE

Expand attribute must be FALSE.

ALL

Expand attribute can be either TRUE or FALSE.

If EXPAND is omitted, ALL is used.

DECK_DIRECTIVES_INCLUDED or *DDI*

Indicates whether a DECK directive precedes each extracted deck on the source file. Options are:

TRUE

A DECK directive is written before each deck.

FALSE

No DECK directives are written.

If DECK_DIRECTIVES_INCLUDED is omitted, FALSE is used.

33

Remarks

- The EXTRACT_DECK subcommand has the same deck selection options as the EXPAND_DECK subcommand. You can select the decks extracted by name, by selection criteria, or by both. However, unlike the EXPAND_DECK subcommand, you can also choose whether to use the expand deck attribute to select the decks to be extracted. With the EXPAND parameter, you can choose to extract decks whose expand attribute is TRUE, FALSE, or either TRUE or FALSE.

- You can use the extracted text as the source text when creating new decks. To include a DECK directive before the source text of each deck, specify DECK_DIRECTIVES_INCLUDED=TRUE on the subcommand. Using the embedded DECK directives, the decks created using the source text file will have the same names and expand attributes as the original decks.

- The EXTRACT_DECK subcommand does not save any of the deck header information such as DECK_DESCRIPTION. You must re-enter this information manually when you add the deck to the new library.

- You can specify alternate base libraries with the ALTERNATE_BASE parameter. SCU first searches the deck list of the working library for the deck and then searches the deck lists of the alternate base libraries in the order the libraries are listed on the ALTERNATE_BASE parameter.

- The LINE_IDENTIFIER, WIDTH, and ORDER parameters affect how the extracted text is written on the source file. The LINE_IDENTIFIER and WIDTH parameters can override the default values in the deck headers. The ORDER parameter allows you to specify the order that SCU writes the decks on the file.

- The line width can be specified by the WIDTH parameter. If the line width for a deck is 0 (zero), EXTRACT_DECK writes each line as it is stored in the deck (no trailing blanks or truncation); a blank line, therefore, is written as a zero-length V record. If the line width for a deck is nonzero, EXTRACT_DECKS writes each line using that width. Lines shorter than the width are padded with trailing blanks; lines longer than the width are truncated.

- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand extracts the text of deck FORTRAN_TEXT and writes the text on file SOURCE.

```
sc/extract_deck d=fortran_text do=full l=output
 FORTRAN_TEXT                    :NVE.PAT.MY_LIBRARY
```

EXTRACT_MODIFICATION SCU Subcommand

Purpose Generates a sequence of EDIT_FILE utility subcommands (INSERT_LINES, DELETE_LINES, and REPLACE_LINES subcommands) that, if processed, would introduce the modification changes.

Format

EXTRACT_MODIFICATION or
EXTRACT_MODIFICATIONS or
EXTM
MODIFICATION=list of: name or *range of name*
EDIT_COMMANDS=file
DECK=keyword or *list of: name* or *range of name*
TERMINATING_DELIMITER=string
LINES=keyword
STATUS=status variable

Parameters

MODIFICATION or *MODIFICATIONS* or *M*

Modifications to be extracted. If MODIFICATION is omitted, the last used modification is extracted.

EDIT_COMMANDS or **EC**

File to which the text and editing commands are written. You can specify a file position as part of the file name. This parameter is required.

DECK or *D*

Indicates the deck or decks to which the extracted modification lines should apply. ALL specifies all all decks. If DECK is omitted, ALL is used.

TERMINATING_DELIMITER or *TD*

Delimiter string used to mark the end of inserted text (from 1 to 31 characters). If TERMINATING_DELIMITER is omitted, '///END\\\' is used.

LINES or *LINE* or *L*

Indicates whether the editor subcommands reference only active lines or all lines. If LINES is omitted, ACTIVE is used.

ACTIVE

Limits processing of the generated sequence of subcommands to only active lines. When the generated file is processed, an error occurs if an inactive line is referenced.

ALL

Allows the processing to reference inactive lines.

Remarks

- The EXTRACT_MODIFICATION subcommand writes the editing commands and inserted text that make up a modification on a file. EXTM does not save any of the modification header information such as the author name or feature name. You must re-enter this information when you add the modification to the new library.

- You can process another deck using the editing commands saved on a file by the EDIT_COMMANDS parameter of the EXTRACT_MODIFICATION subcommand. After editing the deck, you execute an INCLUDE_FILE command using the file name of the file containing the editing commands.

- Before deleting a modification, you can use the EXTRACT_MODIFICATION subcommand to save the modification changes on a separate file. You could then reintroduce the modification by processing the editing commands on the file.

- If you extract a modification and delete it, then re-apply it using the extracted editing commands, you will get an equivalent deck only if the extracted modification was not inter-related with another modification. For example, if two modifications each delete lines introduced by the other, re-applying the editing commands extracted from one modification will not reproduce the same deck.

  You can extract, delete, and re-apply the last modification applied to a deck only if no other modifications have deleted lines introduced under that name. You can repeatedly extract and delete the last modification made to a deck, then re-apply them in the order of the deck's modification list if no modification deletes lines introduced by a modification later in the deck's list. Therefore, you should always alter a deck with the last modification name. More modifications can be altered only if they are independent of one another.

  If it is important that lines continue to have the same sequence numbers, use the SEQUENCE_MODIFICATION subcommand before re-applying each set of extracted editing commands.

- The subcommands can also extract only the modification changes that apply to one or more decks in the working library. To do so, specify the decks on the DECK parameter.

- If more than one modification is specified on the EXTRACT_MODIFICATION subcommand, the sequence of subcommands generated, if executed, would produce the combined modification changes.

- The EXTRACT_MODIFICATION subcommand is valid within an editing session started by an EDIT_DECK subcommand, but the modification changes extracted do not include any changes made since you last started editing the deck.

- The LINES parameter can specify ACTIVE or ALL. If ALL is specified, the generated file automatically contains the following lines at its beginning:

```
ESV$INITIAL_LINES_ENABLED=$LINES_ENABLED
ENABLE_LINES LINES=ALL
```

  When the file is executed, the first line saves the user's current setting for enabling lines (by default, only active lines are enabled.) The second line enables all active and inactive lines. The generated file also automatically contains the following line at its end:

```
ENABLE_LINES LINES=ESV$INITIAL_LINES_ENABLED
```

  Executing this subcommand returns the editing session to the value for the LINES parameter that was enabled before the file was executed.

- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand extracts modification MOD1 onto file SAVE_MOD1.

```
sc/extract_modification m=mod1 ec=save_mod1
```

$FEATURE
SCU Function

Purpose Returns a boolean value indicating whether the specified name is recognized as a feature on the working library.

Format

$FEATURE
(FEATURE: name)

Parameters

FEATURE

Name of the feature to be found. This parameter is required.

Remarks For more information, see the NOS/VE Source Code Management manual.

Examples The following command assigns a boolean value to the SCL variable FEATURE_EXISTS, depending on whether FEATURE1 is recognized as a feature in the working library.

```
sc/feature_exists=$feature(feature1)
```


$FEATURE_MEMBER_NAMES
SCU Function

Purpose Returns a list of names of modifications on the working library that belong to the specified feature.

Format

$FEATURE_MEMBER_NAMES
(FEATURE: name)

Parameters

FEATURE

Name of the feature. This parameter is required.

Remarks

- The names in the list appear in the same order as the names in the modification list on the working library.
- For more information, see the NOS/VE Source Code Management manual.

$FEATURE _NAME _LIST
Selection Criteria Function

Purpose Returns a list of feature names on the working library for those decks currently selected to be written to the result file.

Format **$FEATURE _NAME _LIST**

Parameters None.

Remarks

- The list of feature names is ordered the same as it is on the working library.
- When used in selection criteria subcommand processing, $FEATURE _NAME _LIST reflects the current feature list to be written to the compile, result, or source file being produced.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following command assigns a list of names of features currently selected to the variable FEATURE _LIST.

```
scc/feature_list=$feature_name_list
```

$FEATURE _NAME _LIST
SCU Function

Purpose Returns a list of feature names on the working library.

Format **$FEATURE _NAME _LIST**

Parameters None.

Remarks

- The list of feature names is ordered the same as it is on the working library.
- When used in selection criteria subcommand processing, $FEATURE _NAME _LIST returns the names of currently selected features.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following command assigns a list of names of features on the working library to the variable FEATURE_LIST.

```
sc/feature_list=$feature_name_list
```

$FIRST_DECK_NAME SCU Function

Purpose Returns the name of the first deck in the working library.

Format **$FIRST_DECK_NAME**

Parameters None.

Remarks

- For more information, see the NOS/VE Source Code Management manual.

$FIRST_MODIFICATION_NAME SCU Function

Purpose Returns the name of the first modification in the library modification list.

Format **$FIRST_MODIFICATION_NAME**

Parameters None.

Remarks

- The modification list is in alphabetical order.
- For more information, see the NOS/VE Source Code Management manual.

$GROUP SCU Function

Purpose Returns a boolean value indicating whether a name is recognized as a group in the working library.

Format

$GROUP
(GROUP: name)

Parameters **GROUP**

Name of the group to be searched for on the working library. This parameter is required.

Remarks For more information, see the NOS/VE Source Code Management manual.

Examples The following command assigns a boolean value to the variable GROUP_EXISTS, indicating whether the group TEST exists on the working library.

```
sc/group_exists=$group(test)
```

$GROUP_MEMBER_NAMES SCU Function

Purpose Returns a list of the names of decks on the working library that belong to the specified group.

Format

```
$GROUP_MEMBER_NAMES
   (GROUP: name)
```

Parameters

GROUP

Name of the group. This parameter is required.

Remarks

- The order of names is the same as on the working library.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following command expands those decks on the current library that were changed by a particular modification and are members of the group CYBIL.

```
sc/expand_decks ..
sc../d=$intersection($group_member_names(cybil),..
sc../$modified_deck_names(sc8a751_3))
```

$GROUP_NAME_LIST Selection Criteria Function

Purpose Returns a list of group names on the working library for those decks currently selected to be written to the result file.

Format

```
$GROUP_NAME_LIST
```

Parameters None.

Remarks

- The list of group names is ordered the same as it is on the working library.
- When used in selection criteria subcommand processing, $GROUP_NAME_LIST reflects the current group list to be written to the compile, result, or source file being produced.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following command assigns a list of names of groups currently selected to the variable GROUP_LIST.

```
scc/group_list=$group_name_list
```

$GROUP_NAME_LIST SCU Function

Purpose Returns a list of group names on the working library.

Format **$GROUP_NAME_LIST**

Parameters None.

Remarks

- The list of group names is ordered the same as it is on the working library.
- When used in selection criteria subcommand processing, $GROUP_NAME_LIST returns the names of currently selected groups.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following command assigns a list of names of groups on the working library to the variable GROUP_LIST.

```
sc/group_list=$group_name_list
```

INCLUDE_COPYING_DECKS Selection Criteria Subcommand

Purpose Explicitly includes all decks that contain a COPY or COPYC directive that directly or indirectly copies one of the specified decks.

Format

INCLUDE_COPYING_DECKS or
INCCD
DECK=list of: name or range of name
DECK_RESIDENCE=keyword
STATUS=status variable

Parameters

DECK or DECKS or D

Decks copied by the included decks. This parameter is required.

DECK_RESIDENCE or *DR*

Specifies whether the decks specified on the DECK parameter reside either on the working library or on alternate base libraries used by the subcommand. Options are:

EXTERNAL

The decks do not reside on the libraries.

INTERNAL

The decks reside on the libraries.

If DECK_RESIDENCE is omitted, INTERNAL is used.

Remarks

- The INCLUDE_COPYING_DECKS subcommand allows you to expand or extract only those decks that reference the specified decks.
- For more information, see the NOS/VE Source Code Management manual.

Examples

The following subcommand sequence expands all decks that copy deck COMMON1.

```
sc/expand_decks selection_criteria=command
scc/include_copying_decks deck=common1
scc/quit
```

INCLUDE_DECK
Selection Criteria Subcommand

Purpose Explicitly includes one or more decks.

Format

INCLUDE_DECK or
INCLUDE_DECKS or
INCD
DECK=**list of: name** or **range of name**
STATUS=*status variable*

Parameters

DECK or **DECKS** or **D**

Decks to be included. This parameter is required.

Remarks

- If a deck name in a deck list is in error, the subcommand is not executed.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand sequence excludes all decks in group GROUP1, but includes deck WANTED even if it belongs to GROUP1.

```
sc/expand_decks decks=all selection_criteria=command
scc/exclude_group group1
scc/include_deck wanted
scc/quit
```

INCLUDE_FEATURE
Selection Criteria Subcommand

Purpose Includes all modifications belonging to one or more features.

Format

INCLUDE_FEATURE or
INCLUDE_FEATURES or
INCF
FEATURE=**list of name**
STATE=*integer*
STATUS=*status variable*

Parameters **FEATURE or FEATURES or F**

Features to be included. This parameter is required.

STATE or S

Minimum state (0 through 4) of the modifications included. All modifications whose state is greater than or equal to the specified state are included. If STATE is omitted, all modifications belonging to the feature are included.

Remarks For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand sequence expands DECK1 through DECK5. It includes all modifications belonging to feature NEW_PROMPTS that have a state of 2, 3, or 4.

```
sc/expd decks=deck1..deck5 selection_criteria=command
scc/include_feature feature=new_prompts state=2
scc/quit
```

INCLUDE_GROUP
Selection Criteria Subcommand

Purpose Explicitly includes decks belonging to one or more groups.

Format

INCLUDE_GROUP or
INCLUDE_GROUPS or
INCG
GROUP=list of name
COMBINATION=keyword
STATUS=status variable

Parameters **GROUP or GROUPS or G**

Groups to be included. This parameter is required.

COMBINATION or C

Indicates whether the decks included must belong to any or all specified groups. Options are:

ANY

Included decks must belong to at least one of the specified groups.

ALL

Included decks must belong to all of the specified groups.

If COMBINATION is omitted, ANY is used.

Remarks For more information, see the NOS/VE Source Code Management manual.

Examples The following command sequence extracts all decks belonging to group SECTION_1.

```
sc/extract_decks selection_criteria=command
scc/include_group group=section_1
scc/quit
```

INCLUDE_MODIFICATION Selection Criteria Subcommand

Purpose Explicitly includes one or more modifications.

Format

INCLUDE_MODIFICATION or
INCLUDE_MODIFICATIONS or
INCM
MODIFICATION=list of name
STATUS=status variable

Parameters **MODIFICATION** or **MODIFICATIONS** or **M**

Modifications to be included. This parameter is required.

Remarks For more information, see the NOS/VE Source Code Management manual.

Examples The following command sequence expands all text on deck DECK5 except those lines belonging to feature MY_CHANGES. However, lines belonging to modifications MOD2 and MOD5 are expanded even if the modifications are associated with feature MY_CHANGES.

```
sc/expand_deck deck=deck5 selection_criteria=command
scc/exclude_feature my_changes
scc/include_modifications (mod2,mod5)
scc/quit
```

INCLUDE _MODIFIED _DECKS Selection Criteria Subcommand

Purpose Explicitly includes all decks that are modified by a specified feature or modification. Decks directly modified are always included. Decks which copy modified decks (directly or indirectly through chains of indirect references) can also be optionally included.

Format

INCLUDE _MODIFIED _DECKS or
INCLUDE _MODIFIED _DECK or
INCMD
FEATURES=list of: name or *range of name*
MODIFICATIONS=list of: name or *range of name*
INCLUDE _COPYING _DECKS=boolean
STATUS=status variable

Parameters

FEATURES or *FEATURE* or *F*

Name of features to be included. If FEATURE is omitted, MODIFICATION must be specified.

MODIFICATIONS or *MODIFICATION* or *M*

Names of modifications to be included. If MODIFICATION is omitted, FEATURE must be specified.

INCLUDE _COPYING _DECKS or *ICD*

Specifies whether decks that copy modified decks should be included. If INCLUDE_COPYING_DECKS is omitted, decks that copy modified decks are not included.

Remarks For more information, see the NOS/VE Source Code Management manual.

Examples The following example includes all of the decks modified by the modification ACCOUNTING_FIXES and all the decks that copy modified decks.

```
scc/include_modified_decks ..
scc../feature=accounting_fixes ..
scc../include_copying_decks=true
```

INCLUDE_STATE
Selection Criteria Subcommand

Purpose Includes all modifications whose state is greater than or equal to that specified.

Format

```
INCLUDE_STATE or
INCS
    STATE=integer
    STATUS=status variable
```

Parameters

STATE or S

Minimum state (from 0 through 4) of the modifications included. All modifications whose state is greater than or equal to the specified value are included. This parameter is required.

Remarks For more information, see the NOS/VE Source Code Management manual.

Examples The following command sequence extracts all lines in DECK5 belonging to modifications whose state is 2, 3, or 4.

```
sc/extract_deck deck=deck5 selection_criteria=command
scc/include_state 2
scc/quit
```

$LAST_DECK_NAME
SCU Function

Purpose Returns the name of the last deck on the working library.

Format $LAST_DECK_NAME

Parameters None.

Remarks

- For more information, see the NOS/VE Source Code Management manual.

$LAST_MODIFICATION_NAME SCU Function

Purpose Returns the name of the last modification in the library modification list.

Format **$LAST_MODIFICATION_NAME**

Parameters None.

Remarks

- The modification list is in alphabetical order.
- For more information, see the NOS/VE Source Code Management manual.

$LIBRARY_ATTRIBUTES SCU Function

Purpose Returns the content of a source library header. The value is returned as a record.

Format **$LIBRARY_ATTRIBUTES**

Parameters None.

Remarks

- The attributes have the following types:

  CHANGE_COUNTER: integer
  CREATION_DATE_TIME: date_time
  DECK_COUNT: integer
  FEATURE_COUNT: integer
  FILE: file
  GROUP_COUNT: integer
  KEY: string
  LAST_USED_DECK: name
  LAST_USED_MODIFICATION: name
  LIBRARY: name
  LIBRARY_DESCRIPTION: list of string
  LIBRARY_FORMAT_VERSION: string
  MODIFICATION_COUNT: integer
  MODIFICATION_DATE_TIME: date_time
  SCU_VERSION: string
  VERSION: string

- To use the contents of the header returned, it is best to create a variable implicitly, for example, LA=$LIBRARY_ATTRIBUTES.

- If the library has not been modified, the creation values are returned in the MODIFICATION_DATE_TIME field.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following example uses the $LIBRARY_ATTRIBUTES function to determine which of two libraries has been modified most recently.

```
sc/use_library b=intve.scu.source_library r=$null
sc/scu_header=$library_attributes
sc/end_library wl=no
sc/use_library b=$system.cybil.osf$program_interface
sc/interface_header=$library_attributes
sc/scu_is_more_recent=..
sc../scu_header.modification_date_time > ..
sc../interface_header.modification_date_time
```


$LIBRARY_MODIFIED SCU Function

Purpose Returns a boolean value indicating whether the current working library has been modified.

Format **$LIBRARY_MODIFIED**

Parameters None.

Remarks

- The value of $LIBRARY_MODIFIED is set to FALSE when you enter one of the subcommands WRITE_LIBRARY, END_LIBRARY, or USE_LIBRARY. The value is set to TRUE when you make a change to the current working library that is not recorded on an external file.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following command assigns a boolean value to the SCL variable LIBRARY_CHANGED, depending on whether the current working library has been modified.

```
sc/library_changed=$library_modified
```

$LIST_FILE
SCU Function

Purpose Returns the default listing file for the LIST parameter on SCU subcommands.

Format **$LIST_FILE**

Parameters None.

Remarks For more information, see the NOS/VE Source Code Management manual.

Examples The following command displays the current value of the default listing file.

```
/scu
sc/set_list_options l=$user.fortran_list_file
sc/display_value v=$list_file
:NVE.PAT.FORTRAN_LIST_FILE
```

$MODIFICATION
SCU Function

Purpose Returns a boolean value indicating whether the specified modification is in the working library.

Format **$MODIFICATION**
(MODIFICATION: name)

Parameters **MODIFICATION**

Name of the modification to be found. This parameter is required.

Remarks

- If you exclude the specified modification using a selection criteria command, SCU evaluates the $MODIFICATION function as FALSE.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following command assigns a boolean value to the SCL variable MOD_EXISTS, depending on whether MOD1 is in the working library.

```
sc/mod_exists=$modification(mod1)
```

$MODIFICATION_ATTRIBUTES
SCU Function

Purpose Returns the content of an SCU modification header. The value is returned as a record.

Format

$MODIFICATION_ATTRIBUTES
(*MODIFICATION: name*)

Parameters *MODIFICATION*

Name of the modification for which the header content is returned. If MODIFICATION is omitted, the last used modification is assumed.

Remarks

- The attributes have the following types:

  AUTHOR: string
  CREATION_DATE_TIME: date_time
  FEATURE: name
  MODIFICATION_DESCRIPTION: list of name
  MODIFICATION_DATE_TIME: date_time
  NAME: name
  STATE: integer

- To use the contents of the header returned, it is best to create a variable implicitly, for example, MA1=$MODIFICATION_ATTRIBUTES(MOD1).

- If you use the $MODIFICATION_ATTRIBUTES function to assign attributes to a variable in an iterative process, you must delete and re-create the variable for each iteration. The existing variable cannot be re-assigned the attributes of a different modification.

- If the modification has not been modified, the creation values are returned in the MODIFICATION_DATE_TIME field.

- For more information, see the NOS/VE Source Code Management manual.

$MODIFICATION _NAME _LIST Selection Criteria Function

Purpose Returns a list of names of modifications on the working library for those decks currently selected to go to the result file.

Format **$MODIFICATION _NAME _LIST** or **$MNL**

Parameters None.

Remarks

- The list of modification names is ordered alphabetically, as it is on the working library.
- When used in selection criteria subcommand processing, $MODIFICATION _NAME _LIST reflects the current modification list to be written to the compile, result, or source file being produced.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following command assigns a list of modifications currently selected to the variable MODIFICATION _LIST.

```
scc/modification_list=$modification_name_list
```

$MODIFICATION _NAME _LIST SCU Function

Purpose Returns a list of names of modifications on the working library.

Format **$MODIFICATION _NAME _LIST**

Parameters None.

Remarks

- The list of modification names is ordered the same as it is on the working library.
- When used in selection criteria subcommand processing, $MODIFICATION _NAME _LIST returns the names of currently selected modifications.

- For more information, see the NOS/VE Source Code Management manual.

Examples The following example returns a list of all modifications on a library which have changed since the beginning of 1989.

```
sc/VAR
var/mnl:  list of name
var/start_of_1989:  date=1989-1-1
var/VAREND

sc/mnl=$modification_name_list

sc/mods_in_89=$select(mnl, ..
sc../($modification_attributes(x).modification_..
sc../date_time > start_of_1989))
```

$MODIFIED_DECK_NAMES SCU Function

Purpose Returns a list of names of decks on the working library affected by the specified modification.

Format

$MODIFIED_DECK_NAMES
(MODIFICATION: name)

Parameters

MODIFICATION

Name of the modification. This parameter is required.

Remarks

- The order of names is the same as on the working library.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following example displays active and inactive text for all decks affected by the specified modification.

```
sc/display_decks d=$modified_deck_names(sc8a751) ..
sc../do=all t=i
```

$NEXT_DECK_NAME SCU Function

Purpose Returns the name of the next deck.

Format

```
$NEXT_DECK_NAME
    (DECK: name)
```

Parameters

DECK

Name of the deck whose successor is to be found. This parameter is required.

Remarks

- For more information, see the NOS/VE Source Code Management manual.

$NEXT_MODIFICATION_NAME SCU Function

Purpose Returns the name of the next modification in the library modification list.

Format

```
$NEXT_MODIFICATION_NAME
    (MODIFICATION: name)
```

Parameters

MODIFICATION

Name of the modification whose successor is to be found. This parameter is required.

Remarks

- For more information, see the NOS/VE Source Code Management manual.

$PREVIOUS_DECK_NAME SCU Function

Purpose Returns the name of the previous deck.

Format

```
$PREVIOUS_DECK_NAME
    (DECK: name)
```

Parameters

DECK

Name of the deck whose predecessor is to be found. This parameter is required.

Remarks

- The returned name is uppercase, even when the original entry was lowercase.
- For more information, see the NOS/VE Source Code Management manual.

$PREVIOUS_MODIFICATION_NAME SCU Function

Purpose Returns the name of the previous modification in the library modification list.

Format

$PREVIOUS_MODIFICATION_NAME
(MODIFICATION: name)

Parameters

MODIFICATION

Name of the modification whose predecessor is to be found. This parameter is required.

Remarks

- The returned name is uppercase, even when the original entry was lowercase.
- For more information, see the NOS/VE Source Code Management manual.

QUIT Selection Criteria Subcommand

Purpose Ends SELECTION_CRITERIA_COMMAND command processing.

Format

QUIT or
END or
QUI
STATUS=status variable

Remarks For more information, see the NOS/VE Source Code Management manual.

QUIT
SCU Subcommand

Purpose Ends an SCU session and optionally writes the working library to the result source library.

Format

QUIT or
END or
QUI
WRITE_LIBRARY=boolean
STATUS=status variable

Parameters *WRITE_LIBRARY* or *WL*

Indicates whether SCU should generate a result library from the working library.

TRUE

SCU generates a result library.

FALSE

SCU does not generate a result library.

If WRITE_LIBRARY is omitted, TRUE is used.

Remarks

- The QUIT subcommand indicates whether SCU should generate a result library from the working library. If a library is to be generated, SCU writes the result library on the result library file specified on a CREATE_LIBRARY or USE_LIBRARY subcommand at the beginning of the session. If a WRITE_LIBRARY subcommand specifies a different result library, SCU writes the result library on the file specified by the last WRITE_LIBRARY subcommand. If none of these subcommands are specified, the result library is written on file SOURCE_LIBRARY in your working catalog.

- If the result file is the same as the file named on the BASE parameter of the USE_LIBRARY subcommand, it is rewritten only when the result library has been modified.

- Refer to WRITE_LIBRARY and END_LIBRARY for other subcommands that write a result library.

- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand ends an SCU session and generates a result library.

```
sc/quit wl=true
```

The following sequence changes and rewrites the source library and then ends the SCU session.

```
/source_code_utility
sc/use_library b=$user.my_library
sc/change_deck d=deck1 a='roger'
sc/quit
```

REPLACE_LIBRARY SCU Subcommand

Purpose Replaces decks on the working library with decks from one or more source libraries.

Format

REPLACE_LIBRARY or
REPLACE_LIBRARIES or
REPL
SOURCE_LIBRARY=list of file
LIST=file
DISPLAY_OPTIONS=keyword
ENFORCE_INTERLOCKS=boolean
STATUS=status variable

Parameters

SOURCE_LIBRARY or **SOURCE_LIBRARIES** or **SL**

List of one or more source library names. This parameter is required.

LIST or *L*

Listing file. You can specify a file position as part of the file name. SCU lists the source library origin of each deck in the working library. If LIST is omitted, the listing file is the file specified on the SET_LIST_OPTIONS subcommand. Otherwise, the default is file $LIST.

DISPLAY_OPTIONS or *DO*

Specifies the information listed. Currently, both of the following keywords produce the same listing.

BRIEF or B
FULL or F

If DISPLAY_OPTIONS is omitted, BRIEF is used. ALL is an alias for FULL.

ENFORCE_INTERLOCKS or *EI*

Indicates whether the interlocks must match before a deck can replace a base library deck. Options are:

TRUE

Interlocks must match.

FALSE

Interlocks need not match.

If ENFORCE_INTERLOCKS is omitted, FALSE is used.

Remarks

- REPLACE_LIBRARIES reads the source library deck lists in the order you specify the libraries on the command.
- After reading a deck name, REPLACE_LIBRARIES determines if the deck name is in the working library deck list. If the name is in the list, it replaces the deck in the working library with the deck from the source library. If the name is not in the list, the command does not add the deck to the working library, but it sends a warning message, stating that the deck cannot be replaced because it is not in the working library.
- If no decks could be merged because an exception occurred in each deck, an error status is returned and REPLACE_LIBRARY makes no change to the library.
- REPLACE_LIBRARIES lists the source library origin of each deck in the working library on the listing file.
- Decks, features, groups, and modifications are ordered alphabetically on the REPLACE_LIBRARIES result library.

- You can use this subcommand to merge decks from an extracted library with decks from the original library from which it was extracted to form a new library. You use this command if you do not want to add any new decks to the new library.

  If you set interlocks when you extracted the library, REPLACE_LIBRARY enforces the interlock if you specify ENFORCE_INTERLOCKS=TRUE in the subcommand. Interlock enforcement means that REPLACE_LIBRARY checks whether the original interlock value in the header of the extracted deck copy matches the subinterlock value in the header of the working library copy. If the values match, REPLACE_LIBRARY replaces the working library deck with the extracted deck; otherwise, it does not replace the working library deck.

- Key characters in source libraries that are added to the working library must match the key character in the working library. If the key characters do not match, SCU generates an error message.

- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand replaces decks on the working library with decks from source library NEWLIB.

```
sc/replace_library sl=newlib l=output
 DECKA                    :NVE.PAT.SOURCE_LIBRARY
 DECKB                    :NVE.PAT.NEWLIB
 DECKC                    :NVE.PAT.NEWLIB
 DECKD                    :NVE.PAT.SOURCE_LIBRARY
```

$RESULT
SCU Function

Purpose Returns the result library file.

Format **$RESULT**

Parameters None.

Remarks

- The value of $RESULT is updated when a WRITE_LIBRARY subcommand is entered that specifies a result file.
- For more information, see the NOS/VE Source Code Management manual.

Examples

The following command displays the current value of the result file.

```
/source_code_utility
sc/use_library b=$user.fortran_lib ..
sc../r=$user.new_fortran_lib
sc/display_value v=$result
:NVE.PAT.NEW_FORTRAN_LIB
```

RETAIN_GROUP
Selection Criteria Subcommand

Purpose

Retains from the list of decks currently selected only those decks that are members of the specified group.

Format

RETAIN_GROUP or
RETAIN_GROUPS or
RETG
GROUP=list of name
COMBINATION=keyword
STATUS=status variable

Parameters

GROUP or **GROUPS** or **G**

Names of the groups to be retained. This parameter is required.

COMBINATION or *C*

Decks to be retained. Options are:

ANY

Decks will be retained if they are members of any of the groups specified by the GROUP parameter.

ALL

Decks will be retained if they are members of all of the groups specified by the GROUP parameter.

If COMBINATION is omitted, ANY is used.

Remarks For more information, see the NOS/VE Source Code Management manual.

Examples The following example retains the decks which are at the same time members of group CYBIL and group SCF$UNBOUND_UTILITY.

```
scc/retain_groups g=(cybil,scf$unbound_utility) ..
scc../c=all
```

SEQUENCE_DECK
SCU Subcommand

Purpose Sequences deck lines in released state (state 4).

Format

SEQUENCE_DECK or
SEQUENCE_DECKS or
SEQD
DECK=list of: name or **range of name**
MODIFICATION=keyword or *name*
STATUS=status variable

Parameters

DECK or **DECKS** or **D**

Decks to be sequenced. You can specify a list of one or more names, a list of one or more ranges, or the keyword ALL. ALL specifies all decks in the working library. This parameter is required.

MODIFICATION or *M*

Modification name that is used in the line identifiers for resequenced lines. If the modification already exists, it must be in state 4.

You specify that the creation modification is to be used for each deck by specifying the keyword CREATION_MODIFICATION.

If MODIFICATION is omitted, the creation modification for each deck is used.

Remarks

- To sequence a deck, you must have authority 4 for the file. The creation modification for each sequenced deck must be in state 4.
- The subcommand only sequences lines belonging to modifications in state 4. Each sequenced line is assigned a new line identifier. The line identifier consists of the name of the specified modification and a sequence number. The sequence numbers are assigned in the order the lines appear within the source library.
- After sequencing, all sequenced lines belong to the specified modification. The maximum sequence number is 16,777,214.
- If a sequenced deck has its subinterlock set, SCU reports a warning message.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand sequences all decks in the working library.

```
sc/sequence_deck d=all
```

SEQUENCE_MODIFICATION SCU Subcommand

Purpose Sequences modification lines.

Format

SEQUENCE_MODIFICATION or
SEQUENCE_MODIFICATIONS or
SEQM
MODIFICATION=list of: name or **range of name**
DECK=keyword or *list of: name* or *range of name*
STATUS=status variable

Parameters

MODIFICATION or MODIFICATIONS or M

Modifications to be resequenced. This parameter is required.

DECK or *DECKS* or *D*

One or more decks. You can specify a list of one or more names, a list of one or more ranges, or the keyword ALL. ALL specifies all decks in the working library. If DECK is specified, only the modification lines that apply to the specified decks are sequenced. If DECK is omitted, ALL is used.

Remarks

- The sequenced modifications must be in state 0 (zero).
- Before sequencing, the sequence numbers in the line identifiers of a modification are ordered as the lines were added to the modification. After sequencing, the sequence numbers in the line identifiers are ordered as the lines appear in the deck. The maximum sequence number is 16,777,214.
- If a sequenced deck has its interlock set, SCU sends a warning message.
- You can specify the DECK parameter to limit sequencing to lines in the specified decks.
- For more information, see the NOS/VE Source Code Management manual.

Examples

The following subcommand sequences modification MOD5.

```
sc/sequence_modification m=mod5
```

SET_LIST_OPTIONS SCU Subcommand

Purpose

Establishes a default for the LIST parameters on SCU subcommands. It also specifies the file to which intermediate diagnostic messages are written.

Format

SET_LIST_OPTIONS or
SETLO
LIST=file
ERRORS=file
STATUS=status variable

Parameters

LIST or *L*

Default listing file for the LIST parameter used on subsequent subcommands in an SCU session. You can specify a file position as part of the file name. If LIST is omitted, file $LIST is used.

ERRORS or *E*

Name of the file on which intermediate error messages are written. If ERRORS is omitted, file $ERRORS is used.

Remarks

- This subcommand specifies the default value for the LIST parameter on SCU subcommands. A file specified for a LIST parameter overrides this value.
- The functions $ERRORS_FILE and $LIST_FILE return the values specified for these files.
- For more information, see the NOS/VE Source Code Management manual.

Examples

The following subcommand causes file SCU_LIST to be used as the default value for the LIST parameter on subsequent subcommands. Intermediate error messages are written on file SCU_ERRORS.

```
sc/set_list_options l=scu_list e=scu_errors
```

USE_LIBRARY SCU Subcommand

Purpose

Specifies the base and result libraries for an SCU utility session. This subcommand also specifies where the QUIT, END_LIBRARY, and WRITE_LIBRARY subcommands write their results.

Format

USE_LIBRARY or
USEL
BASE=file
RESULT=file
STATUS=status variable

Parameters

BASE or *B*

Name of the source library copied as the initial working library for the session. The files specified by the BASE and RESULT parameters can be the same. If BASE is omitted, file SOURCE_LIBRARY in your working catalog is used.

RESULT or *R*

Name of the file on which the new source library is written by subsequent END_LIBRARY, WRITE_LIBRARY, or QUIT subcommands. The new source library can be written when either a QUIT, END_LIBRARY, or WRITE_LIBRARY subcommand is entered. The WRITE_LIBRARY subcommand can specify a different source library than that specified by the USE_LIBRARY subcommand. The files specified by the BASE and RESULT parameters can be the same. If RESULT is omitted, the file specified by the BASE parameter is used.

Remarks

- All subcommands in the session affect the same working library. The working library is initially a duplicate of the base library specified on the BASE parameter.
- If no USE_LIBRARY or CREATE_LIBRARY subcommand is issued before other subcommands during an SCU session, file SOURCE_LIBRARY is used for the base and result libraries.
- You must have read permission on the base library. You must have read and write permission on the result library. If you only want to read the base library, specify $NULL as the result library.
- For more information, see the NOS/VE Source Code Management manual.

Examples

The following sequence begins an SCU session and initializes the working library from file FSEWORK in your working catalog, assumed not to be $LOCAL. In this example, source libraries are written on the next cycle of file FSEWORK by subsequent END_LIBRARY, WRITE_LIBRARY, or QUIT subcommands.

```
/source_code_utility
sc/use_library b=fsework r=fsework.$next
```

The following sequence specifies $NULL as the result library. You can use this example to look at a source library, but not to change it.

```
/source_code_utility
sc/use_library ..
sc../b=$system.cybil.osf$program_interface r=$null
```

WRITE_LIBRARY SCU Subcommand

Purpose Generates a result library from the current state of the working library. It writes the result library on the file specified by the RESULT parameter.

Format

WRITE_LIBRARY or
WRIL
RESULT=file
STATUS=status variable

Parameters

RESULT or *R*

File to which the result library is written. If RESULT is omitted, the file used is specified by the RESULT parameter of the CREATE_LIBRARY, previous WRITE_LIBRARY, or USE_LIBRARY subcommand. If RESULT is specified, that file name becomes the default for subsequent QUIT or WRITE_LIBRARY subcommands.

Remarks

- This subcommand allows you to generate more than one source library in an SCU session. This is done if you specify a file on the RESULT parameter. To create an empty library, refer to the CREATE_LIBRARY subcommand.

- The subcommand can save the contents of the working library at an intermediate state in case the system fails during the session. In this case, you can omit the RESULT parameter and use the result file you specified when you began the session. When you end the session, you can overwrite the intermediate library with the final result library.

- If the result file is the same as the file named on the BASE parameter of the USE_LIBRARY subcommand, the file is rewritten only if the working library has been modified.
- The END_LIBRARY and QUIT subcommands also generate a result library.
- Specifying RESULT changes the value of the $RESULT function to reflect the new file name.
- For more information, see the NOS/VE Source Code Management manual.

Examples The following subcommand writes an intermediate library to the result library file.

```
sc/write_library
```

Related Manuals A

This appendix lists the manuals which describe NOS/VE. Also included is information for ordering printed manuals and the way to access online manuals.

Related Manuals A

All NOS/VE manuals and related hardware manuals are listed in table A-1. If your site has installed the online manuals, you can find an abstract for each NOS/VE manual in the online System Information manual. To access this manual, enter:

```
/help manual=nos_ve
```

Ordering Printed Manuals

To order a printed Control Data manual, send an order form to:

Control Data
Literature and Distribution Services
308 North Dale Street
St. Paul, Minnesota 55103-2495

To obtain an order form or to get more information about ordering Control Data manuals, write to the above address or call (612) 292-2101. If you are a Control Data employee, call (612) 292-2100.

Accessing Online Manuals

To access the online version of a printed manual, log in to NOS/VE and enter the online title on the HELP command (table A-1 supplies the online titles). For example, to see the NOS/VE Commands and Functions manual, enter:

```
/help manual=scl
```

or, because SCL is the default for the MANUAL parameter, simply enter

```
/help
```

An online Examples manual contains examples that reside in printed manuals. From within the online Examples manual, you can copy, print, and execute the examples it contains. To access this manual, enter:

```
/help manual=examples
```

When EXAMPLES is listed in the Online Manuals column in table A-1, that manual is represented in the online Examples manual.

Table A-1. Related Manuals

Manual Title	Publication Number	Online Manuals[1]
NOS/VE Site Manuals:		
CYBER 930 Computer System Guide to Operations Usage	60469560	
CYBER Initialization Package (CIP) CYBER 180 Model 810, 830, 815, 825; CYBER 810A, 830A Computer Systems Reference Manual	60000417	
CYBER Initialization Package (CIP) CYBER 180 Model 835, 845, 855; CYBER 840, 850, 860 Computer Systems with IOU AB115A Reference Manual	60000418	
CYBER Initialization Package (CIP) CYBER 180 Model 845, 855; CYBER 840, 850, 860 with IOU AT478/AT481A; CYBER 840A, 850A, 860A, 870A, 990, 990E, 995E Computer Systems Reference Manual	60000419	
CYBER Initialization Package (CIP) CYBER 960, 994 Computer Systems Reference Manual	60000420	
CYBER Initialization Package (CIP) CYBER 962, 992 Computer Systems Reference Manual	60000421	

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

(Continued)

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
NOS/VE Site Manuals (Continued):		
CYBER Initialization Package (CIP) CYBER 170 Model 865, 875; Non-Model 8XX/9XX Series Computer Systems Reference Manual	60000422	
Desktop/VE Host Utilities Usage	60463918	
Mail/VE Version 2 Administration	60464515	MAILVE_ ADMINISTRA-TION
MAINTAIN_MAIL (Version 1)[2] Usage		MAIM
NOS/VE Accounting Analysis System Usage	60463923	
NOS/VE Accounting and Validation Utilities for Dual State Usage	60458910	
NOS/VE File Server for STORNET and ESM-II Usage	60000190	
NOS/VE LCN Configuration and Network Management Usage	60463917	

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

2. To access this manual, you must be the administrator for Mail/VE Version 1.

(Continued)

A

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
Site Manuals (Continued):		
NOS/VE Network Management Usage	60463916	
NOS/VE Operations Usage	60463914	
NOS/VE System Performance and Maintenance Volume 1: Performance Usage	60463915	
NOS/VE System Performance and Maintenance Volume 2: Maintenance Usage	60463925	
NOS/VE Security Administration Usage	60463945	
NOS/VE User Validation Usage	60464513	
NOS/VE User Manuals:		
EDIT_CATALOG Usage		EDIT_ CATALOG
EDIT_CATALOG for NOS/VE Summary	60487719	
Introduction to NOS/VE Tutorial	60464012	EXAMPLES

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

(Continued)

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
NOS/VE User Manuals (Continued):		
NOS/VE Advanced File Management Tutorial	60486412	AFM_T
NOS/VE Advanced File Management Usage	60486413	AFM
NOS/VE Commands and Functions Quick Reference	60464018	SCL
NOS/VE File Editor Tutorial/Usage	60464015	EXAMPLES
NOS/VE Object Code Management Usage	60464413	OCM and EXAMPLES
NOS/VE Screen Formatting Usage	60488813	EXAMPLES
NOS/VE Source Code Management Usage	60464313	SCM and EXAMPLES
NOS/VE System Usage	60464014	EXAMPLES
NOS/VE Terminal Definition Usage	60464016	

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

(Continued)

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
NOS/VE User Manuals (Continued):		
Screen Design Facility/Screen Formatting Usage	60488613	SDF
Screen Design Facility/Data Management Usage	60488618	
CYBIL Manuals:		
CYBIL for NOS/VE File Management Usage	60464114	EXAMPLES
CYBIL for NOS/VE Keyed-File and Sort/Merge Interfaces Usage	60464117	EXAMPLES
CYBIL for NOS/VE Language Definition Usage	60464113	CYBIL and EXAMPLES
CYBIL for NOS/VE Sequential and Byte-Addressable Files Usage	60464116	EXAMPLES
CYBIL for NOS/VE System Interface Usage	60464115	EXAMPLES

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

(Continued)

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
FORTRAN Manuals:		
FORTRAN Version 1 for NOS/VE Language Definition Usage	60485913	
FORTRAN Version 1 for NOS/VE Quick Reference		FORTRAN
FORTRAN Version 2 for NOS/VE Language Definition Usage	60487113	
FORTRAN Version 2 for NOS/VE Quick Reference		VFORTRAN
FORTRAN for NOS/VE Tutorial	60485912	FORTRAN_T
FORTRAN for NOS/VE Topics for FORTRAN Programmers Usage	60485916	
FORTRAN for NOS/VE Summary	60485919	
FORTRAN Keyed-File and Sort/Merge Interfaces Usage	60485917	FORTRAN
COBOL Manuals:		
COBOL for NOS/VE Summary	60486019	

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

(Continued)

A

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
COBOL Manuals (Continued):		
COBOL for NOS/VE Tutorial	60486012	COBOL_T
COBOL for NOS/VE Usage	60486013	COBOL and EXAMPLES
Other Compiler Manuals:		
Ada for NOS/VE Usage	60498113	ADA
Ada for NOS/VE Reference Manual	60498118	EXAMPLES
APL for NOS/VE File Utilities Usage	60485814	
APL for NOS/VE Language Definition Usage	60485813	
BASIC for NOS/VE Summary Card	60486319	
BASIC for NOS/VE Usage	60486313	BASIC
LISP for NOS/VE Usage Supplement	60486213	
Pascal for NOS/VE Summary Card	60485619	

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

(Continued)

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
Other Compiler Manuals (Continued):		
Pascal for NOS/VE Usage	60485618	PASCAL and EXAMPLES
Prolog for NOS/VE Usage	60486713	PROLOG
VX/VE Manuals:		
C/VE for NOS/VE Quick Reference		C
C/VE for NOS/VE Usage	60469830	
DWB/VX Introduction and User Reference Tutorial/Usage	60469890	
DWB/VX Macro Packages Guide Usage	60469910	
DWB/VX Preprocessors Guide Usage	60469920	
DWB/VX Text Formatters Guide Usage	60469900	

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

(Continued)

A

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
VX/VE Manuals (Continued):		
VX/VE Administrator Guide and Reference Tutorial/Usage	60469770	
VX/VE An Introduction for UNIX Users Tutorial/Usage	60469980	
VX/VE Programmer Guide Tutorial	60469790	
VX/VE Programmer Reference Usage	60469820	
VX/VE Support Tools Guide Tutorial	60469800	
VX/VE User Guide Tutorial	60469780	
VX/VE User Reference Usage	60469810	

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

(Continued)

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
Data Management Manuals:		
DM Command Procedures Reference Manual	60487905	
DM Concepts and Facilities Manual	60487900	
DM Error Message Summary for DM on CDC NOS/VE	60487906	
DM Fundamental Query and Manipulation Manual	60487903	
DM Report Writer Reference Manual	60487904	
DM System Administrator's Reference Manual for DM on CDC NOS/VE	60487902	
DM Utilities Reference Manual for DM on CDC NOS/VE	60487901	
IM/DM for NOS/VE Installation and User Guide	60487907	

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

(Continued)

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
Information Management Manuals:		
IM/Control for NOS/VE Quick Reference	L60488918	CONTROL
IM/Control for NOS/VE Usage	60488913	
IM/Fast for NOS/VE Administration Usage	60487513	
IM/Fast for NOS/VE Programming Usage	60487514	
IM/Quick for NOS/VE Tutorial	60485712	
IM/Quick for NOS/VE Summary	60485714	
IM/Quick for NOS/VE Online Help		QUICK
IM/Smart for NOS/VE Usage	60488513	

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

(Continued)

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
CDCNET Manuals:		
CDCNET Access Guide		CDCNET_ ACCESS
CDCNET Batch Device User Guide	60463863	CDCNET_ BATCH
CDCNET Commands Reference	60000414	
CDCNET Conceptual Overview	60461540	
CDCNET Configuration Guide	60461550	
CDCNET DI Dump Analyzer		ANACD
CDCNET Diagnostic Messages	60461600	CDCNET_ MSGS
CDCNET Network Configuration Utility		NETCU
CDCNET Network Operations and Analysis	60461520	
CDCNET Product Descriptions	60460590	
CDCNET Terminal Interface	60463850	
CDCNET TCP/IP Applications	60000214	

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

(Continued)

A

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
Migration Manuals:		
Migration from IBM to NOS/VE Tutorial/Usage	60489507	
Migration from NOS to NOS/VE Tutorial/Usage	60489503	
Migration from NOS/BE to NOS/VE Tutorial/Usage	60489505	
Migration from NOS/BE to NOS/VE Standalone Tutorial/Usage	60489506	
Migration from VAX/VMS to NOS/VE Tutorial/Usage	60489508	
Miscellaneous Manuals:		
ANALYZE_DUMP Utility		ANALYZE_DUMP
Applications Directory	60455370	
Control Data CONNECT User's Guide	60462560	
Control Data CONNECT Plus for the IBM Personal Computer (Version 1.0) User's Guide	60000388	
Control Data CONNECT VIEW for the IBM Personal Computer Version 2.0 User's Guide	60463946	

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

(Continued)

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
Miscellaneous Manuals (Continued):		
Debug for NOS/VE Quick Reference	60488218	DEBUG
Debug for NOS/VE Usage	60488213	
Desktop/VE for Macintosh Usage	60464503	
FTAM/VE Usage	60000455	
Mail/VE (Version 1) Summary Card	60464519	
Mail/VE (Version 1) Usage		MAIL_VE
Mail/VE Version 2 Usage	60464514	MAILVE_V2
Math Library for NOS/VE Usage	60486513	
NOS/VE Build Utility Usage	60487413	
NOS/VE Diagnostic Messages Usage	60464613	MESSAGES
NOS/VE Examples Usage		EXAMPLES

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

(Continued)

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
Miscellaneous Manuals (Continued):		
NOS/VE Global Index Reference	60464010	
NOS/VE Online Manual Systems	60488403	TOPICS_ CONTEXT
NOS/VE System Information		NOS_VE
Programming Environment for NOS/VE Usage		ENVIRON-MENT
Programming Environment for NOS/VE Summary	60486819	
Professional Programming Environment for NOS/VE Quick Reference	60486618	PPE
Professional Programming Environment for NOS/VE Usage	60486613	
Remote Host Facility Usage	60460620	

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

(Continued)

Table A-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Manuals[1]
Hardware Manuals:		
CYBER 170 Computer Systems Models 825, 835, and 855 General Description Hardware Reference	60459960	
HPA/VE Reference	60461930	
Virtual State Volume II Hardware Reference	60458890	

1. This column lists the title of the online version of the manual and indicates whether the examples in the printed manual are in the online Examples manual.

Date/Time Formats B

Date/Time Formats B

This appendix describes date/time string values and the form strings used to interpret them.

Date/Time Form Strings

The following functions allow you to specify the format in which the value returned by the function is presented:

$DATE
$DATE_TIME
$DATE_TIME_STRING
$DAY
$TIME
$TIME_ZONE_IDENTIFIER or $TIME_ZONE_ID

In these cases, the format of the value is determined by a *form string*. A form string is a string containing keyword values corresponding to the elements of a date/time value. These keywords specify how each of the date/time elements is presented.

Table B-1 lists and describes the keyword elements you can use to specify form strings for the functions listed above. Tables B-2 and B-3 list and describe both the date/time strings provided by NOS/VE and form strings used to represent them.

When specifying form strings, you can use the following delimiters to separate keywords:

period (.)
comma (,)
slant (/)
hyphen (-)
space ( )

Table B-1. Form String Elements

Keyword	Description
Y2	A two-digit year (00..99)
Y4	A four-digit year (1900..2155)
M2	A two-digit month (01..12)
MN(language)[1]	The name of the month in the specified language.
MA(language)[1]	The abbreviated name of the month in the specified language.
D2	A two-digit day of the month (01..31)
J3	A three-digit day of the year (001..366)
DN(language)[1]	The name of the day in the specified language.
DA(language)[1]	The abbreviated name of the day in the specified language.
MONTH	The equivalent of the form string 'MN D2, Y4'
MDY	The equivalent of the form string 'M2/D2/Y2'
DMY	The equivalent of the form string 'D2.M2.Y2'
ISOD	The equivalent of the form string 'Y4-M2-D2'
ORDINAL	The equivalent of the form string 'Y4J3'
H24	A two-digit hour using the 24-hour clock (00..23)
H12	A two-digit hour using the 12-hour clock (1..12)

(Continued)

Table B-1. Form String Elements *(Continued)*

Keyword	Description
AMORPM	A two-character day or evening indicator (AM or PM)
MM	A two-digit minute (00..59)
SS	A two-digit second (00..59)
S10	A one-digit tenth of a second (0..9)
S100	A two-digit hundredth of a second (00..99)
S1000	A three-digit thousandth of a second (000..999)
AMPM	The equivalent of the form string 'H12:MM AMORPM'
HMS	The equivalent of the form string 'H24:MM:SS'
MILLISECOND, MS	The equivalent of the form string 'H24:MM:SS.S1000'
ISOT	The equivalent of the form string 'H24:MM:SS,S100'
TZ(language)[1]	The identifier of the system's time zone in the specified language.
TZA(language)[1]	The abbreviated identifier of the system's time zone in the specified language.

1. If the language portion of the keyword is not specified, the currently selected natural language is used. Message modules can be defined to provide month and day names and time zone identifiers in languages other than English. For more information on creating message modules, see the Object Code Management manual.

The following considerations apply to the use of form strings:

- When a date is converted to its string representation, leading zeros are omitted from the day value in any format using MN or MA followed by a space or a hyphen.
- Multiple spaces in form strings are treated as a single space and spaces on either side of a comma are ignored.
- If a format string is omitted from the call to the function, the specified date is interpreted according to the default format chosen by your site from those listed in the next section.

Date/Time Strings

You can specify a date/time value in a number of different formats, provided that the value is specified as a string and that its contents match a recognized date/time format. Tables B-2 and B-3 list and describe the different date and time formats supplied by NOS/VE.

The following considerations apply to strings supplied as date/time values.

- You can specify a string containing individual date or time components or both. If you specify both, you must separate them from each other using one of the following delimiters:

  period (.)
  comma (,)
  slant (/)
  hyphen (-)
  space ( )

- You can omit leading zeros within a date/time string if they are the first character in the string or if they are preceded by a letter or a delimiter such as a colon.

- Multiple spaces are treated as a single space and spaces on either side of a comma are ignored.

Table B-2. Date Strings

Form	Example	Description
'MN D2, Y4' or 'MONTH'	'November 1, 1988'	The name of a month, a two-digit day, a comma, and a four-digit year.
'MA D2, Y4'	'Nov 1, 1988'	The abbreviated name of a month, a two-digit day, a comma, and a four-digit year.
'D2 MN Y4'	'1 November 1988'	A two-digit day, the name of a month, and a four-digit year separated by spaces.
'D2 MA Y4'	'1 Nov 1988'	A two-digit day, the abbreviated name of a month, and a four-digit year separated by spaces.
'D2 MN Y2'	'1 November 88'	A two-digit day, the name of a month, and a two-digit year separated by spaces.
'D2 MA Y2'	'1 Nov 88'	A two-digit day, the abbreviated name of a month, and a two-digit year separated by spaces.
'D2-MN-Y4'	'1-November-1988'	A two-digit day, the name of a month, and a four-digit year separated by hyphens.
'D2-MA-Y4'	'1-Nov-1988'	A two-digit day, the abbreviated name of a month, and a four-digit year separated by hyphens.
'D2-MN-Y2'	'1-November-88'	A two-digit day, the name of a month, and a two-digit year separated by hyphens.

(Continued)

Table B-2. Date Strings *(Continued)*

Form	Example	Description
'D2-MA-Y2'	'1-Nov-88'	A two-digit day, the abbreviated name of a month, and a two-digit year separated by hyphens.
'D2MNY4'	'01November1988'	A two-digit day, the name of a month, and a four-digit year.
'D2MAY4'	'01Nov1988'	A two-digit day, the abbreviated name of a month, and a four-digit year.
'D2MNY2'	'01November88'	A two-digit day, the name of a month, and a two-digit year.
'D2MAY2'	'01Nov88'	A two-digit day, the abbreviated name of a month, and a two-digit year.
'M2/D2/Y4'	'11/01/1988'	A two-digit month, a two-digit day, and a four-digit year separated by slants.
'M2/D2/Y2' or 'DMY'	'11/01/88'	A two-digit month, a two-digit day, and a two-digit year separated by slants.
'Y4-J3'	'1988-306'	A four-digit year and a three-digit day separated by a hyphen.
'Y4J3' or 'ORDINAL'	'1988306'	A four-digit year and a three-digit day.

(Continued)

Table B-2. Date Strings *(Continued)*

Form	Example	Description
'Y2-J3'	'88-306'	A two-digit year and a three-digit day separated by a hyphen.
'Y2J3'	'88306'	A two-digit year and a three-digit day.
'Y4-M2-D2' or 'ISOD'	'1988-11-01'	A four-digit year, a two-digit month, and a two-digit day separated by hyphens.
'Y4M2D2'	'19881101'	A four-digit year, a two-digit month, and a two-digit day.
'Y2-M2-D2'	'88-11-01'	A two-digit year, a two-digit month, and a two-digit day separated by hyphens.
'Y2M2D2'	'881101'	A two-digit year, a two-digit month, and a two-digit day.
'D2.M2.Y4'	'01.11.1988'	A two-digit day, a two-digit month, and a four-digit year separated by periods.
'D2.M2.Y2' or 'DMY'	'01.11.88'	A two-digit day, a two-digit month, and a two-digit year separated by periods.

Table B-3. Time Strings

Form	Example	Description
'H12:MM AMORPM' or 'AMPM'	'2:41 PM'	A two-digit hour (12-hour clock), a colon, a two-digit minute, a space, and a two-character day or evening indicator.
'H24:MM:SS' or 'HMS'	'14:41:38'	A two-digit hour (24-hour clock), a two-digit minute, and a two-digit second separated by colons.
'H24:MM:SS.S1000' or 'MILLISECOND', 'MS'	'14:41:38.629'	A two-digit hour (24-hour clock), a two-digit minute, and a two-digit second separated by colons followed by a period and a three-digit thousandth of a second.
'H24:MM:SS,S100' or ISOT	'14:41:38,62'	A two-digit hour (24-hour clock), a two-digit minute, and a two-digit second separated by colons followed by a comma and a two-digit hundredth of a second.

Replacements for Old NOS/VE Commands and Functions C

Replacements for Old NOS/VE Commands and Functions C

The following table lists old commands from previous versions of NOS/VE and the preferred command or replacement command. Some of the preferred commands may have parameters that differ from an old command. Commands listed more than once were replaced by more than one command.

Old Command	Preferred/Replacement Command
ACCEPT_LINE	GET_LINE
ADMINISTER_USER	ADMINISTER_VALIDATIONS
CHANGE_TERM_CONN_ ATTRIBUTES	CHANGE_CONNECTION_ ATTRIBUTES
CONVERT_SCU10_TO_SCU11	Obsolete
CREATE_VARIABLE	VAR/VAREND
DISPLAY_170_REQUEST	DISPLAY_TAPE_LABEL_ ATTRIBUTES
DISPLAY_7600_REQUEST	DISPLAY_TAPE_LABEL_ ATTRIBUTES
DISPLAY_COMMAND_ PARAMETERS	DISPLAY_COMMAND_ INFORMATION
DISPLAY_IBM_REQUEST	DISPLAY_TAPE_LABEL_ ATTRIBUTES
DISPLAY_JOB_STATUS	DISPLAY_INPUT_STATUS
DISPLAY_PRINT_STATUS	DISPLAY_OUTPUT_STATUS
DISPLAY_TERM_CONN_ ATTRIBUTES	DISPLAY_CONNECTION_ ATTRIBUTES
DISPLAY_VAX_REQUEST	DISPLAY_TAPE_LABEL_ ATTRIBUTES
EDIT_LIBRARY (SCU subcommand)	EDIT_DECK (SCU Subcommand)

Old Command	Preferred/Replacement Command
EXIT_PROC	EXIT
FORMAT_SCL_PROC	FORMAT_SCL_PROCEDURE
REQUEST_OPERATOR_ ACTION	SEND_OPERATOR_MESSAGE
SET_COMMAND_LIST	CREATE_COMMAND_LIST_ ENTRY
SET_COMMAND_LIST	DELETE_COMMAND_LIST_ ENTRY
SET_COMMAND_LIST	CHANGE_COMMAND_SEARCH_ MODE
SET_COMMAND_MODE	CHANGE_INTERACTION_STYLE
SET_COMMAND_MODE	CHANGE_SCL_OPTIONS
SET_JOB_LIMIT	CHANGE_JOB_LIMIT
SET_LINK_ATTRIBUTES	CHANGE_LINK_ATTRIBUTES
SET_MESSAGE_MODE	CHANGE_MESSAGE_LEVEL
SET_PASSWORD	CHANGE_LOGIN_PASSWORD
SET_TERMINAL_ATTRIBUTES	CHANGE_TERMINAL_ ATTRIBUTES
SET_WORKING_CATALOG	CHANGE_WORKING_CATALOG
TERMINATE_JOB	TERMINATE_INPUT
TERMINATE_PRINT	TERMINATE_OUTPUT
TRANSFER_FILE_XMODEM	XMODEM_SEND
TRANSFER_FILE_XMODEM	XMODEM_RECEIVE

The following table lists old functions from previous versions of NOS/VE and the preferred function or replacement function. Some of the preferred functions may have parameters that differ from an old function. Functions listed more than once were replaced by more than one function.

Old Function	Preferred/Replacement Function
$CATALOG	$WORKING_CATALOG
$COMMAND_SOURCE	$SOURCE
$CONDITION_CODE(a)	$STATUS_CODE
$CONDITION_CODE(a,numeric)	$STATUS_CODE
$CONDITION_CODE(a,symbolic)	$STATUS_CODE_STRING
$CONDITION_NAME	$STATUS_CODE_NAME
$DECK_HEADER	$DECK_ATTRIBUTES
$DECK_LIST	$DECK_NAME_LIST
$FEATURE_LIST	$FEATURE_NAME_LIST
$FEATURE_MEMBERS	$FEATURE_MEMBER_NAMES
$FIRST_DECK	$FIRST_DECK_NAME
$FIRST_MODIFICATION	$FIRST_MODIFICATION_NAME
$GROUP_LIST	$GROUP_NAME_LIST
$GROUP_MEMBERS	$GROUP_MEMBER_NAMES
$LAST_DECK	$LAST_DECK_NAME
$LAST_MODIFICATION	$LAST_MODIFICATION_NAME
$LIBRARY_HEADER	$LIBRARY_ATTRIBUTES
$MAX_VALUES	$MAX_LIST
$MAX_VALUE_SETS	$MAX_LIST

Old Function	Preferred/Replacement Function
$MODIFICATION_HEADER	$MODIFICATION_ATTRIBUTES
$MODIFICATION_LIST	$MODIFICATION_NAME_LIST
$MODIFIED_DECKS	$MODIFIED_DECK_NAMES
$NEXT_DECK	$NEXT_DECK_NAME
$NEXT_MODIFICATION	$NEXT_MODIFICATION_NAME
$PARAMETER	Obsolete
$PARAMETER_LIST	Obsolete
$RANGE	$RANGE_SPECIFIED
$SET_COUNT	$SIZE
$SEVERITY	$STATUS_SEVERITY
$STRLEN	$SIZE
$SUBSTR	$SUBSTRING
$VALUE	$PARAMETER_VALUE
$VALUE_COUNT	$SIZE
$VALUE_KIND	$GENERIC_TYPE

Comments (continued from other side)

Please fold on dotted line;
seal edges with tape only.

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
First-Class Mail Permit No. 8241 Minneapolis, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CONTROL DATA
Technical Publications
ARH219
4201 N. Lexington Avenue
Arden Hills, MN 55126-9983

NOS/VE Commands and Functions 60464018 J

We would like your comments on this manual to help us improve it. Please take a few minutes to fill out this form.

Who are you?

- ☐ Manager
- ☐ Systems analyst or programmer
- ☐ Applications programmer
- ☐ Operator
- ☐ Other _____

How do you use this manual?

- ☐ As an overview
- ☐ To learn the product or system
- ☐ For comprehensive reference
- ☐ For quick look-up
- ☐ Other _____

What programming languages do you use? _____

_____

How do you like this manual? Answer the questions that apply.

Yes	Somewhat	No	
☐	☐	☐	Does it tell you what you need to know about the topic?
☐	☐	☐	Is the technical information accurate?
☐	☐	☐	Is it easy to understand?
☐	☐	☐	Is the order of topics logical?
☐	☐	☐	Can you easily find what you want?
☐	☐	☐	Are there enough examples?
☐	☐	☐	Are the examples helpful? (☐ Too simple? ☐ Too complex?)
☐	☐	☐	Do the illustrations help you?
☐	☐	☐	Is the manual easy to read (print size, page layout, and so on)?
☐	☐	☐	Do you use this manual frequently?

Comments? If applicable, note page and paragraph. Use other side if needed.

Check here if you want a reply: ☐

Name _____ Company _____

Address _____ Date _____

_____ Phone _____

Please send program listing and output if applicable to your comment.

CONTROL DATA