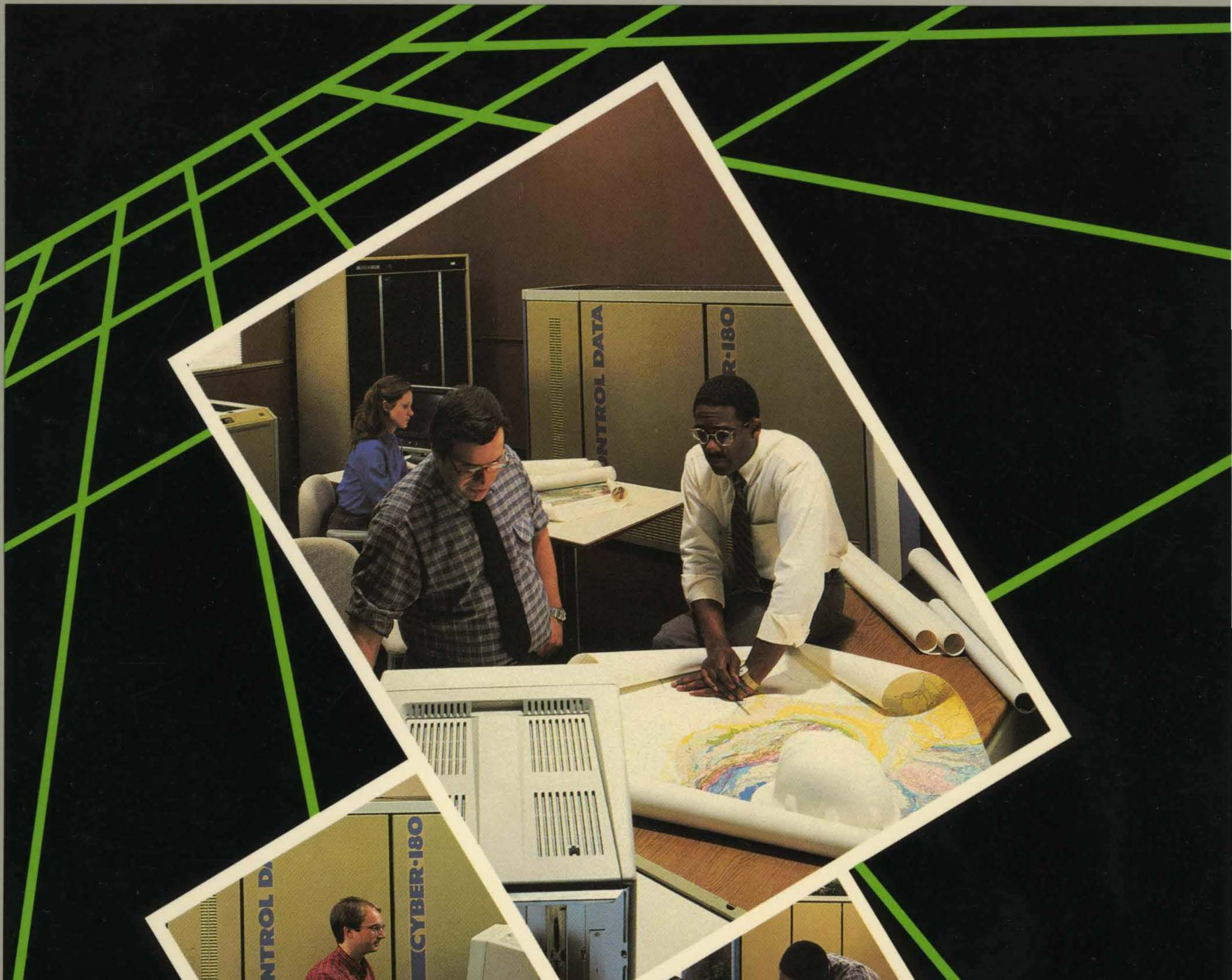
NOS/VE System Performance and Maintenance

Volume 2: Maintenance

CONTROL DATA



Usage

60463925

NOS/VE System Performance and Maintenance Volume 2: Maintenance

Usage

Publication Number 60463925

Manual History

Revision	System Version	PSR Level	Date
A	1.1.1	613	July 1984
B	1.1.3	644	December 1985
C	1.2.1	664	September 1986
D	1.2.1	670	December 1986
E	1.2.2	678	April 1987
F	1.2.3	688	September 1987
G	1.3.1	700	April 1988
H	1.4.1	716	December 1988
J	1.5.1	739	December 1989

This revision reflects the following major additions and changes:

- Support of the CYBER Cartridge Tape System.
- Application support of the CYBER Cartridge Tape System.
- Support of dual I4C IOU.
- Support of the cataloged tape files feature.
- Support of the new DISPLAY_TERMINAL_MODEL SCL command.
- Inclusion of the Logical Configuration Utility (LCU) within the System Operator Utility.
- Support of the large memory dump support feature.
- Changes to LCU network configuration subcommands.
- Support of the DEFINE_INITIAL_APPLICATION SCL command.
- Extensive changes and enhancements to the system core debugger.
- Support of the new SET_SECURITY_OPTION system core command and the $SECURITY_OPTION SCL function.
- Support of the new MANAGE_FIELD_CHANGES utility.

This edition obsoletes all previous editions.

Contents

Figures

Tables

About This Manual

This manual contains system information and information about software tools used by a site analyst to support, maintain, and troubleshoot operations of the CONTROL DATA® Network Operating System/Virtual Environment (NOS/VE). NOS/VE executes as a standalone system, or in a dual-state configuration with either the CDC® Network Operating System (NOS) Version 2 or the CDC Network Operating System Batch Environment (NOS/BE) Version 1.5.

Audience

This manual is written for the site analyst responsible for maintenance, support, and problem troubleshooting at a site using NOS/VE. (On dual-state systems, the reader is assumed to be an experienced NOS or NOS/BE analyst.) It is assumed that the reader has a working knowledge of the System Command Language (SCL).

Conventions

The following conventions are used in this manual:

Boldface	In a command or function format description, names and required parameters are shown in boldface type.
Italics	In a command or function format description, optional parameters are shown in italic type.
Numbers	All numbers are shown in decimal format unless otherwise noted.
Vertical bar	A technical change is indicated by a vertical bar in the margin next to the change.
Shading	In examples of interactive terminal sessions, user input is shaded to distinguish it from system output.

Submitting Comments

There is a comment sheet at the back of this manual. You can use it to give us your opinion of the manual's usability, to suggest specific improvements, and to report errors. Mail your comments to:

Control Data
Technical Publications ARH219
4201 North Lexington Avenue
St. Paul, Minnesota 55126-6198

Please indicate whether you would like a response.

If you have access to SOLVER, the Control Data online facility for reporting problems, you can use it to submit comments about the manual. When entering your comments, use NV0 (zero) as the product identifier. Include the name and publication number of the manual.

If you have questions about the packaging and/or distribution of a printed manual, write to:

Control Data
Literature and Distribution Services
308 North Dale Street
St. Paul, Minnesota 55103-2495

or call (612) 292-2101. If you are a Control Data employee, call (612) 292-2100.

CYBER Software Support Hotline

Control Data's CYBER Software Support maintains a hotline to assist you if you have trouble using our products. If you need help not provided in the documentation, or find the product does not perform as described, call us at one of the following numbers. A support analyst will work with you.

From the USA and Canada: (800) 345-9903

From other countries: (612) 851-4131

Overview 1

Overview 1

This chapter provides an overview of both volumes of the NOS/VE System Performance and Maintenance manual, gives a general description of the CYBER 180 hardware, and supplies an annotated list of important site analyst documentation.

Introduction to the System Performance and Maintenance Manuals

The System Performance and Maintenance manuals comprise a two-volume set. In general, the purpose of the set is to provide analyst-level configuration and maintenance information for the NOS/VE operating system. This includes information about how to configure and tune the operating system itself. It also includes information about various features and utilities closely related to operating system usage such as the Statistics Facility, the Physical Configuration Utility, the system core debugger, permanent file maintenance, and deadstart file maintenance.

Volume 1: Performance

Volume 1 contains information about:

- CPU- and memory-related performance tuning methods.
- Job scheduler.
- Statistics Facility.

Chapter 1: Overview

Chapter 2: Adjusting System Attributes

System attributes are system variables used to define performance parameters and to select certain system options. Each of the system attributes is described in this chapter. This chapter also describes the SET_SYSTEM_ATTRIBUTE system core command, which is used to control the system attributes.

Chapter 3: Managing Memory

Chapter 3 describes the MANAGE_MEMORY utility. This utility controls the attributes used to configure NOS/VE memory.

Chapter 4: Job Scheduling and Load Leveling

Chapter 4 describes the two utilities used to control job scheduling activities in NOS/VE: the ADMINISTER_SCHEDULING utility and the MANAGE_ACTIVE_SCHEDULING utility.

For sites that have multiple mainframes connected by NOS/VE File Server software and STORNET hardware, chapter 4 also describes how to configure a tightly coupled network for automatic load leveling. Load leveling refers to the dynamic routing of batch jobs between mainframes for the purpose of maintaining all mainframes in the tightly coupled network at their maximum load.

Chapters 5, 6, and 7: Special Topics in Performance Tuning

These chapters provide a more detailed description of the processing algorithms used in three major areas of system tuning: CPU scheduling, page aging, and job swapping. Included in each discussion are explanations of the individual system attributes and job scheduling attributes that influence processing performance.

Chapter 8: Performance Optimization

Chapter 8 contains a number of suggestions about how to achieve optimum system performance. Most of the chapter is devoted to the problem of how to balance system performance in three major areas: the CPU, memory, and disk I/O. Achieving a good balance between these three system resources minimizes the formation of bottlenecks that can impede system performance.

Chapter 9: Statistics Facility

The Statistics Facility is a collection of commands and utilities that can be used to obtain statistical reporting on virtually any area of system activity. This chapter explains how to maintain the various binary logs to which statistical information is written and how to obtain readable reports on information recorded in the logs. The chapter also contains format descriptions for the statistics that are predefined by NOS/VE.

Volume 2: Maintenance

Volume 2 contains information about:

- Maintaining the configuration of hardware storage and network devices attached to NOS/VE.
- Deadstart file maintenance.
- Permanent file maintenance.
- Configuring mass storage classes for optimum fault tolerance.
- Recovering from hardware error conditions.

Chapter 1: Overview

Chapters 2 and 3: Peripheral Device Configuration and Management

Chapters 2 and 3 describe the two utilities used to manage peripheral hardware devices attached to NOS/VE. These are the Physical Configuration Utility (PCU) and the Logical Configuration Utility (LCU). Devices controlled by these utilities include disk and tape storage devices, network connection devices, and extended memory devices (STORNET and ESM-II). Devices such as printers and card readers are not included; these are considered terminals rather than peripheral devices.

Appendix E, Supported Hardware Products, should be used in conjunction with chapters 2 and 3. Appendix E contains reference information about the various hardware devices that can be configured to NOS/VE.

Chapter 4: System Core Commands

Chapter 4 describes the system core commands. The set of system core commands includes the INITIALIZE_SYSTEM_DEVICE command, used to initiate an installation deadstart, and the SET_SYSTEM_ATTRIBUTE command, used to change the current value of system attributes.

Chapter 5: Deadstart File Maintenance

This chapter provides information about how to create and install a deadstart file. The information in this chapter can be divided into three types of information: a description of the files and catalogs related to deadstart, reference descriptions of commands related to deadstart, and step-by-step descriptions of the processes used to create a deadstart tape and a deadstart disk file.

Chapter 6: Customizing the Operational Environment

The Site Tailoring chapter describes several aspects of NOS/VE operations that can be customized to fit the needs of a particular site. Examples include information about how to customize the various prologs and epilogs that execute during deadstart, information about how to modify the set of terminal definitions maintained by NOS/VE, and information about how to modify certain system modules.

Chapter 7: Software Installation Utilities

Chapter 7 describes the two utilities used to install software products and batch corrective updates (BCUs). The INSTALL_SOFTWARE utility is used to actually install software products and BCUs. The CREATE_INSTALLATION_ENVIRONMENT utility controls a number of permits and processes related to the task of installing system software.

Chapter 8: Permanent File Maintenance

The information in this chapter is divided into two parts: how to maintain families on NOS/VE, and a description of the analyst-level parts of the utilities used to back up and restore files. For information about the operational aspects of the backup and restore utilities, refer to the NOS/VE Operations manual.

Chapters 9 and 10: Debug and Dump Utilities

The system core debugger described in chapter 9 can be used to debug one or more active tasks in the system. Chapter 10 provides information about the ANALYZE_DUMP utility.

Chapters 11, 12, and 13: Fault Tolerance Configuration and Failure Analysis

The information in these chapters is provided for two purposes:

- To provide information about how to configure storage devices to provide optimum fault tolerance characteristics.
- To provide information about recovering from hardware or software errors as quickly as possible.

All three chapters should be read in their entirety at the time NOS/VE is installed.

CYBER 180 Hardware

The mainframes on which NOS/VE executes provide central memory of up to 256 million bytes, virtual memory management, and 12- or 16-bit peripheral processor (PP) instructions in 16-bit PPs with 12- or 16-bit input/output (I/O) channels.

Some hardware is capable of either standalone or dual-state operations, whereas other hardware (such as a CYBER 930) is capable of only standalone operations. In a standalone system, NOS/VE operates completely independently; no other operating system is executing on the same hardware, and no other is required. In a dual-state system, two independent software systems execute simultaneously on the same hardware. When a task is loaded into the central processing unit (CPU), a flag in the exchange package indicates whether the NOS instruction set or the NOS/VE instruction set is to be executed.

In a dual-state system, the operating systems depend upon one another for services and help. Unit record physical input and output may be done under the control of NOS or NOS/BE. Terminal communications may or may not be controlled by NOS/VE. NOS/VE does all scheduling of tasks to the central processor (including the task that is the NOS or NOS/BE operating system), performs virtual memory management, and monitors and interprets hardware interrupt signals.

In a dual-state system, hardware interrupts cause the NOS/VE monitor to regain control, regardless of which operating system is executing when the interrupt occurs. On the basis of the NOS/VE monitor's interpretation of the interrupt, the monitor may terminate NOS/VE operations and revert to standalone NOS or NOS/BE operations, or, for handling of the interrupt, it may pass control to either NOS (or NOS/BE) or to NOS/VE.

Refer to the Virtual State Hardware Reference manual, Volume II, for a detailed discussion of the theory of hardware operation and for a discussion of how the software systems interact with the hardware.

Site Analyst Documentation Overview

This section lists the other manuals that are of greatest interest to the site analyst. The major topic areas for each manual are also described. See appendix B, Related Manuals, for information about how to order these and other NOS/VE manuals.

NOS/VE Software Release Bulletin (SRB)

The Software Release Bulletin is sent out with each new version of NOS/VE. The SRB describes new features of interest to a site analyst and contains up-to-the-minute information that could not be included in the NOS/VE manuals.

NOS/VE Installation Handbook (SMD132488)

Contains information on how to install the NOS/VE operating system. Installation instructions are included for an initial installation, an upgrade installation, and for product installations. The installation instructions apply to NOS/VE in a standalone environment, as well as to dual-state systems with NOS or NOS/BE.

Site Analyst Examples manual

Online manual that contains examples illustrating installation, configuration, permanent file maintenance, and network management. You can copy and then execute examples in this manual. All user names specified on the SETUP_INSTALLATION_PROCESS command described in the SRB are permitted to access this online manual. To access this manual, enter the following command from your configuration terminal:

```
   help manual=site_analyst_examples
```

NOS/VE Operations manual (60463914)

Describes the operator commands and how to interact with the system through the system console or through an interactive terminal used as a remote console. Other sections of this manual document magnetic tape usage, sending messages to interactive users, and system step/resumption.

NOS/VE User Validation manual (60464513)

Contains the information needed to validate NOS/VE users. It also contains information for host (NOS or NOS/BE) validation of dual-state users.

CYBER Initialization Package (CIP) User's Reference manual

Documents installation and use of the CYBER Initialization Package (CIP) components. These components are microcode, console drivers, and boot programs for the deadstart of NOS/VE, and utilities for dumping the mainframe's memories to magnetic tape.

There are six model-dependent versions of the CIP User's Reference manual:

Publication Number	CYBER Model Numbers
60000417	CYBER 180 Models 810, 830, 815, 825; CYBER 810A, 830A
60000418	CYBER 180 Models 835, 845, 855; CYBER 180 Models 840, 850, 860 with IOU AB115A
60000419	CYBER 180 Models 845, 855; CYBER 840, 850, 860 with IOU AT478/AT481A; CYBER 840A, 850A, 860A, 870A, 990, 990E, 995E
60000420	CYBER 960, 994
60000421	CYBER 962, 992
60000422	CYBER 170 Model 865, 875; Non-Model 8XX/9XX Series

NOS/VE Terminal Definition manual (60464016)

Lists the commands used to define a terminal's capabilities for full screen usage and documents the process of creating or updating terminal definitions.

NOS/VE Network Management manual (60463916)

Describes the MANAGE_NETWORK_APPLICATIONS utility, which is used to define and control network applications.

NOS/VE LCN Configuration and Network Management manual (60463917)

Describes how to use the MANAGE_RHFAM_NETWORK utility to manage the network configuration and network application definitions on NOS/VE systems supported by a loosely coupled network (LCN).

NOS/VE File Server for STORNET and ESM-II (60000190)

Describes how to configure mainframes in a tightly coupled network using the MANAGE_FILE_SERVER utility.

NOS/VE Security Administration manual (60463945)

Describes the system security features of the NOS/VE operating system.

NOS/VE Accounting Analysis System manual (60463923)

Written for the person responsible for customer accounting, this manual documents the Accounting Analysis System and how to create and maintain the pricing information used to bill customers.

NOS/VE Accounting and Validation Utilities for Dual State manual (60458910)

Describes the set of system utilities used to perform resource accounting and user validation functions on a NOS dual-state system.

NOS/VE Diagnostic Messages manual (60464613)

Lists formatted messages that describe the status of system requests, together with recommended user action. You can also access this manual online by entering HELP or EXPLAIN_MESSAGE after receiving a a diagnostic message, or by entering:

```
help manual=messages
```

NOS/VE File Archiving manual (60463944)

Describes the utilities used to archive files to secondary, low-cost storage and then release the disk space used by those files.

Desktop/VE Host Utilities manual (60463918)

Describes the installation and configuration requirements for the Desktop/VE interface to Apple Macintosh[1] microcomputers.

CDCNET Network Operations and Analysis manual (60461520)

Describes the Network Operator Utility which is used to monitor, control, and dynamically reconfigure CDCNET.

CDCNET Configuration Guide (60461550)

Provides the information you need to configure the network software for CDCNET.

CDCNET Batch Device User Guide (60463863)

Provides the information you need to use batch devices for CDCNET.

1. The Apple Macintosh is a product of Apple Computer, Inc. Apple is a registered trademark of Apple Computer, Inc. Macintosh is a registered trademark licensed to Apple Computer, Inc.

NOS 2 Installation Handbook (60459320)

Documents the installation and support of NOS and CYBER 170 portions of NOS dual-state systems. Includes instructions for rebuilding and applying BCUs for the 170 code used in dual-state system operation.

HPA/VE Reference manual (60461930)

Describes how to use the Hardware Performance Analyzer for NOS/VE. HPA/VE is a Control Data maintenance tool that processes the performance data of each active hardware element, alerts you when a hardware element is not operating normally, and provides information that will help you in repairing those elements.

NOS/VE-NOS/BE Dual State Preparation Guide (SMD132178)

Describes how to install a NOS/BE dual-state system. This guide is sent out to all customers who need to install a new NOS/BE dual-state system.

Part I: Peripheral Configuration

Physical Configuration Utility 2

Physical Configuration Utility 2

To control the transfer of data between a mainframe and its attached storage units, NOS/VE must have an exact description of each storage unit. NOS/VE must also know the hierarchical ordering of storage unit connections; that is, which controllers are attached to which data channels, which storage units are attached to which controllers, and so on. You provide this information to the NOS/VE system by means of the Physical Configuration Utility (PCU).

The term physical configuration refers to a complete description of a mainframe's attached storage unit and storage unit connections. Storage units that must be defined in a physical configuration are called elements. Units that must be defined as elements of a physical configuration include disk and tape controllers, disk and tape units, parallel processors, and network access devices.

Table 2-1. Summary of PCU Subcommands

Subcommand	Description
Element definition subcommands:	
DEFINE_ELEMENT (DEFE)	Describes the product type, serial number, and storage unit connections for a given element.
DEFINE_WORKING_MAINFRAME (DEFWM)	Defines a default mainframe identifier to be used in subsequent DEFINE_ELEMENT subcommands.
File editing subcommand:	
EDIT_PHYSICAL_CONFIGURATION (EDIPC)	Calls the EDIT_PHYSICAL_CONFIGURATION subutility, which can be used to create or edit a physical configuration file during deadstart.
Installation and verification subcommands:	
INSTALL_PHYSICAL_CONFIGURATION (INSPC)	Verifies, activates, and installs the physical configuration described in the input file. This subcommand can be used only during deadstart.
VERIFY_PHYSICAL_CONFIGURATION (VERPC)	Performs verification testing of the physical configuration described in the input file.
Termination subcommand:	
QUIT (QUI)	Terminates the PCU.

To install NOS/VE, you must perform two tasks using the PCU:

- Create a physical configuration file to describe the elements of the physical configuration.
- Install the physical configuration file during a deadstart of NOS/VE.

Creating a Physical Configuration File

The following sections present information needed when creating a physical configuration file.

Configuration File Contents

A physical configuration file consists of a series of PCU subcommands that describe the elements of the configuration. There are two PCU subcommands used to specify element definitions: the DEFINE_ELEMENT subcommand and the DEFINE_WORKING_MAINFRAME subcommand. Both of these subcommands are briefly described in table 2-1. Full descriptions of all subcommands are provided under PCU Command and Subcommands later in this chapter.

DEFINE_ELEMENT and DEFINE_WORKING_MAINFRAME are noninteractive subcommands; they can be entered only from a physical configuration file used as input to the PCU.

The following example shows what a physical configuration file looks like.

```
define_working_mainframe $SYSTEM_0855_0002
define_element element=ats_blue ..
               element_identification=$7021_32 ..
               serial_number=0001 ..
               iou_connection=(ch6, ..
                              (ch7,0,$SYSTEM_0855_0109))
define_element element=blue0 ..
               element_identification=$679_7 ..
               serial_number=2345 ..
               peripheral_connection=((ats_blue,0))
define_element element=blue1 ..
               element_identification=$679_6 ..
               serial_number=1234 ..
               peripheral_connection=((ats_blue,1))
define_element element=green_disc ..
               element_identification=$7155_14 ..
               serial_number=4321 ..
               iou_connection=(ch2, ..
                              (ch6,0,$SYSTEM_0855_0109))
define_element element=green32 ..
               element_identification=$885_11 ..
               serial_number=1222 ..
               peripheral_connection=((green_disc,32))
define_element element=green0 ..
               element_identification=$844_41 ..
               serial_number=1576 ..
               peripheral_connection=((green_disc,0))
```

Multiple Mainframes

The example file on the previous page contains references to two mainframes (one of the mainframes, $SYSTEM_0855_0002, is defined as the default mainframe in the DEFINE_WORKING_MAINFRAME subcommand; the other mainframe, $SYSTEM_0855_0109, is explicitly referenced in two of the DEFINE_ELEMENT subcommands). NOS/VE permits you to create a single physical configuration file that describes all mainframes and their attached storage units located at a site. There are a number of advantages to using a single configuration file to describe all of your mainframes:

- You need not create redundant storage unit definitions for each mainframe to which a storage unit is connected.
- Since each storage unit is identified by a single, unique name, the reporting of error conditions, usage statistics, or other information on a particular storage unit can be more easily coordinated within the site.
- Because each mainframe knows which of its storage units are connected to other mainframes, NOS/VE can make appropriate decisions regarding online concurrent maintenance access to peripherals.
- You can create a single deadstart file to be used by all mainframes at your site. This simplifies the creation and storage of deadstart tapes and eliminates the possibility of using the wrong tape.

An example of multiple mainframe configuration is found in the Site Analyst Examples online manual.

Dual-State Systems

In a dual-state system, you may define some or all of your NOS (or NOS/BE) peripherals in your NOS/VE physical configuration. The following restrictions apply:

- Only storage units supported by NOS/VE may be described in the NOS/VE configuration.
- Storage units dedicated to NOS (or NOS/BE) must be in the OFF state when defined in the NOS/VE configuration.

MALET/VE Restrictions on CYBER 93x Configurations

To allow MALET/VE to be used on CYBER 930-series mainframes with no impact on production, the system should be configured so that NOS/VE uses no more than seven I/O channels. To support this restriction, NOS/VE must use no more than four I/O channels in each of the clusters. This ensures a free peripheral processor in each cluster without making it necessary to idle an I/O subsystem or redeadstart the system.

Defining your NOS (or NOS/BE) peripherals in the NOS/VE configuration has the following advantages:

- Elements may be used alternately by both systems. The Logical Configuration Utility subcommand CHANGE_ELEMENT_STATE may be used to exchange ownership of tape units between the NOS/VE system and NOS (or NOS/BE).
- You can easily convert to a standalone NOS/VE system. To do this, you need only change device states from OFF to ON and initialize those mass storage (disk) volumes formerly used by NOS (or NOS/BE).
- Elements in the OFF state cannot be accessed by NOS/VE online concurrent maintenance (MALET/VE). This affords your NOS (or NOS/BE) elements more protection in a dual-state environment.

Your CIP (CYBER Initialization Package) device can, but need not be, defined in the NOS/VE configuration. If the CIP device is included in the NOS/VE configuration, it should be in the OFF state, even though NOS/VE occasionally accesses reserved cylinders on this device.

CIP Device Configuration

When configuring the CIP device, different considerations apply depending on whether the system is standalone or dual state. The following restriction applies to both types of systems: if any NOS/VE volume on an 844-4x, 885-1x, or 895-2 disk unit has CIP installed on it and the volume is to be initialized by NOS/VE, the system must have access to the volume through a nonconcurrent I/O (NIO) channel; otherwise CIP will be destroyed. Specifically, if the system device is also the CIP device, an NIO channel must be identified in the Deadstart and System Device Configuration Selections Menu when the system device is initialized; otherwise CIP will be destroyed.

For CYBER 962 and 992 mainframes (which support an I4C IOU subsystem), the CC598A PC console contains an 86-megabyte hard disk that serves as the CIP device for these configurations. The PC hard disk can be partitioned so that you can store multiple versions of CIP. We recommend that partition 0 always be used for your production version of CIP.

Standalone Systems

In a standalone system, the CIP device must be configured to NOS/VE. If the CIP device is in the OFF state, NOS/VE cannot access the device except for dedicated fault tolerance (DFT), whose access is confined to maintenance cylinders. If the CIP device is in the DOWN state, both DFT and MALET/VE may access the CIP device. If the CIP device is in the ON state, it may be initialized and added to the system set without compromising the CIP program. In a standalone system, we recommend that the system device also be the CIP device.

In a standalone system, take the following precautions when configuring the CIP device:

- On mainframes other than CYBER 930-series, 962, and 992 models:
  - If you have more than one physical path to the CIP device and the CIP device is also the NOS/VE system device, the CIP path must be identical to the path defined in the NOS/VE Deadstart and System Device Configuration Selections menu. If this condition is not met, deadstart fails. To ensure that the physical path is the same, the same IOU (IOU0), deadstart channel, equipment number, and unit number must be specified.

    If you must reconfigure access to the CIP device, you must remember to describe the reconfiguration in both the deadstart panel and the NOS/VE Deadstart and System Device Configuration menu the next time you deadstart.
  - If you have more than one physical path to the CIP device and the CIP path is not the path that NOS/VE is using to access files that may be on an 834 or 836 CIP device, you may encounter an error message when using the LCU subcommand CHANGE_ELEMENT_STATE to change the state of a channel or CPU. If this occurs, the state change is effective only until the next deadstart. The path NOS/VE is using is the first connection that is in the ON state in the DEFINE_ELEMENT subcommand for the controller and unit in the NOS/VE physical configuration. You should configure so that the CIP PATH and the NOS/VE path are identical.
  - If you have more than one physical path to the CIP device and any element in the CIP path is faulty, you may encounter an error message when using the LCU subcommand CHANGE_ELEMENT_STATE to change the state of a channel or CPU. If this occurs, the state change is effective only until the next deadstart.
- On CYBER 930-series mainframes: if you have more than one physical path to the CIP device and the CIP device is also the NOS/VE system device, the CIP PATH (the path to the CIP device specified in the Configure Critical Devices menu) must be identical to the path defined in the NOS/VE Deadstart and System Device Configuration Selections menu. If this condition is not met, deadstart fails. To ensure that the physical path is the same, the same channel, equipment number, and unit number must be specified. If you must reconfigure access to the CIP device, you need to describe the reconfiguration in both the Configure Critical Devices menu and the NOS/VE Deadstart and System Device Configuration menu the next time you deadstart.

Dual-State Systems

In a dual-state system, the CIP device may be defined in the NOS/VE configuration, but it must be in the OFF state. If the CIP device is not in the OFF state, NOS/VE will not deadstart successfully because it will fail to acquire the CIP device from the NOS (or NOS/BE) system.

A dual-state site need not have two CIP devices in order to experiment with NOS/VE standalone. The CIP disk in this case would be a NOS or NOS/BE disk that is only used by NOS/VE for booting the system and for error and clock recording. To prevent MALET/VE from accessing this device, we recommend that the NOS or NOS/BE CIP device be defined in the OFF state in the NOS/VE configuration.

Dual I4 IOU Configuration Restrictions

CYBER 180 model 840, 840A, 850, 850A, 860, 860A, 870A, 960, 990, 994, 990E, and 995E mainframes support a dual I4[1] IOU subsystem. The following restrictions apply to these configurations:

- The CIP device must be an 844-4x, 885-1x, or an 895-2 product.
- The CIP device must be connected to both IOUs (IOU0 and IOU1) via two NIO channels. The path from each IOU to the CIP device must have the same channel number, equipment number, and unit number.

The following additional restrictions apply to dual-state systems:

- NOS/BE dual-state systems do not support a dual I4 configuration.
- Only NOS/VE can use IOU1; NOS does not recognize IOU1.
- NOS does not recognize the possible interconnection of peripherals between the two IOUs.
- NOS/VE acquires ownership of NOS channels and storage units only if they are defined as ON or DOWN in the NOS/VE configuration.

Take the following precautions to coordinate dual-state access to storage units that are accessible from dual I4 IOUs. These precautions reduce the risk that NOS/VE storage units normally accessed only from IOU1 are inadvertently used by NOS via IOU0.

- Describe, in the NOS/VE configuration, all elements supported by NOS/VE even though they may be primarily or exclusively used by NOS. Describe the connections of these elements to both IOUs.
- Do not define the NOS/VE storage units in the NOS equipment status table (EST) if:
  - 844-4x or 885-1x storage units are accessible from both IOUs and the 7155-1x controller or controllers involved are not exclusively used by NOS/VE.
  - 895-2 storage units are accessible from both IOUs and the 7165-2x controller or controllers involved are not exclusively used by NOS/VE.
  - 887-1 storage units are accessible from both IOUs.

If 679-x or 698-3x tape storage units are accessible from both IOUs and the controller or controllers involved are not exclusively used by NOS/VE, perform the following steps to ensure that the NOS/VE tape storage units are DOWN in the NOS equipment status table (EST) during NOS/VE execution:

1. Define any of the storage units that are primarily used by NOS in the OFF state in the NOS/VE configuration.
2. Define any of the storage units used by NOS/VE in the ON or DOWN state in the NOS/VE configuration.

1. Here and elsewhere in this manual, references to I4 IOUs apply to I4A IOUs as well. From a software performance and maintenance standpoint, there is no difference between them.

3. Ensure that the state of all the channels that access the elements from both IOUs are in either the ON or DOWN state in the NOS/VE configuration.

4. If you want NOS to access some of the storage units that are accessible from both IOUs, you must change the state of the IOU0 channel to OFF after each deadstart. In addition, you must turn the channel back to ON in the NOS/VE configuration during each NOS/VE deadstart. NOS/VE retains dual-state ownership of any storage unit connected to the channel that was turned OFF as long as the IOU1 channel to the storage unit remains ON or DOWN in the NOS/VE configuration.

Configuration File Editors

Either of two editors can be used to create or modify a physical configuration file:[2]

- EDIT_PHYSICAL_CONFIGURATION subutility of the PCU
- EDIT_FILE utility

The EDIT_PHYSICAL_CONFIGURATION subutility provides a means of creating or modifying a physical configuration file during deadstart when the EDIT_FILE utility is not available. Information about how to use the EDIT_PHYSICAL_CONFIGURATION subutility and its subcommands is provided in the section of the same name later in this chapter.

The EDIT_FILE utility is available after the installation tape has been loaded during the operating system installation process and after the configuration terminal has been brought up (the NOS/VE system console supports EDIT_FILE utility operations in line mode only). Because of its screen mode operating features, the EDIT_FILE utility is more convenient to use than the EDIT_PHYSICAL_CONFIGURATION subutility.

2. During an installation deadstart, the system automatically creates a partial physical configuration file for you. This partial configuration file defines only the deadstart tape and system disk units that you describe on the Deadstart and System Device Configuration Selections Menu display. The partial configuration file is sufficient to complete the installation deadstart, but you must define and install the complete configuration before your system will be fully operational. A copy of the partial configuration file is available on file $SYSTEM.MAINFRAME.CONFIGURATION.

Installing a Physical Configuration File

A new physical configuration can be installed only during deadstart. The USE_INSTALLED_CONFIGURATION system core command determines which physical configuration file is installed at a given deadstart.

If the USEIC system core command is entered with a value of TRUE, or if no USE_INSTALLED_CONFIGURATION command is entered, the system reinstalls the active physical configuration. The active configuration is the physical configuration that was installed at the preceding deadstart and that may have since been changed using the LCU subcommand CHANGE_ELEMENT_STATE. Changes that occur in the active configuration after deadstart are retained across a continuation deadstart. These changes are not reflected in the file $SYSTEM.MAINFRAME.CONFIGURATION until the next deadstart completes.

If the USE_INSTALLED_CONFIGURATION system core command is entered with a value of FALSE, the system installs the physical configuration described on the deadstart file. Chapter 5, Deadstart File Software, describes how to put a physical configuration file on your deadstart file.

Before the configuration is installed, you have the option of modifying it using the EDIT_PHYSICAL_CONFIGURATION subutility. For details about how to do this, see Configuration File Editors later in this chapter.

The installed configuration is the physical configuration that has been manually created or read from the deadstart file during a deadstart and that has been effected by the PCU subcommand INSTALL_PHYSICAL_CONFIGURATION. When the deadstart is completed, a copy of the installed configuration is stored on file $SYSTEM.MAINFRAME.CONFIGURATION for use as read-only input to building a new deadstart file.

Activating the Physical Configuration

During deadstart, NOS/VE activates the most recently installed physical configuration. Activation of the physical configuration takes place only at a NOS/VE deadstart. The PCU subcommand INSTALL_PHYSICAL_CONFIGURATION is used to perform the actual activation.

Once a peripheral element is included in the active configuration, you can control its subsequent use by using the LCU subcommand CHANGE_ELEMENT_STATE (see chapter 3, Logical Configuration Utility).

At activation, NOS/VE does the following:

- Determines which elements in the installed configuration are physically connected to the mainframe being deadstarted.
- Determines whether an element is a unit. That is, it determines whether the element is a disk unit or tape unit for which software data structures must be established to support I/O request communication. An element is considered a unit if it satisfies one of the following criteria:
  - It has peripheral connections and no IOU connections.
  - It has IOU connections and no peripheral connections.
- Creates a unit interface table and a logical unit entry for each element classified as a unit.
- Determines which channels are connected to each unit.
- Activates each channel from the preceding step that satisfies the following conditions:
  - The channel is in an ON or DOWN state in the NOS/VE configuration.
  - At least one unit in an ON or DOWN state is connected to the channel.
  - All the units connected to the channel have ELEMENT_IDENTIFICATION entries that NOS/VE recognizes. If an element is not recognized by NOS/VE (that is, if it was defined using the VERIFY_ELEMENT_IDENTIFICATION=FALSE parameter of the PCU subcommand DEFINE_ELEMENT), it is not activated at deadstart.
  - The IOU program name for the unit is defined in the physical configuration file (or can be determined by NOS/VE from the unit's element identification), and the object module corresponding to the IOU program name is resident on the NOS/VE deadstart file.
  - The channel (in a dual-state system) is requested from NOS (or NOS/BE) and was acquired. In addition, if the units must be acquired from NOS (or NOS/BE), all the units were also acquired.
- Determines the number of PPs required to support the channel access to the unit. If the IOU program name of the peripheral subsystem begins with the letter E and the second character is a 1 or a 2, one or two PPs are used, respectively. Otherwise, only one PP is assigned. For more information, see the description of the PCU subcommand DEFINE_ELEMENT.

- Creates a PP interface table for each PP required to support access to an activated channel. If two PPs are required for a channel, NOS/VE arbitrarily selects one PP to be the master and the other to be the slave. This assignment is indicated in the communication buffer for each PP. Within the active configuration, a unit descriptor is created for each unit that is accessible to the PP being configured.
- Acquires the number of PPs required for each activated channel that is in an ON state. In a dual-state system, PPs are acquired from NOS (or NOS/BE).
- Deadstarts the appropriate IOU program in each PP required.
- Searches for system labels on disk units and bootstraps the file system, during a continuation deadstart.

Hardware Element Identification

NOS/VE can access a hardware element (a controller, tape unit, disk unit, or MAP V parallel processor) only if the element is fully described by PCU subcommands. Three pieces of information are necessary to uniquely identify a hardware element:

- Element name
- Product identification
- Serial number

An element is a hardware product or component of a product that is individually addressable. For example, a 7155-12 disk controller and an 844-44 disk unit are both hardware elements. The 885-11 disk product consists of two separate elements, since each of the two spindles of the 885-11 product is individually addressable.

A product name is the name of the hardware that was purchased from Control Data. For example, a 7155-12 disk controller, an 844-44 disk unit, and an 885-11 disk unit are products. As mentioned above, several products (including the 885-1x, the 836-1xxx, and the 895-2) consist of multiple elements.

An equipment is a component of a product. An equipment has its own Control Data manufacturing identification, which includes a serial number. This equipment identification is in the form of a label located either on the outside or inside of the cabinet of the product that includes the equipment.

Hardware Addressing

A hardware address is the numerical identification of a hardware device on a channel cable or connecting cable. The numerical identification is defined by mechanical means, such as attaching the cable to a numbered, physical port.

The hardware address from a data channel to an element is referred to as the equipment number. The address from a controller to a storage unit is called the unit number.

Element Name

An element name is the name you assign to a hardware element. Because the system uses the element name to report errors, usage, and other facts about the element, you must assign unique names to all peripheral elements.

When you communicate with the system, you use element names to identify individual elements. It is helpful if you assign an element a name that is easily associated with the element. For example, if you have two disk controllers, you could name one BLACK_CONTROLLER and the other WHITE_CONTROLLER. Then you could name the disk units connected to these controllers BLACK_0, BLACK_1, WHITE_0, WHITE_1, and so forth.

If an element is removed from the physical configuration and replaced by an element that is functionally equivalent, the element name should remain the same. You need change only the serial number. This way the operator does not have to learn a new name for what appears to be the same element.

Element definitions for controllers or for 887 disk units require you to specify the name of the channel or channels to which the controller is attached. Names of NOS/VE channels, IOUs, and mainframes are assigned by the system.

Channel names have the following format:[3]

CHn	For nonconcurrent I/O (NIO) channels on an I0, I1, I2, I4, or I4A IOU.
CCHn	For concurrent DMA (CIO) channels without a port designation on an I4, I4A, or I4C IOU.
CCHnp	For concurrent DMA (CIO) channels with a port designation on an I4, I4A, or I4C IOU.

The n field specifies the decimal channel number, and the p field specifies the port (either A or B). A NOS (or NOS/BE) channel 20 (octal), for example, is automatically assigned the name CH16 by NOS/VE. Refer to appendix E, Supported Hardware Products, for more information about which peripherals may be connected to these channels.

Whenever you need to supply the name of an IOU, the name must have the following form:

IOUn

The n field is either 0 (zero) or 1.

3. An NIO channel is a channel that does not allow the PP that controls the data transfer to execute other instructions during the transfer. A CIO channel allows the PP to execute independently of the data transfer. A CIO channel is only supported in the I4 class of IOUs. A DMA channel allows data to be transferred directly between central memory and the channel. A DMA channel may be used in either a DMA or non-DMA mode. See appendix E for information about which devices may be connected to each type of channel.

Whenever you need to supply a mainframe name, the name must have the following form:

$SYSTEM_mmmm_nnnn

The mmmm field is the mainframe model number (for example, 0830, 0855, or 0990); the nnnn field is the serial number of the mainframe. If you have a dual-processor mainframe, use the serial number of CPU 0. The NOS/VE function $mainframe(identifier) returns a string representation of the mainframe name.

For a list of reserved element names, refer to the description of the DEFINE_ELEMENT subcommand later in this chapter.

Hardware Product Identification

The element identification, which is specified on the DEFINE_ELEMENT subcommand, identifies a hardware product or a component. The value specified for the element identification is the element's product name expressed in the following format:

$product_model

For example, the product identification of a 7155-11 fixed module drive (FMD) disk controller is $7155_11.

Two products may be made up of identical equipments but may differ in capability because one product has more optional equipments installed than the other. The model number in the product identification reflects this difference. For example, the 844-41 and 844-44 disk units are products with identical equipments. They differ in that the 844-41 has two controller accesses and the 844-44 has four controller accesses.

Serial Number

A serial number is a 1- to 6-digit integer number that uniquely identifies a Control Data equipment. A product may be made up of several equipments. For the NOS/VE system to uniquely identify a hardware product, one equipment has been selected from those that make up the product; the serial number of this equipment is used as the serial number for the entire product. Refer to table E-1, Hardware Product Serial Numbers, to determine from which equipment the serial number should be taken. The serial number of the selected equipment and the product identification uniquely identify the product for maintenance purposes.

The PCU requires that you specify a unique serial number for most elements. The only exception is for multispindle disk units such as the 885-1x and 895-2 disk units. For the 885-1x and 895-2 units, NOS/VE uses the physical address, in addition to the element identification and serial number, to identify each element. The 885-1x disk unit can have up to two elements with the same serial number, and the 895-2 disk unit can have up to four elements with the same serial number.

If you replace an element by one with an identical product identification, the element name and element identification can remain the same; however, you may need to change the serial number of the element. Refer to table E-1, Hardware Product Serial Numbers. If you replace the equipment identified in column 2 of that table, change the serial number of the corresponding elements in the physical configuration.

PCU Command and Subcommands

This section describes the PHYSICAL_CONFIGURATION_UTILITY (PCU) command, followed by descriptions of the PCU subcommands and the EDIT_PHYSICAL_CONFIGURATION subutility. The PCU subcommands are presented in alphabetical order and are then followed by a description of the subcommands of the EDIT_PHYSICAL_CONFIGURATION subutility.

PHYSICAL_CONFIGURATION_UTILITY command

- DEFINE_ELEMENT subcommand
- DEFINE_WORKING_MAINFRAME subcommand
- EDIT_PHYSICAL_CONFIGURATION subcommand
- INSTALL_PHYSICAL_CONFIGURATION subcommand
- QUIT subcommand
- VERIFY_PHYSICAL_CONFIGURATION subcommand

EDIT_PHYSICAL_CONFIGURATION subutility

- ADD_ELEMENT_DEFINITION subcommand
- CHANGE_CONNECTION_REFERENCES subcommand
- CHANGE_ELEMENT_DEFINITION subcommand
- CHANGE_ELEMENT_NAME subcommand
- DELETE_ELEMENT_DEFINITION subcommand
- DISPLAY_CONNECTED_ELEMENTS subcommand
- DISPLAY_ELEMENT_DEFINITIONS subcommand
- $ELEMENT_DEFINITION function
- QUIT subcommand
- REPLACE_ELEMENT_DEFINITION subcommand

You can enter the DEFINE_ELEMENT and DEFINE_WORKING_MAINFRAME subcommands only from a physical configuration file used as input to the EDIT_PHYSICAL_CONFIGURATION, INSTALL_PHYSICAL_CONFIGURATION, or VERIFY_PHYSICAL_CONFIGURATION subcommands. You cannot enter them interactively.

Except for INSTALL_PHYSICAL_CONFIGURATION and QUIT, you can enter the PCU subcommands in any order. INSTALL_PHYSICAL_CONFIGURATION is the last subcommand you enter before QUIT. The QUIT subcommand terminates the utility session.

PHYSICAL_CONFIGURATION_UTILITY Command

Purpose Calls the Physical Configuration Utility.

Format **PHYSICAL_CONFIGURATION_UTILITY** or
PCU
  STATUS=status variable

Parameters *STATUS*

Returns the completion status for the entire utility. The status variable is automatically initialized at the beginning of the utility and set appropriately at the end of the utility. Optionally, you can explicitly create your own status variable to control error processing. See the NOS/VE System Usage manual for details about how to create a status variable.

A status variable on a subcommand returns the completion status for that subcommand.

DEFINE_ELEMENT Subcommand

Purpose Defines an element of the physical configuration.

Format

DEFINE_ELEMENT or
DEFE
  ELEMENT=name
  SAME_AS=name
  ELEMENT_IDENTIFICATION=keyword
  IOU_PROGRAM_NAME=name
  SERIAL_NUMBER=integer
  STATE=keyword
  CENTRAL_MEMORY_CONNECTION=list of record
  IOU_CONNECTION=list of record
  PERIPHERAL_CONNECTION=list of record
  VERIFY_ELEMENT_IDENTIFICATION=boolean
  APPLICATION_INFORMATION=string
  SITE_INFORMATION=string

Parameters

ELEMENT or E

Specifies the name to be given to the element being defined. This parameter is required. The element name must be unique within the physical configuration. You can specify any valid SCL name except the following reserved names:

ALL
CCHn (where n is an integer)
CCHnp (where n is an integer and p is either A or B)
CHn (where n is an integer)
CM
CPn (where n is an integer)
CPPn (where n is an integer)
IOUn (where n is an integer)
PPn (where n is an integer)
$CENTRAL_MEMORY
$CENTRAL_PROCESSOR
$CHANNEL
$CHANNEL_ADAPTER
$COMMUNICATIONS_ELEMENT
$CONTROLLER
$EXTERNAL_PROCESSOR
$IOU
$MAGNETIC_TAPE_DEVICE
$MAINFRAME
$MASS_STORAGE_DEVICE
$NETWORK_DEVICE
$PERIPHERAL_PROCESSOR
$RHFAM_DEVICE
$STORAGE_DEVICE
$SYSTEM_mmmm_nnnn(where mmmm is the mainframe model number and nnnn is the serial number)

SAME_AS or *SA*

Specifies the name of a previously defined element in the physical configuration from which the element identification, serial number, IOU program name, and state information for this element are copied. Note that connection-related information is not copied.

You can specify any or all of the ELEMENT_IDENTIFICATION, SERIAL_NUMBER, IOU_PROGRAM_NAME, or STATE parameters in addition to the SAME_AS parameter. In this case, the values specified for the individual parameters take precedence over the values obtained by the SAME_AS parameter.

You can only use this parameter if the DEFINE_ELEMENT subcommand that defines the referenced element precedes the current DEFINE_ELEMENT subcommand within the physical configuration file.

ELEMENT_IDENTIFICATION or **EI**

Specifies the hardware identification of the element. The value you specify must be either one of the names in the first column of table E-1, Hardware Product Serial Numbers, or a keyword. If you specify a keyword, NOS/VE substitutes the appropriate element name as described in Remarks. This parameter is required.

IOU_PROGRAM_NAME or *IPN*

Specifies the name of the IOU program NOS/VE will use to communicate with the element. This parameter is valid only for elements with IOU connections. The program specified by this parameter is loaded into each peripheral processor (PP) that communicates with this element. Only one IOU program name may be defined for a particular subsystem; if a peripheral element is accessible from more than one PP, the same program must be executing in each PP. The exception is the 895 family of disk units, which can be accessed from two channels. A different program is used for an NIO channel connection than for a CIO channel connection. NOS/VE automatically selects the correct program for each 895 channel. Note that each 895 channel requires two PPs.

If you omit this parameter, NOS/VE automatically selects the correct IOU program based upon the mainframe model and/or channel name.

By default, NOS/VE requires one PP to drive each magnetic tape controller. For the $7021_3x, $698_1x, and $698_2x magnetic tape controllers, specify IOU program names E1A7021 or E1C7021 if you want to use one PP to drive the subsystem. Use E2A7021 or E2C7021 if you want to use two PPs. For the $7221_1 controller, use E1C7021 for a single PP driver, and E2C7021 for two PPs. A single PP driver for $7021_3x, $7221_1, $698_1x, and $698_2x products is limited to reading and writing physical records (blocks) of 4,128 bytes or less. Blocks greater than 4,128 bytes require a different IOU program that requires two PPs. You must specify this IOU program when configuring the magnetic tape controller. Refer to Appendix E, Supported Hardware Products, for more information on magnetic tape block size restrictions.

The IOU program for the $9639_1 and $5698_1x products processes tape block sizes in excess of 4,128 bytes with a single PP. When you configure the $7221_11 controller, you do not need to specify an IOU program name.

All open-reel magnetic tape subsystems connected to non-CYBER 930-series[4] NOS/VE systems must be configured for the same maximum magnetic tape block size. For example, if you have a $5698_1x product and you configure the E1A7021 program for a $7021_31 controller, you must limit open-reel tape use to blocks that are less than or equal to 4,128 bytes. Failure to do so may cause jobs to fail. A system interruption is required to reconfigure the system to a different maximum block size.

The $5680_11 cartridge tape subsystem always requires two PPs. However, it does not require open-reel subsystems to be configured with two PPs.

When you configure the elements described in the following table, we recommend that you allow this parameter to default except when defining magnetic tape products. Specify the IOU_PROGRAM_NAME value that is appropriate for the maximum tape block size you want to access.

The IOU programs for NOS/VE storage units are listed in the following table. The first program listed for a storage unit is the default program for that unit.

Element Identification	IOU Program Name
$2620_21x/$2621_21x	E1A2620 or E1C2620
$2629_2	E1I2629
$380_170	E1A380 or E1C380 or E1I380
$5380_100/$7040_200	E1A5380 or E1C5380
$5680_11	E2A5680 or E2C5680
$5698_1x (I0 IOU)	E5P5698
$5698_1x (I4 IOU)	E9P5698
$65354_1x	E1A6535 or E1C6535
$698_xx	E1A7021 or E1C7021 or E2A7021 or E2C7021
$7021_3x	Same as $698_xx
$7155_1	E1A7155 or E1C7155
$7155_1x	E1C715B
$7165_2x	E9A7165 or E2C7165
$7221_1	E1C7021 or E2C7021
$7221_11	E5I9639
$887_1	E9S887
$FA7B4_D/$10395_11	E1I7255
$FA7B5_A (I0 IOU)	E5P9836
$FA7B5_A (I4 IOU)	E9P9853

4. The CYBER 930 series includes the CYBER 930-11, 930-31, 932-11, 932-31, and 932-32.

IOU program names are assigned according to specific naming conventions. The following name format is used:

Epcssss

E	Identifies the named program as being an operating system or I/O program.
p	Indicates one of the following:

1	Single PP required; data transfer through processor memory.
2	Two PPs required; data transfer through processor memory.
5	Single PP required; data transfer directly to and from central memory (Direct Memory Access [DMA])
9	Same as 5 except the data transfer is concurrent with IOU program execution.

c	Indicates the type of channel:

A	I4 concurrent channel adapted to CYBER 170 12-bit channel.
C	CYBER 170 12-bit channel.
I	CYBER 170 16-bit ICI channel.
P	IPI 16-bit channel.
S	ISI 16-bit channel.

ssss	Are alphanumeric characters that identify the elements that are accessed through the IOU program. NOS/VE places no restrictions on the characters used for this field.

For more information about IOUs, see table E-4, Characteristics of CYBER 180 IOUs.

SERIAL_NUMBER or SN

Specifies the serial number of the element defined by the ELEMENT_IDENTIFICATION parameter. This parameter is required.

Each hardware product includes several equipments, each having its own serial number. The second column of table E-1, Hardware Product Serial Numbers, shows which equipment serial number to use to define the peripheral device specified for the ELEMENT_IDENTIFICATION parameter.

STATE or *S*

Specifies the state of the element being defined. You can specify one of the following keywords; the default is ON:

ON

Specifies that the element is operational and is available for concurrent maintenance.

DOWN

Specifies that the element is available only for maintenance purposes.

OFF

Specifies that the element is not available for normal operations or for maintenance purposes.

CENTRAL_MEMORY_CONNECTION or CENTRAL_MEMORY_ CONNECTIONS or *CMC*

Specifies a list of one or more records that define the mainframe and central memory ports to which the element is connected; valid only for the $65354_xx, MAP V parallel processor. Each record has the following format:

(**port**, *mainframe)*

port

Specifies an integer value from 0 to 3. This entry is required.

mainframe

Defines the name of the mainframe to which the element is connected. The default is the mainframe name specified in the DEFINE_ WORKING_MAINFRAME subcommand.

IOU_CONNECTION or *IOU_CONNECTIONS* or *IC*

Specifies a list of one or more records that define connections between this element and a mainframe IOU. Each record has the following format:

(**channel**, *equipment, mainframe, iou)*

channel

Specifies the name of the channel that connects the peripheral element to the host mainframe. This entry is required.

equipment

Specifies the decimal address of the peripheral on the channel. The default is 0 (zero).

mainframe

Specifies the name of the mainframe to which the IOU and channel belong. The default is the mainframe name specified by the most recent DEFINE_WORKING_MAINFRAME subcommand.

iou

Identifies the IOU to which the channel belongs. If you have a system that uses multiple IOUs, always specify the correct IOU name in the IOU_CONNECTION parameter. The default for this parameter is IOU0.

For the controllers $FA7B4_D, $FA7B5_A, $10395_11, $5698_1x, $7155_1x, and $7155_1, if you define more than one IOU connection, only one connection is activated at deadstart. The first connection whose corresponding channel is in the ON state is chosen.

For the $887_1 storage unit, if you define more than one IOU connection and the connections are to two ports of the same channel, only the first connection is activated at deadstart. If the connections are to two different channels, any channel in the ON state is activated at deadstart.

The $698_3x, $7165_2x and $7021_32 controllers support connection to up to two host channels. For the current version of NOS/VE, if you define both channels in the NOS/VE physical configuration for a dual-state system, NOS/VE attempts to acquire both channels from NOS (or NOS/BE). This could cause the installation to fail if one of the channels was intended for use by NOS (or NOS/BE). If one of the channels is to be used by NOS (or NOS/BE), do not define that channel by an IOU_CONNECTION parameter; define only the channel to be used by NOS/VE.

Each NIO and CIO channel configured to a $7165_2x controller requires two PPs. When configured using a CIO channel, both PPs and the corresponding channel must be from the same cluster (that is, a grouping of PPs and channels that are physically distinct from other such groupings). For more information, refer to the peripheral device descriptions in appendix E, Supported Hardware Products.

The following table lists the peripheral elements that have IOU connections to be defined. The second column defines the address or range of addresses you can specify in the equipment field.

Element	Equipment Address
$10395_11	0 to 7
$380_170	0
$5380_100	0
$5680_11	0
$5698_1x	0 to 7
$65354_xx	0
$698_1x	0 to 7
$698_2x	0 to 7
$7021_3x	0
$7040_200	0
$7155_1x	0
$7165_2x	0 or 1
$7221_1	0
$7221_11	0
$887_1	0 to 7
$FA7B4_D	0 to 7
$FA7B5_A	0 to 7

PERIPHERAL_CONNECTION or *PERIPHERAL_CONNECTIONS* or *PC*

Specifies a list of one or more records that define connections between this element and another peripheral element or elements in the physical configuration. Peripheral elements are usually connected in a hierarchical manner; therefore, when describing the connections of an element, it is only necessary to describe its connections to the parent or upline element to which it is attached.

Each record has the following format:

(element, *address)*

element

Specifies the name of the element to which this element is connected. This entry is required.

address

Specifies the hardware address to the element being defined from the parent connection element. The default is zero.

The following table lists the elements having peripheral connections that must be defined:

Element Being Defined	Parent Connection	Valid Addresses[5] (Unit Number)
$5682_1x	$5680_11	0 to 15
$639_1	$7221_1	0
$679_x	$7021_3x	0 to 7
$698_3x	$5698_1x	0 to 7
$698_3x	$698_1x	0 to 7
$698_3x	$698_2x	0 to 7
$834_12	$10395_11	0 to 3
$836_xxx	$FA7B4_D	0 to 3
$844_4x	$7155_1x	0 to 7
$885_1x	$7155_1x	32 to 47
$895_2	$7165_2x	0 to 31[6]
$9639_1	$7221_11	0 to 1
$9836_1	$FA7B5_A	0 to 7
$9853_x	$FA7B5_A	0 to 7

5. Decimal values.

6. For the $7165_2x controller, the valid physical address is obtained using the expression: (DDC address * 16) + unit number. The DDC is described in appendix E, Supported Hardware Products, under $7165_2x.

VERIFY_ELEMENT_IDENTIFICATION or *VEI*

Specifies whether or not this element should be validated against the list of elements which are recognized by NOS/VE. You can specify the following keywords; the default is TRUE:

TRUE

Validation is performed.

FALSE

No validation is performed.

Specify FALSE for this parameter if you are defining an element that is not recognized by NOS/VE. Specifying FALSE for elements that are recognized by NOS/VE inhibits the system from checking for spelling errors in the element identification.

APPLICATION_INFORMATION or *AI*

Specifies information to be used by an application that logically configures this element.

SITE_INFORMATION or *SI*

Specifies information that the site chooses to associate with this element.

Remarks

- The DEFINE_ELEMENT subcommand is included in the physical configuration file when that file is created or edited. You cannot enter the DEFINE_ELEMENT subcommand interactively.
- For dual-state systems, each $679_x, $844_4x, or $885_1x disk storage or tape unit must have an EST entry in the NOS EQPDECK or NOS/BE CMR. Tables E-2 and E-3 (NOS EQPDECK Entries for Peripherals and NOS/BE CMR Entries for Peripherals, respectively) list the EST and CMR requirements for each storage unit. All disk storage and tape units to be used by NOS/VE must also be specified in the NOS/VE physical configuration.
- For $9836_1 and $9853_x units: if you define more than one peripheral connection, only one connection is activated at deadstart. The first connection whose corresponding controller is activated is used.
- The $885_12 and $887_1 disk storage products should have two DEFINE_ELEMENT subcommands, one for each spindle (if both spindles are to be used by NOS/VE). The values for the ELEMENT_IDENTIFICATION and SERIAL_NUMBER parameters of DEFINE_ELEMENT should be the same for both spindles of the $885_12. For the $887_1, each spindle has a unique serial number.
- An $895_2 disk storage product should have four DEFINE_ELEMENT subcommands, one for each spindle. All four spindles have the same serial number.

- If, on the ELEMENT_IDENTIFICATION parameter, you specify a generic keyword (such as $7165_2x), NOS/VE derives the specific element identification by examining the number and types of connections specified for the element. For example:

  - If you specify one or two IOU connections and an element identification of $7165_2x, NOS/VE assumes you have a $7165_21 controller. If you specify three or four IOU connections, NOS/VE assumes you are identifying a $7165_22 controller.

  - If you specify $5682_1x, NOS/VE assumes you are identifying a $5682_12 cartridge tape unit.

  - If you specify $698_1x, $698_2x, or $698_3x, NOS/VE assumes you are identifying a $698_10, $698_20, or $698_30 tape subsystem component.

  - If you specify $836_xxx, NOS/VE assumes you are identifying a $836_110 disk subsystem.

  - If you specify $2620_xxx or $2621_xxx, NOS/VE assumes you are identifying a $2620_210 or $2621_210 MTI, respectively.

  - If you specify $65354_xx and do not specify a central memory connection, NOS/VE assumes you are identifying a $65354_10 MAP V. For a CYBER 835 mainframe, this value is interpreted as $65354_11. For a CYBER 855 mainframe, the $65354_xx keyword is interpreted as $65354_12.

- For each element, you must specify either an IOU_CONNECTION or a PERIPHERAL_CONNECTION parameter. For a $65354_xx element, you must specify both an IOU_CONNECTION and a CENTRAL_MEMORY_CONNECTION parameter.

- If you want to access open-reel magnetic tapes whose block size exceeds 4,128 bytes, you must specify one of the following IOU program names when defining the magnetic tape controller:

Element	IOU Program Name
$698_1x	E2A7021 or E2C7021
$698_2x	E2A7021 or E2C7021
$7021_3x	E2A7021 or E2C7021
$7221_1	E2C7021

- When you configure a $5380_100 (STORNET) or $7040_200 (ESM-II) element, it is extremely important that you identify all mainframe connections to these elements in the IOU_CONNECTION parameter of the DEFINE_ELEMENT subcommand for the NOS/VE physical configuration. Otherwise, the NOS/VE online maintenance of the element may prevent that element from being used by other mainframes.

- When you configure a $7040_200 (ESM-II) subsystem, it is essential that you identify each CYBER 170 mainframe that is connected to the $7040_200 element via a high-speed port. NOS/VE online maintenance requires knowledge of these connections. For each CYBER 170 mainframe, specify an IOU connection in the following form:

  ```
  (CH10 0 $SYSTEM_mmmm_nnnn IOU0)
  ```

  The CH10 value is not actually an external channel. Rather, it indicates, by convention, that a CYBER 170 CPU connection exists.

  The mmmm value specifies the CYBER 170 CPU model number.

  The nnnn value specifies the CYBER 170 CPU serial number of processor zero.

- A side-door (maintenance) port of a $5380_100 or $7040_200 element must be connected to a NOS system. NOS is responsible for monitoring the element and logging faults for maintenance action.

Examples

The following example defines an ESM-II element to NOS/VE. The three mainframes are a CYBER 180 model 990, a CYBER 180 model 855, and a CYBER 170 model 173. The CH10 identifier makes NOS/VE aware of the existence of the CYBER 170 mainframe on a high-speed port.

```
define_element element=esm_2 ..
  element_identification=$7040_200 ..
  serial_number=nnnn ..
  iou_connection=((ch0 $system_0990_0102 iou0) ..
                  (ch1 $system_0855_0109 iou0) ..
                  (ch10 $system_0173_0004 iou0))
```

DEFINE_WORKING_MAINFRAME Subcommand

Purpose

Defines the working (default) mainframe name for subsequent DEFINE_ELEMENT subcommands, specifically for the IOU_CONNECTION and CENTRAL_MEMORY_CONNECTION parameters of that command.

Format

DEFINE_WORKING_MAINFRAME or
DEFWM
   NAME=name

Parameters

NAME or N

Specifies the name of the mainframe. This parameter is required. The name specified must have the following format:

   $SYSTEM_mmmm_nnnn

The mmmm value is the mainframe model number, and the nnnn value is the mainframe serial number.

Remarks

- You cannot enter the DEFINE_WORKING_MAINFRAME subcommand interactively.
- If your site has only one mainframe, the DEFINE_WORKING_MAINFRAME subcommand is not necessary; the mainframe field in the IOU_CONNNECTION and CENTRAL_MEMORY_CONNECTION definitions of all elements defaults to the mainframe on which the system is being installed.
- Use the DEFINE_WORKING_MAINFRAME subcommand when creating a physical configuration file using COLLECT_TEXT or EDIT_FILE after NOS/VE has been installed. When a file prepared in this manner is processed by the EDIT_PHYSICAL_CONFIGURATION subutility, the mainframe names provided by the DEFINE_WORKING_MAINFRAME subcommand are written into the appropriate DEFINE_ELEMENT subcommmands; the actual DEFINE_WORKING_MAINFRAME subcommands are not written to the output file of the EDIT_PHYSICAL_CONFIGURATION subutility.
- For an example, refer to Configuration Examples later in this chapter.

EDIT_PHYSICAL_CONFIGURATION Subcommand

The EDIT_PHYSICAL_CONFIGURATION subcommand of the PCU is described in the EDIT_PHYSICAL_CONFIGURATION Subutility section later in this chapter.

INSTALL_PHYSICAL_CONFIGURATION Subcommand

Purpose This subcommand verifies, activates, and installs a new physical configuration.

Format **INSTALL_PHYSICAL_CONFIGURATION** or
INSPC
INPUT=file
ERRORS=file
STATUS=status variable

Parameters *INPUT* or *I*

Specifies the name of the file containing the physical configuration. The input file may contain configuration definition subcommands and SCL control statements. Other NOS/VE commands are not allowed. The default is $LOCAL.PHYSICAL_CONFIGURATION.

We recommend that you not specify $INPUT as the file name if the definition subcommands are to be entered interactively; should the installation attempt fail, the subcommands would need to be reentered in their entirety. Instead, prepare a file using the EDIT_PHYSICAL_CONFIGURATION subutility or the COLLECT_TEXT command.

ERROR or *ERRORS* or *E*

Specifies the name of the file to which errors detected by this subcommand are written. If a subcommand has a syntax error, the subcommand is echoed to the file, followed by an error message. Messages describing errors arising from interdependencies between subcommands are written at the end of the file. The default file name is $ERRORS.

STATUS

Returns the completion status for this subcommand.

Remarks

- You must enter this command from the system console.
- You can execute this command only during deadstart.
- You must create the input file containing the element definition subcommands before you enter the INSTALL_PHYSICAL_CONFIGURATION subcommand.
- Verification determines the logical validity of the connections specified in the physical configuration. Installation copies the file onto the system disk. Activation refers to software initialization necessary to access elements in the physical configuration. In a dual-state environment, activation includes obtaining PPs, channels, and peripheral elements from the NOS or NOS/BE system.
- For a physical configuration that includes more than one mainframe, the INSTALL_PHYSICAL_CONFIGURATION subcommand activates only the subset of the configuration that describes the mainframe on which the PCU is currently executing.
- If this subcommand returns an abnormal status, any previously installed physical configuration remains unaffected by the installation attempt.

QUIT Subcommand

Purpose	Terminates the Physical Configuration Utility (PCU).
Format	**QUIT** or **QUI**
Parameters	None.

VERIFY_PHYSICAL_CONFIGURATION Subcommand

Purpose Performs verification of a physical configuration prior to installing the configuration. Only the logical consistency of the defined connections are verified; this subcommand does not validate that the specified elements are physically connected as described.

Format

VERIFY_PHYSICAL_CONFIGURATION or
VERPC
MAINFRAME=name
INPUT=file
ERRORS=file
STATUS=status variable

Parameters

MAINFRAME or *M*

Specifies the name of the mainframe for which the physical configuration is to be verified. The name specified must be in the format $SYSTEM_mmmm_nnnn. The mainframe specified need not be the same mainframe on which the PCU is currently executing. The default is the name of the mainframe on which the PCU is currently executing.

INPUT or *I*

Specifies the name of the file containing the physical configuration to be verified. The input file may contain the PCU subcommands DEFINE_WORKING_MAINFRAME and DEFINE_ELEMENT as well as SCL control statements; other NOS/VE commands are not allowed. The default is $LOCAL.PHYSICAL_CONFIGURATION.

PCU subcommands to be verified can be input interactively from file $INPUT; however, each time an error is encountered, you are required to reenter all subcommands. For this reason, we recommend you use the EDIT_PHYSICAL_CONFIGURATION subutility, the EDIT_FILE utility, or the COLLECT_TEXT command to create a file containing the subcommands; then submit the entire file for verification.

ERRORS or *E*

Specifies the name of the file to which error messages are to be written. The default is $ERRORS.

If a subcommand has a syntax error that can be detected independently of other subcommands, the subcommand is echoed to the file followed by an error message. Messages describing errors in the description of logical connections are written to the end of the file.

STATUS

Returns the completion status for this subcommand.

EDIT_PHYSICAL_CONFIGURATION Subutility

The EDIT_PHYSICAL_CONFIGURATION subutility is available during system deadstart or during normal operations. You initiate the subutility with the EDIT_PHYSICAL_CONFIGURATION subcommand and terminate the subutility with the QUIT subcommand. Subutility subcommands allow you to display, add, modify, or delete element definitions in the physical configuration.

You specify input and output files for the subutility on the EDIT_PHYSICAL_CONFIGURATION subcommand. The default file for both input and output is $LOCAL.PHYSICAL_CONFIGURATION. During a continuation deadstart, the previously installed physical configuration is automatically copied from a system file to $LOCAL.PHYSICAL_CONFIGURATION. In this way, the system makes the installed configuration available for modification through the subutility.

The following example demonstrates how a file can be modified using the EDIT_PHYSICAL_CONFIGURATION subutility. This example assumes that the installed configuration is being modified during a continuation deadstart. The PCU session is initiated when the operator responds positively to the deadstart PCU prompt.

In this example, an existing definition for element GREEN0 is replaced by a new definition. The element definition is then displayed, a correction is made, and the element is displayed a second time. After exiting the subutility, the operator enters an INSTALL_PHYSICAL_CONFIGURATION subcommand to install the modified file, then enters QUIT to terminate the PCU.

```
You have the following choices for reconfiguration:

1 - Intervene before installing the physical configuration.

2 - Intervene before activating existing mass storage set members.

3 - Intervene after activating existing mass storage set members.

Enter selection, GO, or ? for HELP.
? 1

PCU/edit_physical_configuration
PCE/replace_element_definition e=green0 ei=$844_41 ..
PCE../sn=1576 pc=((green_7155 32))
PCE/display_element_definition e=green0
DEFINE_ELEMENT E = GREEN0 ..
   ELEMENT_IDENTIFICATION = $844_41 STATE = ON SERIAL_NUMBER = 1576 ..
   PERIPHERAL_CONNECTION = ( ..
        ( GREEN_7155 32 )) ..
   VERIFY_ELEMENT_IDENTIFICATION = TRUE
PCE/change_element_definition e=green0 sn=1516
PCE/display_element_definition e=green0
```

```
DEFINE_ELEMENT E = GREENO ..
   ELEMENT_IDENTIFICATION = $844_41 STATE = ON SERIAL_NUMBER = 1516 ..
   PERIPHERAL_CONNECTION = ( ..
          ( GREEN_7155 32 )) ..
   VERIFY_ELEMENT_IDENTIFICATION = TRUE

PCE/quit
PCU/install_physical_configuration
PCU/quit
/
```

The EDIT_PHYSICAL_CONFIGURATION utility subcommands and function are as follows:

ADD_ELEMENT_DEFINITION
CHANGE_CONNECTION_REFERENCES
CHANGE_ELEMENT_DEFINITION
CHANGE_ELEMENT_NAME
DELETE_ELEMENT_DEFINITION
DISPLAY_CONNECTED_ELEMENTS
DISPLAY_ELEMENT_DEFINITIONS
$ELEMENT_DEFINITION
QUIT
REPLACE_ELEMENT_DEFINITION

EDIT_PHYSICAL_CONFIGURATION Subcommand

Purpose Initiates the EDIT_PHYSICAL_CONFIGURATION subutility. This subutility provides subcommands that add, change, delete, and display element definitions contained in the input file. The result is written to the output file.

Format **EDIT_PHYSICAL_CONFIGURATION** or
EDIPC
INPUT=file
OUTPUT=file
ERRORS=file
STATUS=status variable

Parameters *INPUT* or *I*

Specifies the name of the file that contains the physical configuration definition subcommands to be edited. The default is $LOCAL.PHYSICAL_CONFIGURATION.

OUTPUT or *O*

Specifies the name of the file to receive the edited physical configuration subcommands. The default is $LOCAL.PHYSICAL_CONFIGURATION. If you enter QUIT FALSE to exit the utility, the output file is not rewritten.

ERRORS or *E*

Specifies the name of the file to which errors relating to DEFINE_ELEMENT subcommands on the input file are written. The default is $ERRORS.

STATUS

Returns the completion status for the EDIT_PHYSICAL_CONFIGURATION subutility.

Remarks

- When you enter the EDIT_PHYSICAL_CONFIGURATION subutility, the following prompt appears:

  ```
  PCE/
  ```

  Following the prompt, you can enter any EDIT_PHYSICAL_CONFIGURATION subcommands. Terminate the subutility by entering a QUIT subcommand.

- If you specify an input file for the EDIT_PHYSICAL_CONFIGURATION subcommand, any DEFINE_WORKING_MAINFRAME subcommands contained on the input file are interpreted and the mainframe name specified is substituted into subsequent DEFINE_ELEMENT subcommands until another DEFINE_WORKING_MAINFRAME subcommand is encountered. Once all DEFINE_ELEMENT subcommands contained on the input file have been interpreted, the EDIPC prompt appears. The default mainframe name for all EDIT_PHYSICAL_CONFIGURATION subutility subcommands entered interactively (in response to the prompt) is the name of the mainframe on which the PCU editor is executing.

- All element definitions from the input file are written to the output file in the same order as they appeared in the input file. Definitions deleted from the input file do not appear in the output file. Any element definitions added during the editing session are appended to the end of the configuration file in the order they were created.
- The output file is written in a stylized format. All the default values for components of the connection parameters in the DEFINE_ELEMENT subcommands are substituted and the values copied by the SAME_AS parameter are also substituted into the element definitions.
- Any SCL commands or control statements contained in the input file are interpreted but are not copied to the output file.
- Executing this subcommand from within the EDIT_PHYSICAL_CONFIGURATION subutility is not supported.
- This subutility does not perform any verification of the connections of the elements on which it operates. Use the VERIFY_PHYSICAL_CONFIGURATION subcommand for this purpose.

ADD_ELEMENT_DEFINITION Subcommand

Purpose Adds an element definition to the physical configuration file.

Format **ADD_ELEMENT_DEFINITION** or
ADDED
  ELEMENT=name
  SAME_AS=name
  ELEMENT_IDENTIFICATION=keyword
  IOU_PROGRAM_NAME=name
  SERIAL_NUMBER=integer
  STATE=keyword
  CENTRAL_MEMORY_CONNECTION=list of record
  IOU_CONNECTION=list of record
  PERIPHERAL_CONNECTION=list of record
  VERIFY_ELEMENT_IDENTIFICATION=boolean
  APPLICATION_INFORMATION=string
  SITE_INFORMATION=string
  STATUS=status variable

Parameters Refer to the DEFINE_ELEMENT subcommand for the parameter descriptions.

Remarks

- Element definitions added by this subcommand are inserted at the end of the output file in the order they were added.
- The mainframe field of the CENTRAL_MEMORY_CONNECTION and IOU_CONNECTION parameters defaults to the name of the executing mainframe.
- The IOU field of the IOU_CONNECTION parameter defaults to IOU0.

CHANGE_CONNECTION_REFERENCES Subcommand

Purpose Changes all defined connections to a particular mainframe, data channel, or peripheral. This subcommand facilitates the description of a recabling of a peripheral subsystem to a new mainframe, a different channel, or a different controller.

Format CHANGE_CONNECTION_REFERENCES or
CHANGE_CONNECTION_REFERENCE or
CHACR
OLD_CHANNEL_NAME=record
NEW_CHANNEL_NAME=record
OLD_MAINFRAME_NAME=record
NEW_MAINFRAME_NAME=record
OLD_PERIPHERAL_NAME=name
NEW_PERIPHERAL_NAME=name
STATUS=status variable

Parameters *OLD_CHANNEL_NAME* or *OCN*

Specifies a record identifying the channel that previously connected the subsystem to the mainframe. The record has the following format:

(**channel**, *mainframe*, *iou*)

channel

Specifies the name of the channel. This entry is required.

mainframe

Specifies the name of the mainframe to which the IOU is connected. The default value for the mainframe name is the name of the mainframe on which the subcommand is executed.

iou

Specifies the name of the IOU to which the channel is connected. The default value for the IOU field is IOU0.

NEW_CHANNEL_NAME or *NCN*

Specifies a record defining the channel that now connects the subsystem to the mainframe. This parameter is required if the OLD_CHANNEL_NAME parameter is specified. The record has the following format:

(**channel**, *mainframe*, *iou*)

channel

Specifies the name of the channel. This entry is required.

mainframe

Specifies the name of the mainframe to which the IOU is connected. The default value for the mainframe name is the name of the mainframe on which the subcommand is executed.

iou

Specifies the name of the IOU to which the channel is connected. The default value for the IOU field is IOU0.

OLD_MAINFRAME_NAME or *OMN*

Specifies a record identifying the mainframe and IOU previously connected to the peripheral(s). The record has the following format:

(**mainframe**, *iou*)

mainframe

Specifies the name of the mainframe to which the IOU is connected. This entry is required.

iou

Specifies the name of the IOU to which the channel is connected. The default value for the IOU field is IOU0.

NEW_MAINFRAME_NAME or *NMN*

Specifies a record identifying the mainframe to which the peripheral(s) are now connected. This parameter is required if the OLD_MAINFRAME_NAME parameter is specified. The record has the following format:

(**mainframe**, *iou*)

mainframe

Specifies the name of the mainframe to which the IOU is connected. This entry is required.

iou

Specifies the name of the IOU to which the channel is connected. The default value for the IOU field is IOU0.

OLD_PERIPHERAL_NAME or *OPN*

Specifies the name of the peripheral to which affected elements were previously connected.

NEW_PERIPHERAL_NAME or *NPN*

Specifies the name of the peripheral to which affected elements are now connected. This parameter is required if the OLD_PERIPHERAL_NAME parameter is specified.

STATUS

Returns the completion status for this subcommand.

Remarks

The mainframe and IOU values are not required for the OLD_CHANNEL_NAME and NEW_CHANNEL_NAME parameters. However, if a mainframe has more than one IOU, you must supply the IOU name to avoid ambiguity. The mainframe name defaults to the name of the mainframe on which the utility is currently executing. The IOU value defaults to IOU0. See the following examples.

Examples

- To recable all controllers in the physical configuration file from channel 1 on mainframe $SYSTEM_0855_0109 to channel 2 on the same mainframe, enter:

  ```
  chacr ocn=(ch1 $system_0855_0109) ..
    ncn=(ch2 $system_0855_0109)
  ```

  If $SYSTEM_0855_0109 is the only mainframe identified in the physical configuration, enter:

  ```
  chacr ocn=ch1 ncn=ch2
  ```

- To replace a mainframe with another mainframe while maintaining all the same channel assignments, enter:

  ```
  chacr omn=$system_0855_0109 ..
    nmn=$system_0855_0002
  ```

- To recable all controllers from channel 1 on $SYSTEM_0855_0109 to channel 2 on mainframe $SYSTEM_0855_0002, enter:

  ```
  chacr ocn=(ch1 $system_0855_0109) ..
    ncn=(ch2 $system_0855_0002)
  ```

  Note that the addresses of all controllers on channel 2 are assumed to be the same as when connected to channel 1.

- To recable all controllers from channel 1 IOU0 to channel 2 IOU1 on the same mainframe, enter:

  ```
  chacr ocn=(ch1 $system_0855_0109 IOU0) ..
    ncn=(ch2 $system_0855_0109 IOU1)
  ```

CHANGE_ELEMENT_DEFINITION Subcommand

Purpose

Modifies the definition of an element in the physical configuration file. Each parameter value specified for this subcommand replaces the value currently specified in the physical configuration file.

Format

CHANGE_ELEMENT_DEFINITION or
CHAED
  ELEMENT=name
  SAME_AS=name
  ELEMENT_IDENTIFICATION=keyword
  IOU_PROGRAM_NAME=name
  SERIAL_NUMBER=integer
  STATE=keyword
  CENTRAL_MEMORY_CONNECTION=list of record
  IOU_CONNECTION=list of record
  PERIPHERAL_CONNECTION=list of record
  VERIFY_ELEMENT_IDENTIFICATION=boolean
  APPLICATION_INFORMATION=string
  SITE_INFORMATION=string
  STATUS=status variable

Parameters

Refer to the DEFINE_ELEMENT subcommand for the parameter descriptions.

Remarks

- If a change is made to an element's peripheral connection, IOU connection, or central memory connection, the connection described completely replaces the element's previous connection description.
- The mainframe field of the CENTRAL_MEMORY_CONNECTION and IOU_CONNECTION parameters defaults to the name of the executing mainframe.

CHANGE_ELEMENT_NAME Subcommand

Purpose Changes the name of an element. This command can also change all references to the element (from the old name to the new name) where the element is named in the PERIPHERAL_CONNECTIONS parameter of other element definitions.

Format **CHANGE_ELEMENT_NAME** or
CHAEN
  ELEMENT=name
  NEW_ELEMENT_NAME=name
  CHANGE_REFERENCES=boolean
  STATUS=status variable

Parameters **ELEMENT** or **E**

Specifies the old name of the element to be changed. This parameter is required.

NEW_ELEMENT_NAME or **NEN**

Specifies the new name of the element. This parameter is required.

CHANGE_REFERENCES or *CHANGE_REFERENCE* or *CR*

Specifies whether all references to the element (in the PERIPHERAL_CONNECTIONS parameter of other element definitions) should be updated to reflect the new name. The default is TRUE.

TRUE

References to the element are updated to reflect the new name.

FALSE

References to the element remain unchanged.

STATUS

Returns the completion status for this subcommand.

DELETE_ELEMENT_DEFINITION Subcommand

Purpose Deletes an element definition from the physical configuration file.

Format **DELETE_ELEMENT_DEFINITION** or
DELED
  ELEMENT=keyword or list of name
  RETAIN=list of name
  STATUS=status variable

Parameters **ELEMENT** or **ELEMENTS** or **E**

Specifies the name or names of the elements to be deleted. You can also specify the keyword ALL. This parameter is required.

RETAIN or *R*

Specifies the name of one or more elements whose definition is not to be deleted. You can specify this parameter only if the value of the ELEMENT parameter is set to ALL. If you specify ALL, the definitions of all of the elements except those identified for this parameter are deleted. If you omit this parameter and you specified ELEMENT=ALL, the definitions of all elements are deleted.

STATUS

Returns the completion status for this subcommand.

Remarks This subcommand does not remove references to the deleted element from the element definitions of other elements in the file.

DISPLAY_CONNECTED_ELEMENTS Subcommand

Purpose Displays information on elements that are connected to a particular mainframe, IOU, channel, or peripheral element.

Format **DISPLAY_CONNECTED_ELEMENTS** or
DISPLAY_CONNECTED_ELEMENT or
DISCE
ELEMENT=name or *keyword*
CHANNEL=name
IOU=name
MAINFRAME=name
TYPE=keyword
DISPLAY_OPTIONS=list of keyword
OUTPUT=file
STATUS=status variable

Parameters *ELEMENT* or *E*

Specifies the element or elements to be displayed. The value specified can be the name of a peripheral element or one of the keywords listed below. If you specify a peripheral name for this parameter, the CHANNEL, IOU, and MAINFRAME parameters are ignored. The default is the keyword MAINFRAME.

CHANNEL or C

All elements connected to a specified channel are displayed. The channel is identified by the MAINFRAME, IOU, and CHANNEL parameters.

IOU

All elements connected to a specified IOU are displayed. The IOU is identified by the MAINFRAME and IOU parameters.

MAINFRAME or M

All elements connected to a specified mainframe are displayed. The mainframe is identified by the MAINFRAME parameter.

CHANNEL or *C*

Specifies the name of the mainframe channel for which connected elements are to be displayed. This parameter is required if you specified CHANNEL for the ELEMENT parameter; otherwise, this parameter is ignored.

IOU

Specifies the name of a CYBER 180 IOU whose connected elements are to be displayed. This parameter is required if you specified CHANNEL or IOU for the ELEMENT parameter; otherwise, this parameter is ignored.

The value specified for this parameter must have the following format:

IOUn

The n field is a decimal number.

MAINFRAME or *M*

Specifies the name of the mainframe whose connected elements are to be displayed. A mainframe name is required if you specify CHANNEL, IOU, or MAINFRAME for the ELEMENT parameter; otherwise, this parameter is ignored. If a mainframe name is required and you do not specify a name using this parameter, the system uses the name of the mainframe on which the PCU is currently executing.

TYPE or *T*

Specifies whether logical or physical element connections are to be displayed. You can specify the following keywords; the default is LOGICALLY_ACCESSIBLE:

LOGICALLY_ACCESSIBLE or LA

Displays only the elements that are logically connected to the element specified by the ELEMENT parameter (refer to the Remarks section for an explanation of what constititutes a logical connection).

PHYSICALLY_ACCESSIBLE or PA

Displays all elements that are physically connected to the element specified by the ELEMENT parameter.

DISPLAY_OPTIONS or *DO*

Specifies what information is to be displayed for each displayed element. You can specify the following keyword values for this parameter; the default is ALL:

ELEMENT_IDENTIFICATION or EI
IOU_PROGRAM_NAME or IPN
NAME or N
SERIAL_NUMBER or SN
STATE or S
VERIFY_ELEMENT_IDENTIFICATION or VEI
ALL

For a description of these values, see the parameter descriptions for the DEFINE_ELEMENT subcommand.

OUTPUT or *O*

Specifies the name of the file to receive the displayed information. The default is $OUTPUT.

STATUS

Returns the completion status for this subcommand.

Remarks

If you specify TYPE=LOGICALLY_ACCESSIBLE, the following rules determine which elements are displayed:

- If you specify a peripheral name for the ELEMENT parameter, only those elements in the ON or the DOWN state that are connected to the subject element are displayed.
- If you specify CHANNEL, IOU, or MAINFRAME for the ELEMENT parameter, a controller (that is, an element that has both IOU connections and peripheral connections) is displayed only if it is in the ON or DOWN state. A unit (that is, an element that has only peripheral connections) is displayed only if the unit itself is in an ON or DOWN state, and if at least one controller exists in an ON or DOWN state and is connected both to the unit and to the element specified by the ELEMENT parameter.

Examples

- The following subcommand displays information about all elements that are logically connected to hypothetical channel CH9:

```
PCE/disce e=channel c=ch9 t=la

Element Name            Product   State Serial number  VEI
============================================================
 GREEN_7165_1           $7165_22  ON    628            TRUE
 GREEN_895_0            $895_2    ON    1372           TRUE
 GREEN_895_1            $895_2    DOWN  1372           TRUE
 GREEN_895_2            $895_2    ON    1372           TRUE
```

- The following subcommand displays information about all elements that are physically connected to hypothetical channel CH9:

```
PCE/disce e=channel c=ch9 t=pa

Element Name            Product   State Serial number  VEI
============================================================
 GREEN_7165_1           $7165_22   ON   628            TRUE
 GREEN_895_0            $895_2     ON   1372           TRUE
 GREEN_895_1            $895_2     DOWN 1372           TRUE
 GREEN_895_2            $895_2     ON   1372           TRUE
 GREEN_895_3            $895_2     OFF  1372           TRUE
 GREEN_895_4            $895_2     OFF  330            TRUE
 GREEN_895_5            $895_2     OFF  330            TRUE
 GREEN_895_6            $895_2     OFF  330            TRUE
 GREEN_895_7            $895_2     OFF  330            TRUE
 VIOLET_7165_1          $7165_22   OFF  633            TRUE
 VIOLET_895_0           $895_2     OFF  1184           TRUE
 VIOLET_895_1           $895_2     OFF  1184           TRUE
 VIOLET_895_2           $895_2     OFF  1184           TRUE
 VIOLET_895_3           $895_2     OFF  1184           TRUE
 VIOLET_895_4           $895_2     OFF  1446           TRUE
 VIOLET_895_5           $895_2     OFF  1446           TRUE
 VIOLET_895_6           $895_2     OFF  1446           TRUE
 VIOLET_895_7           $895_2     OFF  1446           TRUE
```

- The following subcommand displays information about all elements that are physically connected to hypothetical element VIOLET_7165_1 (mass storage controller):

```
PCE/disce violet_7165_1 t=pa

Element Name              Product   State Serial number  VEI
=============================================================
 VIOLET_7165_1            $7165_22   ON   633             TRUE
 VIOLET_895_0             $895_2     ON   1184            TRUE
 VIOLET_895_1             $895_2     DOWN 1184            TRUE
 VIOLET_895_2             $895_2     ON   1184            TRUE
 VIOLET_895_3             $895_2     OFF  1184            TRUE
 VIOLET_895_4             $895_2     OFF  1446            TRUE
 VIOLET_895_5             $895_2     OFF  1446            TRUE
 VIOLET_895_6             $895_2     OFF  1446            TRUE
 VIOLET_895_7             $895_2     OFF  1446            TRUE
```

DISPLAY_ELEMENT_DEFINITIONS Subcommand

Purpose Displays the definitions of all or a subset of the elements defined in the physical configuration file.

Format **DISPLAY_ELEMENT_DEFINITIONS** or
DISPLAY_ELEMENT_DEFINITION or
DISED
ELEMENTS=list of name or *keyword*
OUTPUT=file
STATUS=status variable

Parameters *ELEMENTS* or *ELEMENT* or *E*

Specifies the names of elements or the classes of elements for which definitions are to be displayed. You select classes of elements using the following keywords. By default, all element definitions are displayed.

$CHANNEL_ADAPTER
$COMMUNICATIONS_ELEMENT
$CONTROLLER
$EXTERNAL_PROCESSOR
$STORAGE_DEVICE
ALL

OUTPUT or *O*

Specifies the output file to receive the listing of element definitions. The default is $OUTPUT.

STATUS

Returns the completion status for this subcommand.

Remarks

- This subcommand displays definitions of the elements of the working physical configuration file that is being edited within the EDIT_PHYSICAL_CONFIGURATION subutility.
- The APPLICATION_INFORMATION and SITE_INFORMATION fields of an element definition are displayed only for those elements for which one of these fields has been defined.

Examples The following subcommand displays definitions for hypothetical mass storage controller element VIOLET_7165_1:

```
PCE/dised e=violet_7165_1

DEFINE_ELEMENT E = VIOLET_7165_1 ..
  ELEMENT_IDENTIFICATION = $7165_22 STATE = OFF SERIAL_NUMBER = 633 ..
  IOU_CONNECTION = ( ..
      (  CH1  1 $SYSTEM_0860_0302 IOU0 )  ..
      (  CH3  1 $SYSTEM_0860_0302 IOU0 )  ..
      (  CH9  1 $SYSTEM_0855_0109 IOU0 )  ..
      (  CH27  1 $SYSTEM_0855_0109 IOU0 )) ..
  VERIFY_ELEMENT_IDENTIFICATION = TRUE
```

$ELEMENT_DEFINITION Function

Purpose

Returns a list of one or more records containing the definition data for one or more elements.

Format

$ELEMENT_DEFINITION
(element,
channel,
mainframe,
iou)

Parameters

element

Specifies the name of the element whose definition is to be returned. You can specify an element name or the keyword ALL. By default, definitions of all elements are returned as qualified by the other parameters.

channel

Specifies the name of a channel. Specifying a channel name restricts the returned definitions to those elements associated with that channel. The element definitions to be returned may be further restricted by the other parameters.

mainframe

Specifies the name of a mainframe. Specifying a mainframe name restricts the returned definitions to those elements associated with that mainframe. The element definitions to be returned may be further restricted by the other parameters (see Remarks). You can specify a mainframe name or the keyword ALL. Specifying ALL returns the definitions of all elements in the physical configuration. The default is the name of the executing mainframe.

iou

Specifies the name of an IOU. Specifying an IOU name restricts the returned definitions to those elements associated with that IOU. The element definitions to be returned may be further restricted by the other parameters (see Remarks). The default for the iou parameter is IOU0 if you specify the channel parameter.

Remarks

- The following qualifications apply to using this function's parameters:
  - If you specify the mainframe parameter and either the channel parameter or iou parameter, the returned element definitions are restricted to those elements directly or indirectly connected to the mainframe through the specified channel or IOU.
  - Specifying both the the channel parameter and the iou parameter qualifies the channel in a multiple IOU configuration.

- This function returns the following type:

```
"element_definition" list of record
  element: name
  application_information: string
  central_memory_connection: list of record
    port: integer 0..3
    mainframe: name
  recend
  element_identification: name 10
  iou_connection: list of record
    equipment: integer 0..7
    mainframe: name
  recend
  iou_program_name: name
  peripheral_connection: list of record
    peripheral_element: name
    physical_address: integer 0..65535
  recend
  serial_number: integer 1..999999
  site_information: string
  state: key
    on, off, down
  keyend
  verify_element_identification: boolean
recend
```

- If a particular element's definition does not include a specification for a field within the "element definition" type (see below), the field is defined using the SCL type UNSPECIFIED. The following element fields may be unspecified:

  central_memory_connection
  iou_program_name
  iou_connection
  peripheral_connection

QUIT Subcommand

Purpose Terminates processing of the EDIT_PHYSICAL_CONFIGURATION subutility.

Format

QUIT or
QUI
  WRITE_PHYSICAL_CONFIGURATION=boolean

Parameters *WRITE_PHYSICAL_CONFIGURATION* or *WPC*

Specifies a boolean value indicating whether the file named in the OUTPUT parameter of the EDIT_PHYSICAL_CONFIGURATION subcommand should be updated to reflect the changes made during the editing session. The default is TRUE.

TRUE

The file is updated.

FALSE

The file is not updated.

REPLACE_ELEMENT_DEFINITION Subcommand

Purpose Replaces an existing element definition in the physical configuration file with a new definition. As a result of this subcommand, the old element definition is completely deleted from the physical configuration file and is replaced by the new definition.

Format **REPLACE_ELEMENT_DEFINITION** or
REPED
- **ELEMENT=name**
- *SAME_AS=name*
- **ELEMENT_IDENTIFICATION=keyword**
- *IOU_PROGRAM_NAME=name*
- **SERIAL_NUMBER=integer**
- *STATE=keyword*
- *CENTRAL_MEMORY_CONNECTION=list of record*
- *IOU_CONNECTION=list of record*
- *PERIPHERAL_CONNECTION=list of record*
- *VERIFY_ELEMENT_IDENTIFICATION=boolean*
- *APPLICATION_INFORMATION=string*
- *SITE_INFORMATION=string*
- *STATUS=status variable*

Parameters Refer to the DEFINE_ELEMENT subcommand for the parameter descriptions.

Remarks The mainframe field of the CENTRAL_MEMORY_CONNECTION and IOU_CONNECTION parameters defaults to the name of the executing mainframe.

Configuration Examples

This section contains a number of examples of configuration files or configuration file segments. The examples show how various types of equipment might be defined in a configuration file. Each example consists of a pictorial representation of the configuration, followed by its corresponding configuration file segment.

Example 1: Dual-Mainframe Configuration

This example shows how two controllers can be cabled to two different mainframes. The mainframe names are $SYSTEM_0855_0002 and $SYSTEM_0855_0109. The configuration described by this configuration file is shown in figure 2-1. The following elements will be included in the physical configuration of either mainframe as a result of the installation of this configuration file:

ATS_BLUE	(7021-32 tape controller)
BLUE0	(697-7 tape storage unit)
BLUE1	(679-6 tape storage unit)
GREEN_DISC	(7155-14 disk controller)
GREEN0	(844-41 disk storage unit)
GREEN32	(885-11 disk storage unit)

Although elements BLUE0, BLUE1, GREEN0, and GREEN32 are in the physical configuration for both mainframes, they cannot be accessed by both mainframes concurrently. If these elements are in the ON state on $SYSTEM_0855_0002, they must be in the OFF state of $SYSTEM_0855_0109. In this example, the tape controller definitions do not specify the IOU_PROGRAM_NAME parameter. Allowing this parameter to default means that data transfers to and from the tape units are limited to a maximum block size of 4,128 bytes.

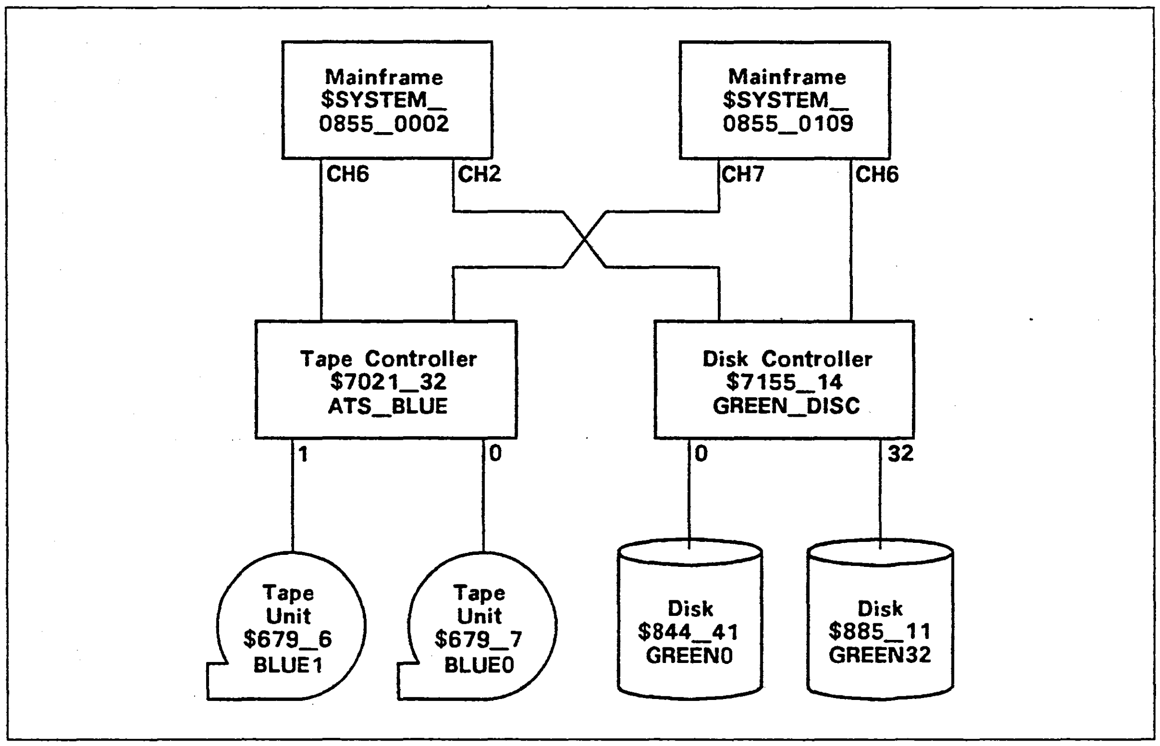

Mainframe $SYSTEM_ 0855_0002
Mainframe $SYSTEM_ 0855_0109
CH6
CH2
CH7
CH6
Tape Controller $7021_32 ATS_BLUE
Disk Controller $7155_14 GREEN_DISC
1
0
0
32
Tape Unit $679_6 BLUE1
Tape Unit $679_7 BLUE0
Disk $844_41 GREEN0
Disk $885_11 GREEN32

Figure 2-1. Dual-Mainframe Configuration Example

```
define_working_mainframe $SYSTEM_0855_0002
define_element element=ats_blue ..
               element_identification=$7021_32 ..
               serial_number=0001 ..
               iou_connection=(ch6, ..
                              (ch7,0,$SYSTEM_0855_0109))
define_element element=blue0 ..
               element_identification=$679_7 ..
               serial_number=2345 ..
               peripheral_connection=((ats_blue,0))
define_element element=blue1 ..
               element_identification=$679_6 ..
               serial_number=1234 ..
               peripheral_connection=((ats_blue,1))
define_element element=green_disc ..
               element_identification=$7155_14 ..
               serial_number=4321 ..
               iou_connection=(ch2, ..
                              (ch6,0,$SYSTEM_0855_0109))
define_element element=green32 ..
               element_identification=$885_11 ..
               serial_number=1222 ..
               peripheral_connection=((green_disc,32))
define_element element=green0 ..
               element_identification=$844_41 ..
               serial_number=1576 ..
               peripheral_connection=((green_disc,0))
```

This page intentionally left blank.

Example 2: 836-441 Configuration

This example shows part of a configuration file that defines an 836-441 disk subsystem for a CYBER model 810 or 830 mainframe. Figure 2-2 illustrates this configuration.

There are four disk units and four controllers (control modules) per $836_441 product. The controllers are dual-ported but only one NOS/VE channel is used per instance of deadstart. The disk units are single-ported. A maximum of eight 836 disk units may be connected to the same NOS/VE channel.

On the model 810 and 830 mainframes, only channels 0, 1, 3, 6, 16, and 22 can be connected to a $FA7B4_D control module.

The configuration defined in this example shows one cabinet containing four $FA7B4_D controllers and four $836_441 disk units. Although all four controllers are connected to both channels 0 and 1, the red controllers are normally accessed only by channel 0, and the blue controllers are normally accessed only by channel 1.

NOS/VE only uses one channel to a $FA7B4_D controller per instance of deadstart. The channel used is the first one defined in the IOU_CONNECTION parameter for the controller that is in the ON state. Therefore, if it becomes necessary to switch a controller to its redundant channel access, you must:

1. Edit the physical configuration file so that the desired connection is described first.

2. Redeadstart the system to install the modified physical configuration.

In this example, the unit number is omitted from all PERIPHERAL_CONNECTION parameter specifications. All unit numbers are 0, which is the default value for this specification.

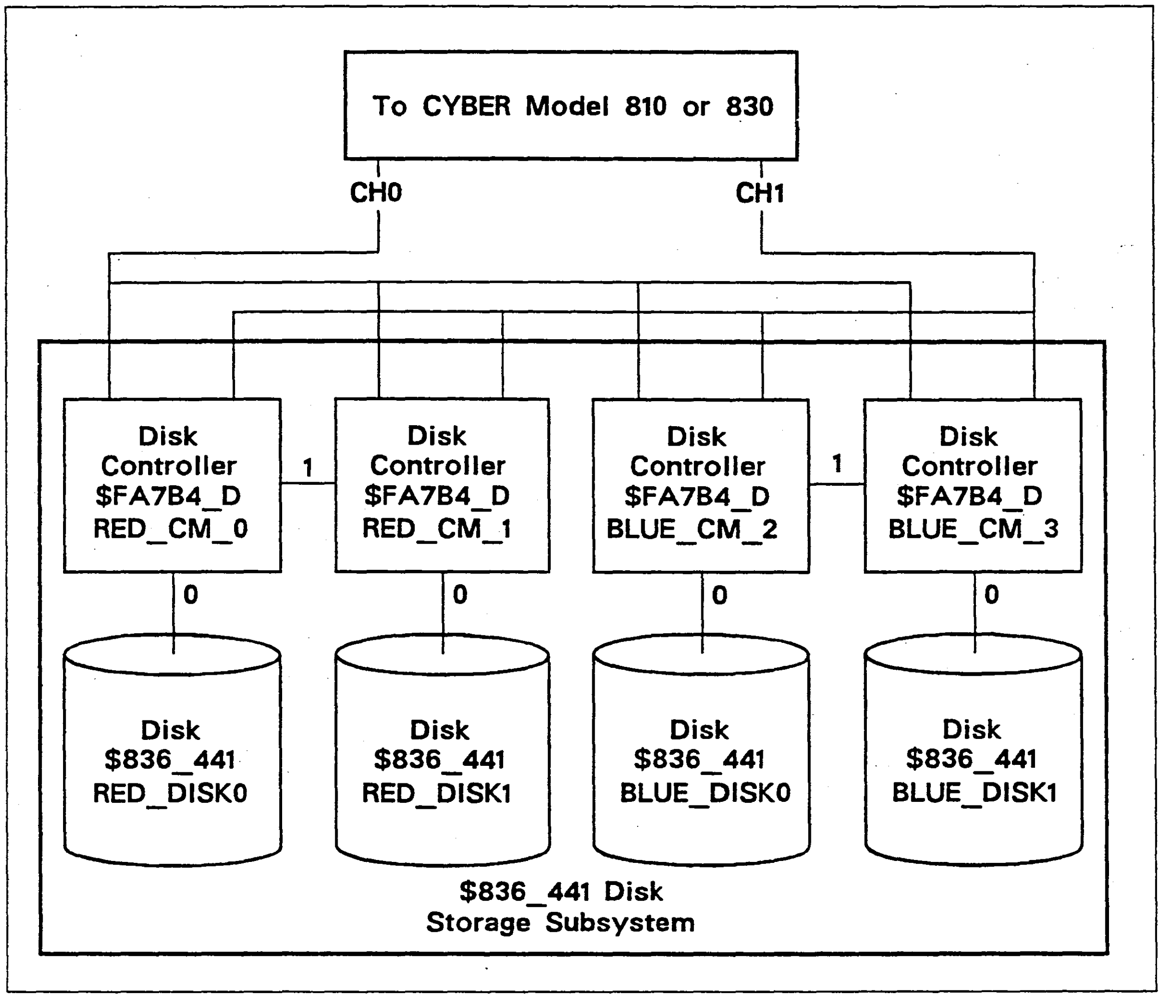

To CYBER Model 810 or 830
CH0
CH1
Disk Controller $FA7B4_D RED_CM_0
Disk Controller $FA7B4_D RED_CM_1
Disk Controller $FA7B4_D BLUE_CM_2
Disk Controller $FA7B4_D BLUE_CM_3
1
1
0
0
0
0
Disk $836_441 RED_DISK0
Disk $836_441 RED_DISK1
Disk $836_441 BLUE_DISK0
Disk $836_441 BLUE_DISK1
$836_441 Disk Storage Subsystem

Figure 2-2. 836-441 Configuration Example

```
"Define each controller.

define_element element=red_cm_0 ..
               element_identification=$FA7B4_D ..
               serial_number=1056 ..
               iou_connection=((ch0 0),(ch1 0))

define_element element=red_cm_1 ..
               element_identification=$FA7B4_D ..
               serial_number=1057 ..
               iou_connection=((ch0 1),(ch1 1))

define_element element=blue_cm_2 ..
               element_identification=$FA7B4_D ..
               serial_number=1058 ..
               iou_connection=((ch1 2),(ch0 2))

define_element element=blue_cm_3 ..
               element_identification=$FA7B4_D ..
               serial_number=1059 ..
               iou_connection=((ch1 3),(ch0 3))

"Define each disk unit.

define_element element=red_disk0 ..
               element_identification=$836_441 ..
               serial_number=2155 ..
               peripheral_connection=red_cm_0

define_element element=red_disk1 ..
               element_identification=$836_441 ..
               serial_number=2167 ..
               peripheral_connection=red_cm_1

define_element element=blue_disk0 ..
               element_identification=$836_441 ..
               serial_number=2324 ..
               peripheral_connection=blue_cm_2

define_element element=blue_disk1 ..
               element_identification=$836_441 ..
               serial_number=2567 ..
               peripheral_connection=blue_cm_3
```

This page intentionally left blank.

Example 3: 7165-21/895-2 Configuration

This example shows a segment of the physical configuration file that defines a 7165-21 disk subsystem. This configuration is shown in figure 2-3.

The 7165-21 has four independent controllers: two CYBER channel couplers (CCCs) and two storage directors (SDs). Only one CCC may be connected to a given CYBER channel. The address specified in the IOU_CONNECTION parameter is the address from the CCC to the SD which, in this example, is 0 in both cases (the CCC itself is not defined as an element in the physical configuration; its presence is assumed by the system).

An 895-1 dual disk controller (DDC) connects from 4 to 16 disk units (the $895_2's) to a 7165 controller. The 895-1 is not an element defined in the physical configuration (that is, it cannot be defined by a DEFINE_ELEMENT subcommand). The address of the DDC is implicitly specified as part of the disk unit address according to the following equation:

diskaddress = (16 * ddc) + unitnumber

The ddc value is the address of the DDC. The unitnumber value is the unit number of the disk unit.

In this example, the address of the DDC is zero (0).

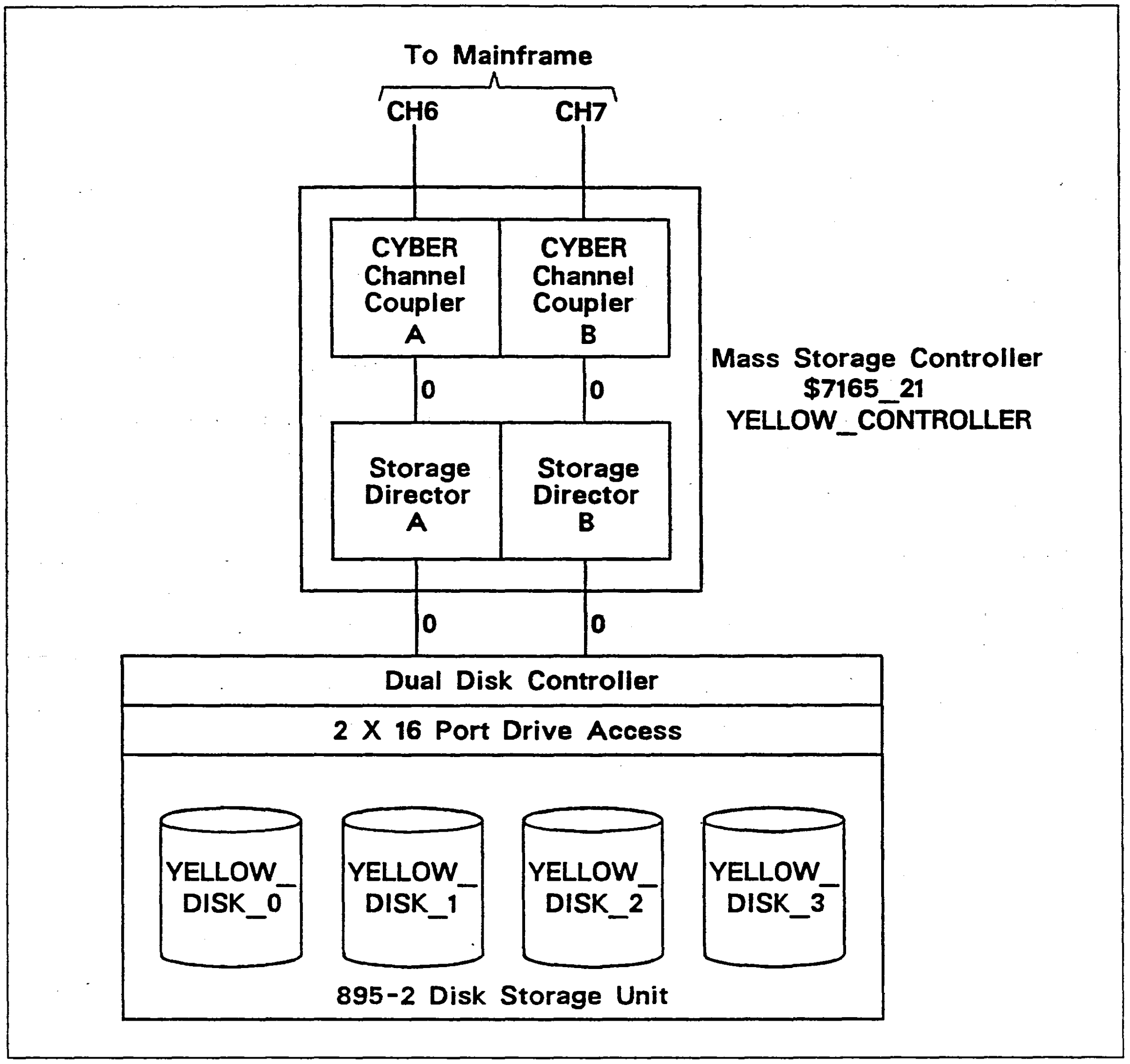


Figure 2-3. 7165-21/895-2 Configuration Example

```
"Define the 7165-21 disk controller.

define_element element=yellow_controller ..
               element_identification=$7165_21 ..
               serial_number=0001 ..
               iou_connection=((ch6 0),(ch7 0))

"Define the 895-2 disk units.

define_element element=yellow_disk_0 ..
               element_identification=$895_2 ..
               serial_number=1234 ..
               peripheral_connection=yellow_controller

define_element element=yellow_disk_1 ..
               same_as=yellow_disk_0 ..
               peripheral_connection=((yellow_controller,1))

define_element element=yellow_disk_2 ..
               same_as=yellow_disk_0 ..
               peripheral_connection=((yellow_controller,2))

define_element element=yellow_disk_3 ..
               same_as=yellow_disk_0 ..
               peripheral_connection=((yellow_controller,3))
```

Example 4: 7165-22/895-2 Configuration

This example defines two 7165-22 disk controllers with attached 895-2 disk units. The 7165-22 disk controller consists of two CYBER channel couplers (CCCs) and two dual-ported storage directors (SDs). Figure 2-4 shows the configuration of the CCCs, SDs, and disk units. The only difference between a 7165-21 and a 7165-22 is that the latter has an additional port on each SD, which allows two 7165 controllers to access the same disk units.

The sample configuration illustrates how to provide redundant access to a group of disk units from two different mainframes. Both controllers could also have been connected to the same mainframe. Note that NOS/VE does not allow both mainframes to access the disk units concurrently. Each 7165-22 controller can have up to two actively configured channels. Both channels must belong to the same mainframe.

In this example, the violet 895 disk units are connected to a dual disk controller (DDC) whose address is 1.

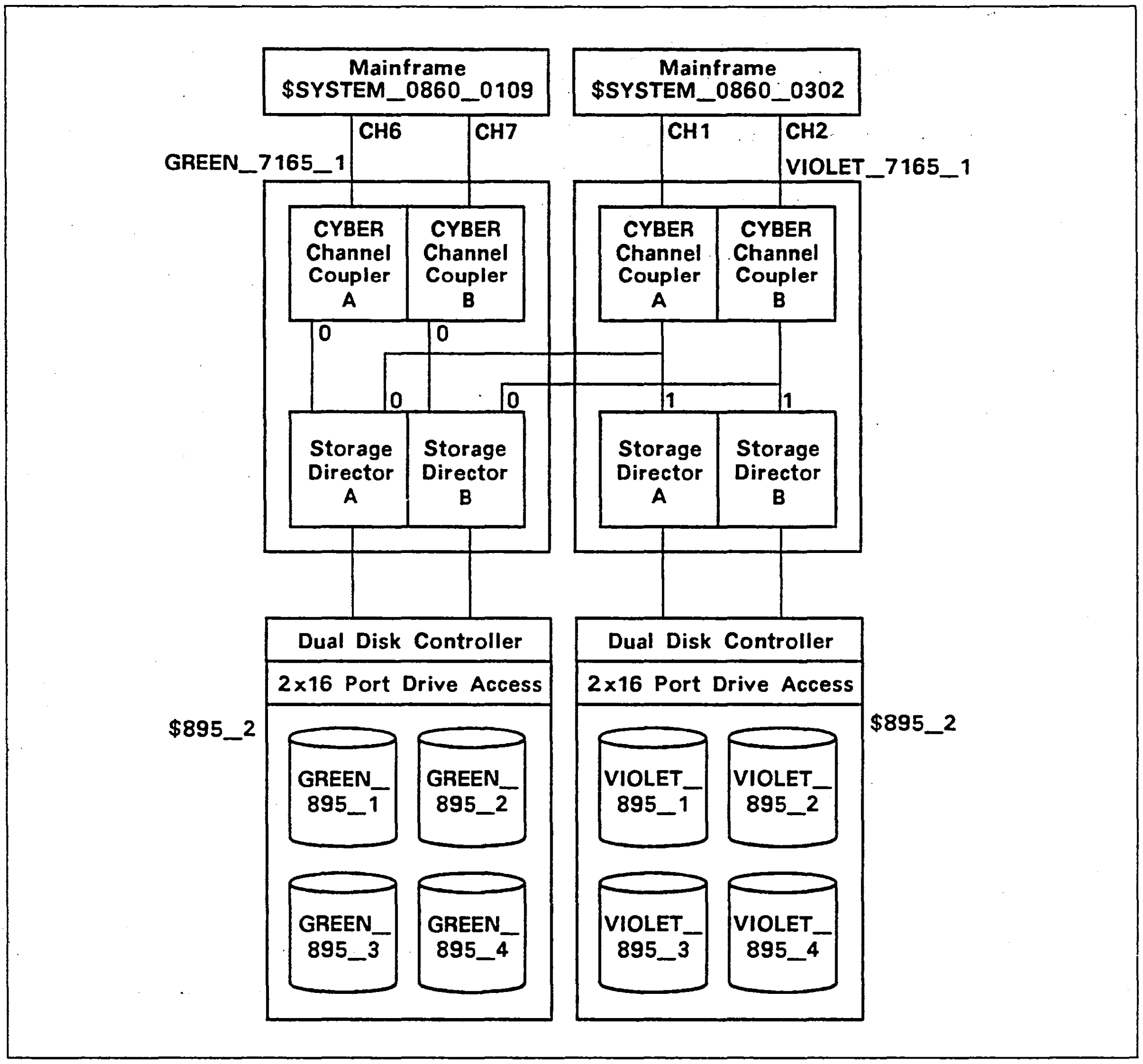


Figure 2-4. 7165-22/895-2 Configuration Example

```
define_element element=green_7165_1 ..
               element_identification=$7165_22 ..
               serial_number=634 ..
               iou_connection=( ..
                 (ch6 0 $system_0860_0109) ..
                 (ch7 0 $system_0860_0109) ..
                 (ch1 0 $system_0860_0302) ..
                 (ch2 0 $system_0860_0302))

define_element element=violet_7165_1 ..
               element_identification=$7165_21 ..
               serial_number=588 ..
               iou_connection=( ..
                 (ch1 1 $system_0860_0302) ..
                 (ch2 1 $system_0860_0302))

define_element element=green_895_1 ..
               element_identification=$895_2 ..
               serial_number=1249 ..
               peripheral_connection=( ..
                 (green_7165_1 0))

define_element element=green_895_2 ..
               element_identification=$895_2 ..
               serial_number=1249 ..
               peripheral_connection=( ..
                 (green_7165_1 1))

define_element element=green_895_3 ..
               element_identification=$895_2 ..
               serial_number=1249 ..
               peripheral_connection=( ..
                 (green_7165_1 2))

define_element element=green_895_4 ..
               element_identification=$895_2 ..
               serial_number=1249 ..
               peripheral_connection=( ..
                 (green_7165_1 3))

define_element element=violet_895_1 ..
               element_identification=$895_2 ..
               serial_number=1249 ..
               peripheral_connection=( ..
                 (violet_7165_1 16))
```

```
define_element element=violet_895_2 ..
               element_identification=$895_2 ..
               serial_number=1249 ..
               peripheral_connection=( ..
                 (violet_7165_1 17))

define_element element=violet_895_3 ..
               element_identification=$895_2 ..
               serial_number=1249 ..
               peripheral_connection=( ..
                 (violet_7165_1 18))

define_element element=violet_895_4 ..
               element_identification=$895_2 ..
               serial_number=1249 ..
               peripheral_connection=( ..
                 (violet_7165_1 19))
```

This page intentionally left blank.

Example 5: 834-12/10395-11 Daisy-Chain Configuration

This example is part of a configuration file showing the interconnection of $834_12 disk units, $10395_11 control modules, and a CYBER 810 or 830 mainframe. The example shows two disk units attached to each controller, but up to four $834_12 units can be attached to a single controller. The configuration described in this example is shown in figure 2-5.

On the model 810 and 830 mainframes, only channels 0, 1, 3, 6, 16 and 22 can be connected to a $10395_11 control module.

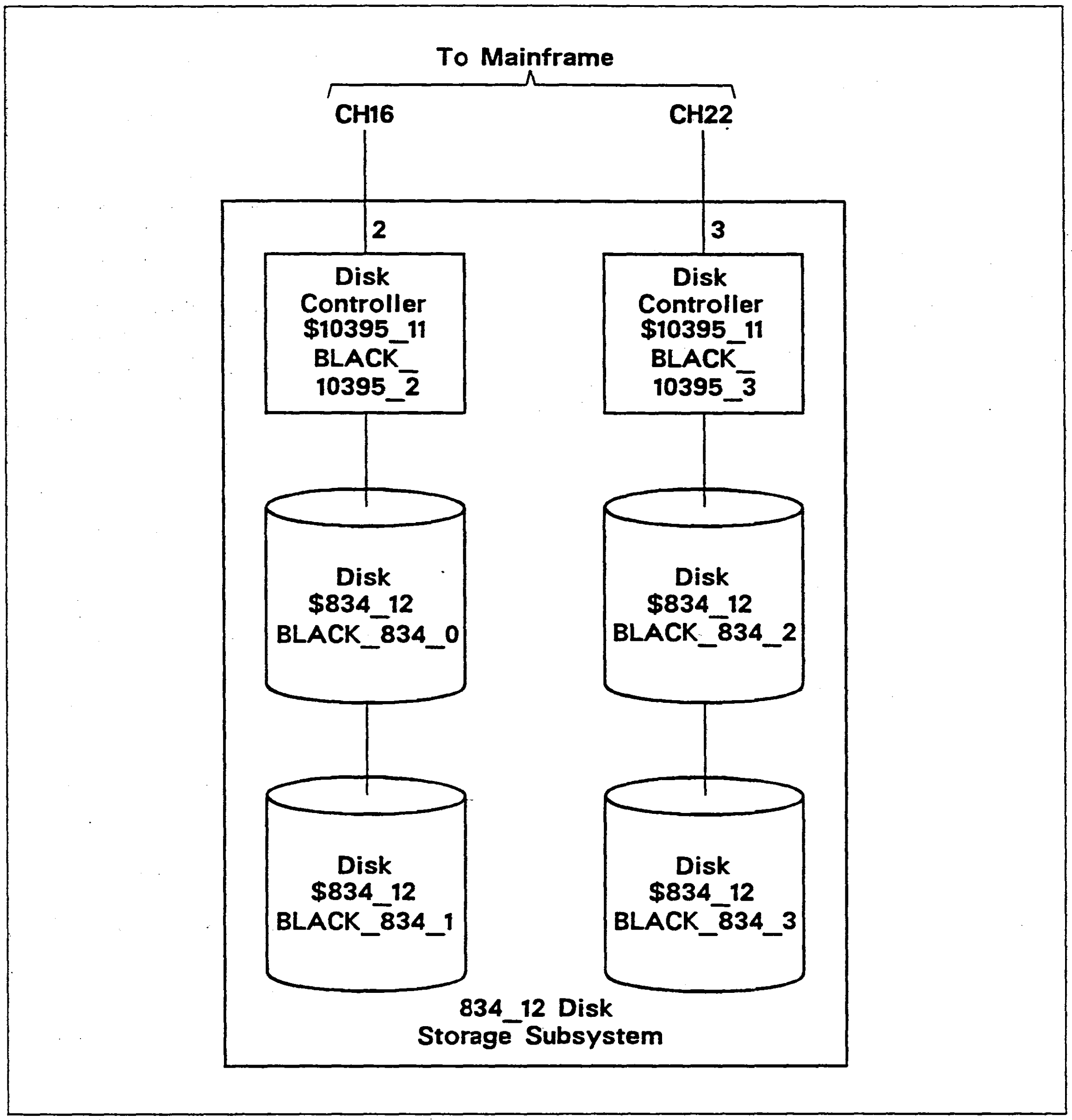


Figure 2-5. 834-12/10395-11 Daisy-Chain Configuration Example

```
"Define each controller.

define_element element=black_10395_2 ..
               element_identification=$10395_11 ..
               serial_number=579 ..
               iou_connection=((ch16 2))

define_element element=black_10395_3 ..
               element_identification=$10395_11 ..
               serial_number=580 ..
               iou_connection=((ch22 3))

"Define each disk unit.

define_element element=black_834_0 ..
               element_identification=$834_12 ..
               serial_number=110 ..
               peripheral_connection=((black_10395_2 0))

define_element element=black_834_1 ..
               element_identification=$834_12 ..
               serial_number=111 ..
               peripheral_connection=((black_10395_2 1))

define_element element=black_834_2 ..
               element_identification=$834_12 ..
               serial_number=112 ..
               peripheral_connection=((black_10395_3 2))

define_element element=black_834_3 ..
               element_identification=$834_12 ..
               serial_number=113 ..
               peripheral_connection=((black_10395_3 3))
```

This page intentionally left blank.

Example 6: 887-1 Configuration

This example shows part of a physical configuration file in which 887-1 disk units are attached in a string or daisy-chain formation to an I4 IOU. (See table E-4, Characteristics of CYBER 180 IOUs, for a list of the CYBER mainframes that currently support the I4 IOU.)

The IOU configuration of these mainframes uses 10 concurrent DMA (direct memory access) channels and PPs divided into two clusters of five channels. The PPs and channels are numbered 0 to 9. A PP in one cluster cannot access a channel in another cluster, but two channels in different clusters can access the same 887-1 disk subsystem.

A concurrent DMA channel is a channel that allows the data to bypass PP memory and transfer directly to or from the mainframe central memory while the PP program does other processing. Each concurrent data channel has two ports that may be configured; these are referred to as ports A and B.

There are two disk units per 887-1 product. Up to eight disk units may be connected in a string to any one channel port. The two disk units in the same product (and cabinet) must both be connected to the same string or strings.

Two strings of disk units may be connected to the same concurrent channel, one string on each port. However, only one port can be used at a time; NOS/VE processes requests for both ports in an alternating fashion.

The following example defines two strings of 887-1 disk units. The blue string allows access from two different IOU channel/PP clusters. The red string is accessed by a single channel. Figure 2-6 shows the sample configuration.

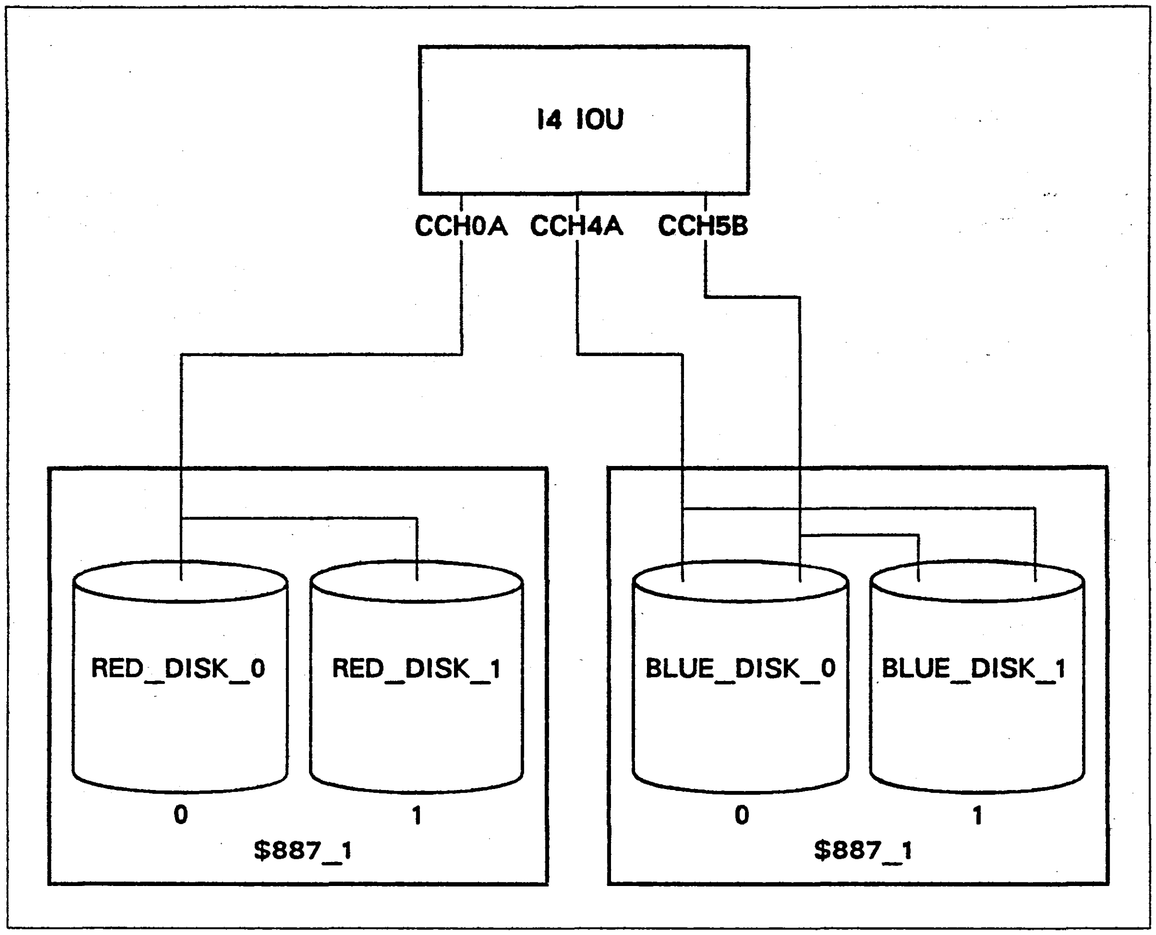


Figure 2-6. 887-1 Configuration Example

```
define_element element=red_disk_0 ..
               element_identification=$887_1 ..
               serial_number=1234 ..
               iou_connection=cch0a

define_element element=red_disk_1 ..
               element_identification=$887_1 ..
               serial_number=1235 ..
               iou_connection=((cch0a,1))

define_element element=blue_disk_0 ..
               element_identification=$887_1 ..
               serial_number=1236 ..
               iou_connection=((cch4a,0),(cch5b,0))

define_element element=blue_disk_1 ..
               element_identification=$887_1 ..
               serial_number=1237 ..
               iou_connection=((cch4a,1),(cch5b,1))
```

Example 7: 9836-1 Configuration

This example shows part of a physical configuration file that defines a $9836_1 disk subsystem for a CYBER 930-series system.

Each subsystem has four $9836_1 disk units and two $FA7B5_A controllers per subsystem. The controllers are dual-ported, but only one NOS/VE channel per controller is activated. The disk units are also dual-ported; however, only one controller is used to access a particular disk unit.

On CYBER 930-series systems, only channels 1, 3, 5, 17, 19, and 21 can be connected to a $FA7B5_A controller. However, hardware options are available that allow even-numbered channels to be connected to a $FA7B5_A controller.

In this sample configuration, one cabinet contains two controllers and four disk units. Two cabinets are shown. In the figure, the bold lines represent active connections, while the normal lines represent inactive (redundant) connections.

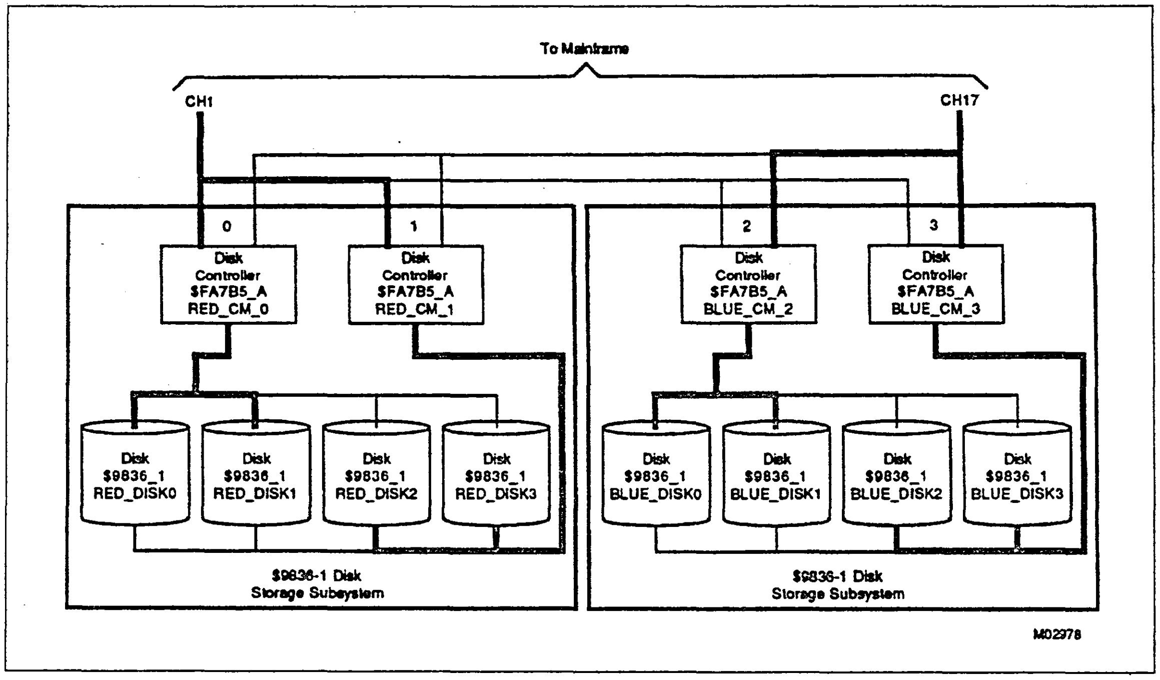


Figure 2-7. 9836-1 Configuration Example

```
"Define each controller.

define_element element=red_cm_0 ..
               element_identification=$FA7B5_A ..
               serial_number=1056 ..
               iou_connection= ((ch1 0) (ch17 0))

define_element element=red_cm_1 ..
               element_identification=$FA7B5_A ..
               serial_number=1057 ..
               iou_connection= ((ch1 1) (ch17 1))

define_element element=blue_cm_2 ..
               element_identification=$FA7B5_A ..
               serial_number=1058 ..
               iou_connection= ((ch17 2) (ch1 2))

define_element element=blue_cm_3 ..
               element_identification=$FA7B5_A ..
               serial_number=1059 ..
               iou_connection= ((ch17 3) (ch1 3))

"Define each disk unit.

define_element element=red_disk0 ..
               element_identification=$9836_1 ..
               serial_number=2155 ..
               peripheral_connection= ((red_cm_0 0) (red_cm_1 0))

define_element element=red_disk1 ..
               element_identification=$9836_1 ..
               serial_number=2156 ..
               peripheral_connection= ((red_cm_0 1) (red_cm_1 1))

define_element element=red_disk2 ..
               element_identification=$9836_1 ..
               serial_number=2157 ..
               peripheral_connection= ((red_cm_1 2) (red_cm_0 2))

define_element element=red_disk3 ..
               element_identification=$9836_1 ..
               serial_number=2158 ..
               peripheral_connection= ((red_cm_1 3) (red_cm_0 3))

define_element element=blue_disk0 ..
               element_identification=$9836_1 ..
               serial_number=2159 ..
               peripheral_connection= ((blue_cm_2 0) (blue_cm_3 0))

define_element element=blue_disk1 ..
               element_identification=$9836_1 ..
               serial_number=2160 ..
               peripheral_connection= ((blue_cm_2 1) (blue_cm_3 1))
```

```
define_element element=blue_disk2 ..
               element_identification=$9836_1 ..
               serial_number=2161 ..
               peripheral_connection= ((blue_cm_3 2) (blue_cm_2 2))

define_element element=blue_disk3 ..
               element_identification=$9836_1 ..
               serial_number=2162 ..
               peripheral_connection= ((blue_cm_3 3) (blue_cm_2 3))
```

Example 8: 9853-3 Configuration

This example shows part of a physical configuration file that defines a $9853_3 disk subsystem for a CYBER 930-series system.

Each subsystem has four $9853_3 disk units and two $FA7B5_A controllers. The controllers are dual-ported, but only one NOS/VE channel per controller is activated. The disk units are also dual-ported; however, only one controller is used to access a particular disk unit.

On CYBER 930-series systems, only channels 1, 3, 5, 17, 19, and 21 can be connected to a $FA7B5_A controller. However, hardware options are available that allow even-numbered channels to be connected to a $FA7B5_A controller.

On an I4 IOU, only CIO IPI channels can be connected to the $FA7B5_A controller.

In this sample configuration, one cabinet contains two controllers and four disk units. Two cabinets are shown. In the figure, the bold lines represent active connections, while the normal lines represent inactive (redundant) connections.

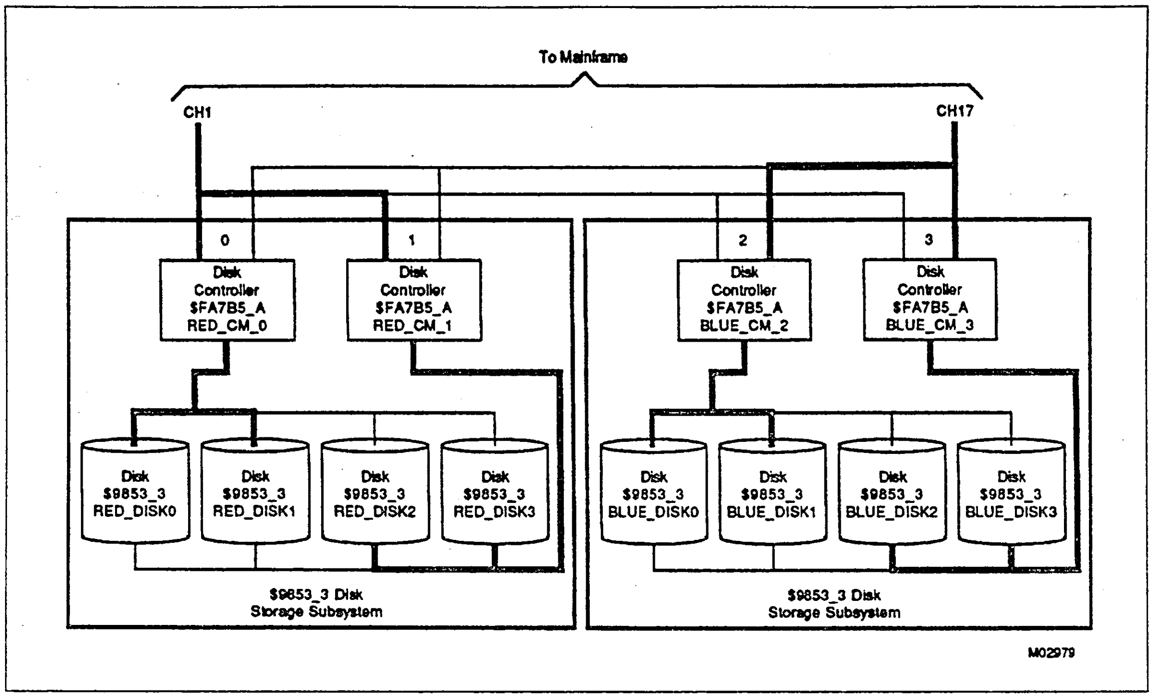


Figure 2-8. 9853-3 Configuration Example

```
"Define each controller.

define_element element=red_cm_0 ..
               element_identification=$FA7B5_A ..
               serial_number=1056 ..
               iou_connection= ((ch1 0) (ch17 0))

define_element element=red_cm_1 ..
               element_identification=$FA7B5_A ..
               serial_number=1057 ..
               iou_connection= ((ch1 1) (ch17 1))

define_element element=blue_cm_2 ..
               element_identification=$FA7B5_A ..
               serial_number=1058 ..
               iou_connection= ((ch17 2) (ch1 2))

define_element element=blue_cm_3 ..
               element_identification=$FA7B5_A ..
               serial_number=1059 ..
               iou_connection= ((ch17 3) (ch1 3))

"Define each disk unit.

define_element element=red_disk0 ..
               element_identification=$9853_x ..
               serial_number=2155 ..
               peripheral_connection= ((red_cm_0 0) (red_cm_1 0))

define_element element=red_disk1 ..
               element_identification=$9853_x ..
               serial_number=2156 ..
               peripheral_connection= ((red_cm_0 1) (red_cm_1 1))

define_element element=red_disk2 ..
               element_identification=$9853_x ..
               serial_number=2157 ..
               peripheral_connection= ((red_cm_1 2) (red_cm_0 2))

define_element element=red_disk3 ..
               element_identification=$9853_x ..
               serial_number=2158 ..
               peripheral_connection= ((red_cm_1 3) (red_cm_0 3))

define_element element=blue_disk0 ..
               element_identification=$9853_x ..
               serial_number=2159 ..
               peripheral_connection= ((blue_cm_2 0) (blue_cm_3 0))

define_element element=blue_disk1 ..
               element_identification=$9853_x ..
               serial_number=2160 ..
               peripheral_connection= ((blue_cm_2 1) (blue_cm_3 1))
```

```
define_element element=blue_disk2 ..
               element_identification=$9853_x ..
               serial_number=2161 ..
               peripheral_connection= ((blue_cm_3 2) (blue_cm_2 2))

define_element element=blue_disk3 ..
               element_identification=$9853_x ..
               serial_number=2162 ..
               peripheral_connection= ((blue_cm_3 3) (blue_cm_2 3))
```

This page intentionally left blank.

Examples 9 to 13 illustrate configurations involving $5698_1x tape controllers and $698_3x tape units. Table 2-2 summarizes the interconnections among them.

Table 2-2. Analysis of $5698 Tape Subsystem Configurations

Analysis	9[1]	10[1]	11[1]	12[1]	13[1]
Only one tape unit may transfer data at a time.	No	No	No	Yes[2]	No
Any two tape units may transfer data concurrently.	Yes[3]	No	Yes[3]	No	No
If the operator is careful, two tape units may transfer data concurrently.	NA	Yes[4]	NA	No	Yes[4]
If the operator is careful, four tape units may transfer data concurrently.	NA	No	Yes[5]	No	No
Number of host channels active:[6]	2	1	2	1	1 or 2[7]
Number of host channels physically cabled:	2	1	2	2	2
Each tape unit has alternate or redundant controller access for reconfiguration.	Yes[8]	No	Yes[8]	No	No
Each controller has redundant channel access for reconfiguration.	No[9]	No	No[9]	Yes[10]	Yes[10]
Maximum number of tape controllers in subsystem:[11]	16	8	16	8	8
Maximum number of tape units in subsystem:[12]	64	64	64	64	64

The following notes apply to table 2-2:

1. The numbers in the column headings refer to examples 9 to 13.
2. There is only one tape controller. A controller is required for each unit that is to transfer data concurrently.
3. These configurations provide alternate access and the greatest flexibility for tape mounting by the operator.
4. For best performance, the operator must assign tapes equally among the two controllers. The performance of the configuration shown in example 9 is equivalent to that of the configuration shown in example 10.
5. For best performance, the operator must assign tapes equally among the two pairs of controllers.
6. The word *active* implies that the channel is set to ON and a PP is assigned.
7. See example 13 for an explanation of why one or two channels may be configured.
8. Any single channel or controller may fail without denying access to a unit.
9. The existence of an alternate access to the tape units in examples 9 and 11 allows for reconfiguration for channel or controller failure.
10. Any single channel may fail without denying access to the controller.
11. The maximum number of controllers is derived by extending the configuration shown in the corresponding example to the fullest degree possible under NOS/VE. Two strings of controllers are considered for the alternate access configurations. Otherwise, only one string of controllers is considered.
12. Examples 9 and 11 have alternate access. Therefore, each pair of controllers can be connected to a maximum of eight tape units.

This page intentionally left blank.

Example 9: 5698-1x Alternate-Access Configuration

The example illustrated in figure 2-9 is part of a configuration file showing the interconnection of $5698_1x tape controllers and $698_3x tape units in an alternate-access configuration. This means that, at any one instant, two tape units can be accessed, one per controller. Either controller can be used to transfer data to or from a particular tape unit; however, only one controller can access a particular tape unit at any one time.

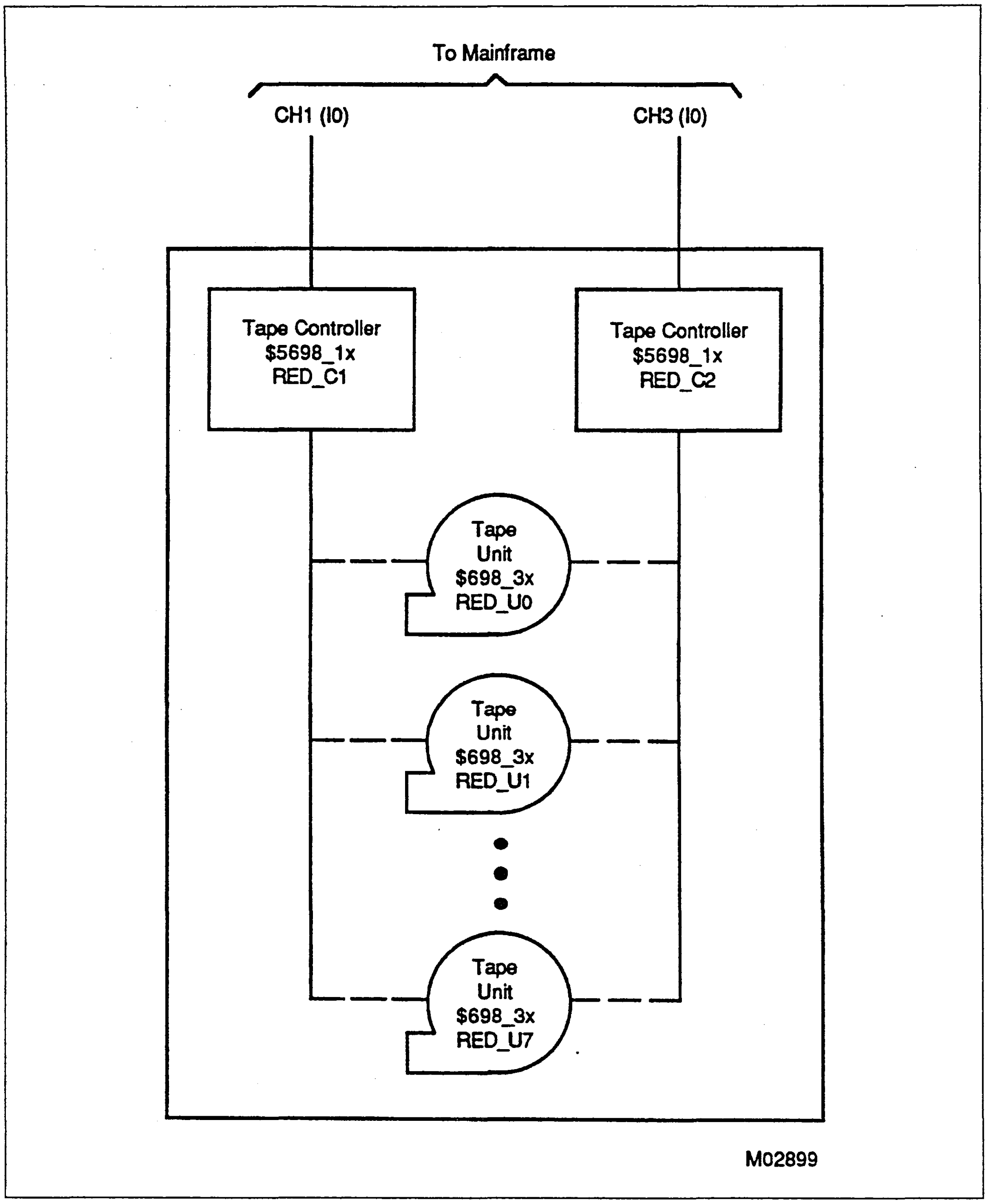


Figure 2-9. 5698-1x Alternate-Access Configuration Example

```
"Define each controller.

define_element element=red_c1 ..
               element_identification=$5698_1x ..
               serial_number=991 ..
               iou_connection=((ch1 0))

define_element element=red_c2 ..
               element_identification=$5698_1x ..
               serial_number=992 ..
               iou_connection=((ch3 0))

"Define each tape unit.

define_element element=red_u0 ..
               element_identification=$698_3x ..
               serial_number=2010 ..
               peripheral_connection=((red_c1 0) ..
                                      (red_c2 0))

define_element element=red_u1 ..
               element_identification=$698_3x ..
               serial_number=2011 ..
               peripheral_connection=((red_c1 1) ..
                                      (red_c2 1))
      .
      .
      .
define_element element=red_u7 ..
               element_identification=$698_3x ..
               serial_number=2017 ..
               peripheral_connection=((red_c1 7) ..
                                      (red_c2 7))
```

Example 10: 5698-1x Daisy-Chain Configuration

The example illustrated in figure 2-10 is part of a configuration file showing the interconnection of $5698_1x tape controllers and $698_3x tape units. The configuration shown supports a daisy-chain connection. In this case, a string of tape units is configured to a single controller. The address of each controller on the string must be unique.

Although both concurrent and nonconcurrent channels are shown in the figure, the accompanying coding reflects only the nonconcurrent channel (CH1). A concurrent channel may be used on an I4 IOU.

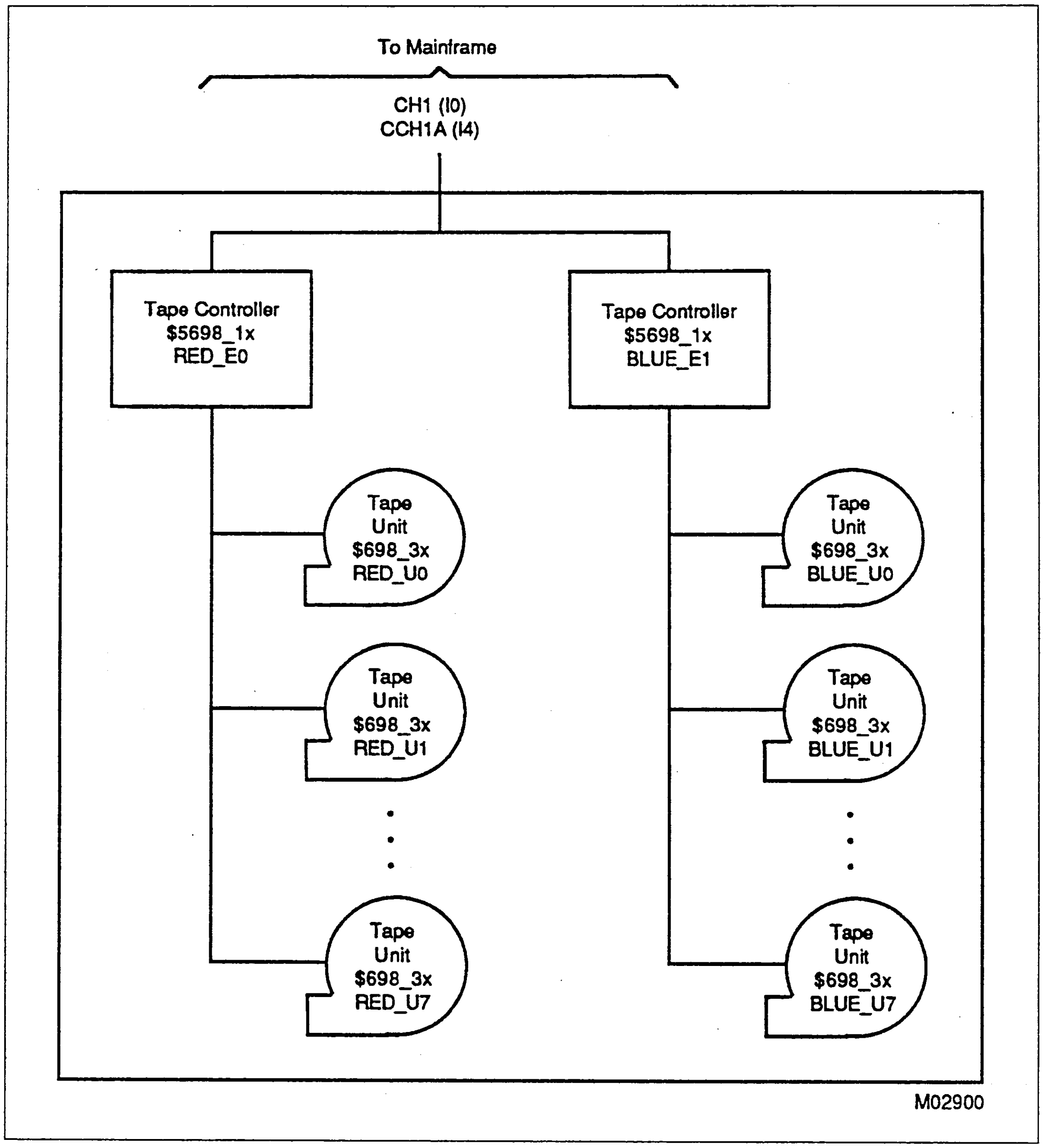

To Mainframe
CH1 (I0)
CCH1A (I4)
Tape Controller $5698_1x RED_E0
Tape Controller $5698_1x BLUE_E1
Tape Unit $698_3x RED_U0
Tape Unit $698_3x RED_U1
Tape Unit $698_3x RED_U7
Tape Unit $698_3x BLUE_U0
Tape Unit $698_3x BLUE_U1
Tape Unit $698_3x BLUE_U7
M02900

Figure 2-10. 5698-1x Daisy-Chain Configuration Example

```
"Define each controller.

define_element element=red_e0 ..
               element_identification=$5698_1x ..
               serial_number=990 ..
               iou_connection=((ch1 0))

define_element element=blue_e1 ..
               element_identification=$5698_1x ..
               serial_number=991 ..
               iou_connection=((ch1 1))

"Define each tape unit.

define_element element=red_u0 ..
               element_identification=$698_3x ..
               serial_number=2010 ..
               peripheral_connection=((red_e0 0))

define_element element=red_u1 ..
               element_identification=$698_3x ..
               serial_number=2011 ..
               peripheral_connection=((red_e0 1))
      .
      .
      .
define_element element=red_u7 ..
               element_identification=$698_3x ..
               serial_number=2017 ..
               peripheral_connection=((red_e0 7))

define_element element=blue_u0 ..
               element_identification=$698_3x ..
               serial_number=2020 ..
               peripheral_connection=((blue_e1 0))

define_element element=blue_u1 ..
               element_identification=$698_3x ..
               serial_number=2021 ..
               peripheral_connection=((blue_e1 1))
      .
      .
      .
define_element element=blue_u7 ..
               element_identification=$698_3x ..
               serial_number=2027 ..
               peripheral_connection=((blue_e1 7))
```

Example 11: 5698-1x Alternate-Access, Daisy-Chain Configuration

The example illustrated in figure 2-11 is part of a configuration file showing the interconnection of $5698_1x tape controllers and $698_3x tape units in an alternate-access, daisy-chain configuration. In this case, each channel has access to two tape controllers and the same 16 units. Four tape units can be accessed at any one time, one per controller. Each channel is time-shared. This allows each of the channel's two controllers to maintain the data transfer rate of a $698_3x tape unit. This arrangement is possible because the channel is approximately six times faster than the $698_3x tape unit.

Although both concurrent and nonconcurrent channels are shown in the figure, the accompanying coding reflects only nonconcurrent channels (CH1 and CH3). Concurrent channels may be used on an I4 IOU.

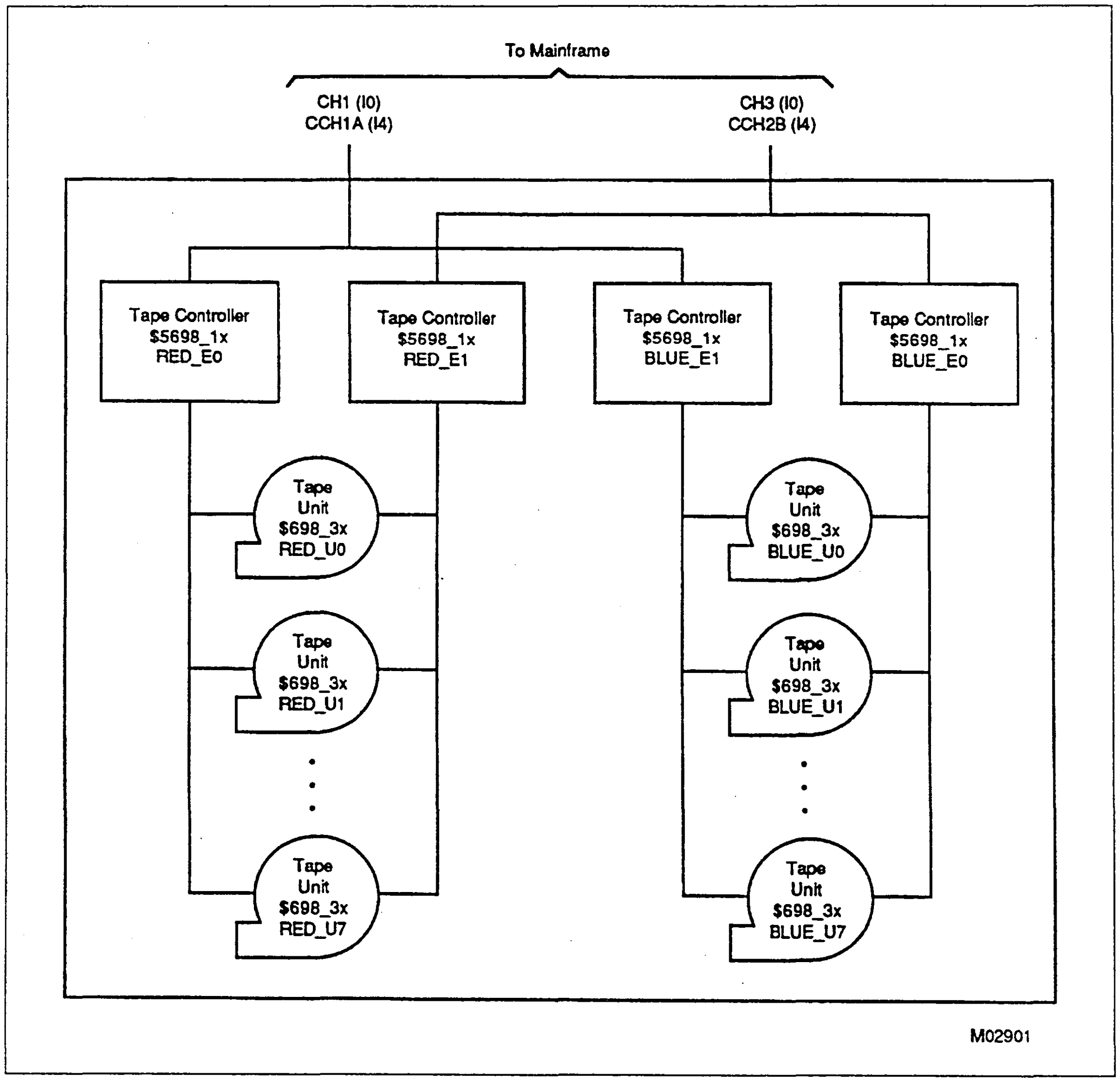

To Mainframe
CH1 (I0)
CCH1A (I4)
CH3 (I0)
CCH2B (I4)
Tape Controller $5698_1x RED_E0
Tape Controller $5698_1x RED_E1
Tape Controller $5698_1x BLUE_E1
Tape Controller $5698_1x BLUE_E0
Tape Unit $698_3x RED_U0
Tape Unit $698_3x RED_U1
Tape Unit $698_3x RED_U7
Tape Unit $698_3x BLUE_U0
Tape Unit $698_3x BLUE_U1
Tape Unit $698_3x BLUE_U7
M02901

Figure 2-11. 5698-1x Alternate-Access, Daisy-Chain Configuration Example:

```
"Define each controller.

define_element element=red_e0 ..
               element_identification=$5698_1x ..
               serial_number=990 ..
               iou_connection=((ch1 0))

define_element element=red_e1 ..
               element_identification=$5698_1x ..
               serial_number=992 ..
               iou_connection=((ch3 1))

define_element element=blue_e0 ..
               element_identification=$5698_1x ..
               serial_number=991 ..
               iou_connection=((ch3 0))

define_element element=blue_e1 ..
               element_identification=$5698_1x ..
               serial_number=993 ..
               iou_connection=((ch1 1))

"Define each tape unit.

define_element element=red_u0 ..
               element_identification=$698_3x ..
               serial_number=2010 ..
               peripheral_connection=((red_e0 0) ..
                                      (red_e1 0))

define_element element=red_u1 ..
               element_identification=$698_3x ..
               serial_number=2011 ..
               peripheral_connection=((red_e0 1) ..
                                      (red_e1 1))
      .
      .
      .
define_element element=red_u7 ..
               element_identification=$698_3x ..
               serial_number=2017 ..
               peripheral_connection=((red_e0 7) ..
                                      (red_e1 7))
```

```
define_element element=blue_u0 ..
               element_identification=$698_3x ..
               serial_number=2020 ..
               peripheral_connection=((blue_e0 0) ..
                                      (blue_e1 0))

define_element element=blue_u1 ..
               element_identification=$698_3x ..
               serial_number=2021 ..
               peripheral_connection=((blue_e0 1) ..
                                      (blue_e1 1))

     .
     .
     .
define_element element=blue_u7 ..
               element_identification=$698_3x ..
               serial_number=2027 ..
               peripheral_connection=((blue_e0 7) ..
                                      (blue_e1 7))
```

Example 12: 5698-1x Redundant Path Configuration

The example illustrated in figure 2-12 is part of a configuration file showing the interconnection of $5698_1x tape controllers and $698_3x tape units in a redundant path configuration. In this case, only one PP is loaded and is active at a time because only one tape controller port can be active at a time. At deadstart, NOS/VE chooses, as the primary channel, the first ON channel specified by the IOU_CONNECTION parameter of the PCU subcommand DEFINE_ELEMENT that defines the controller. In the example, CH1 is the primary channel, and CH3 is the redundant channel.

You can switch connections to a controller without interrupting the system by using the LCU subcommand CHANGE_ELEMENT_STATE to set the faulty channel to DOWN.

NOTE

The address of both ports of the controller must be the same.

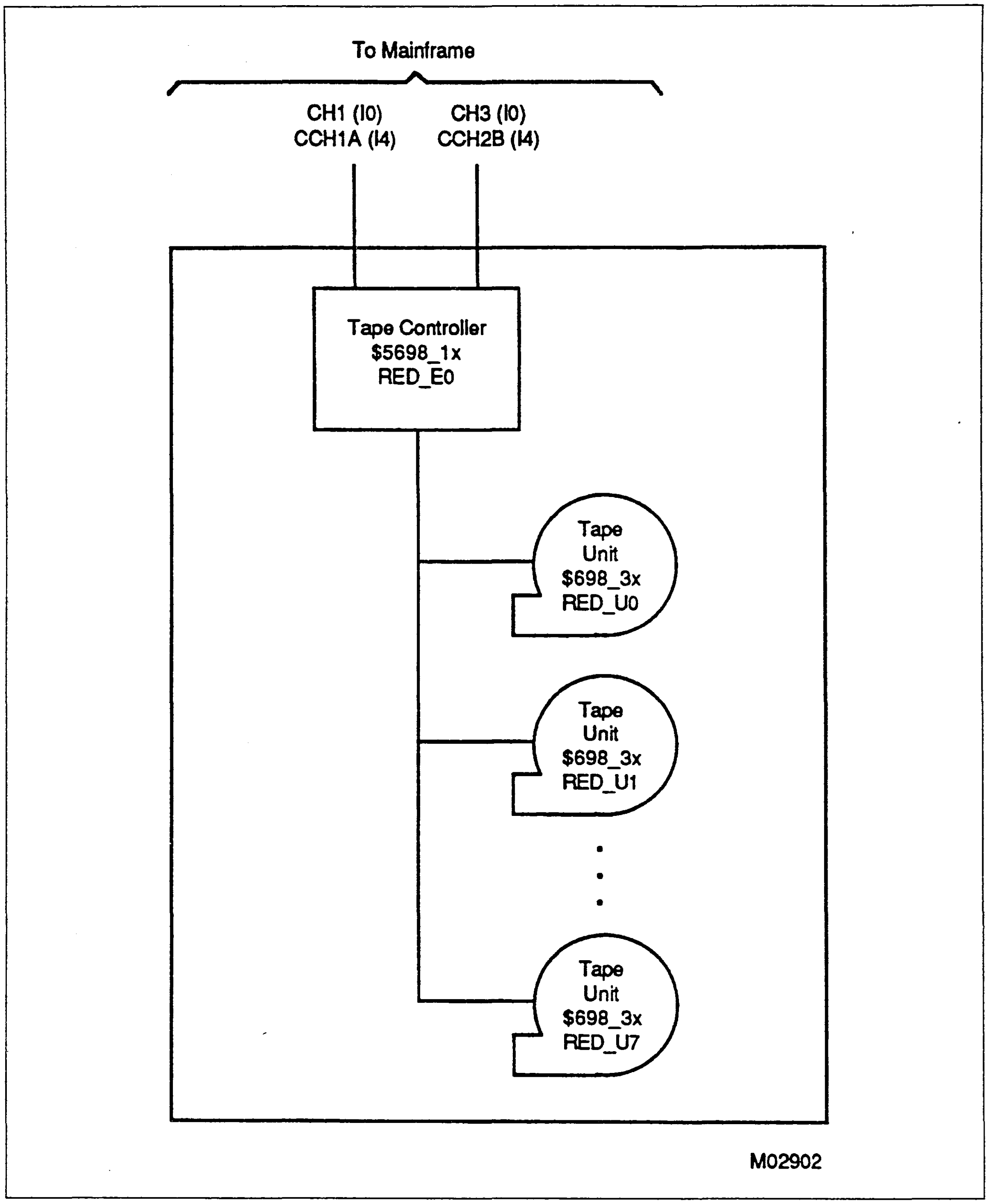


Figure 2-12. 5698-1x Redundant Path Configuration Example

```
"Define each controller.

define_element element=red_e0 ..
               element_identification=$5698_1x ..
               serial_number=990 ..
               iou_connection=((ch1 0)(ch3 0))

"Define each tape unit.

define_element element=red_u0 ..
               element_identification=$698_3x ..
               serial_number=2010 ..
               peripheral_connection=((red_e0 0)) ..

define_element element=red_u1 ..
               element_identification=$698_3x ..
               serial_number=2011 ..
               peripheral_connection=((red_e0 1)) ..
     .
     .
     .
define_element element=red_u7 ..
               element_identification=$698_3x ..
               serial_number=2017 ..
               peripheral_connection=((red_e0 7)) ..
```

This page intentionally left blank.

Example 13: 5698-1x Daisy-Chain, Redundant Path, Single-Access Configuration

The example illustrated in figure 2-13 is part of a configuration file showing the interconnection of $5698_1x tape controllers and $698_3x tape units in a daisy-chain, redundant path, single-access configuration. In this case, the number of channels actively configured depends on how the controllers are defined in the NOS/VE physical configuration.

For a dual-CPU 960-series system, we do not recommend that multiple $5698_1x tape controllers be configured in any of the possible daisy-chain configurations on the same channel. This configuration may cause controller machine exceptions (data overrun) on physical tape records that are longer than 8,192 bytes. Although this error can be recovered by the system, tape subsystem performance is adversely affected. When the default record size for backing up permanent files (4,128 bytes) is in effect, controller machine exceptions errors are not encountered when multiple controllers are daisy-chained on the same IPI channel.

NOTE

The address of both ports of the same controller must be the same. However, the address of each controller on each string must be unique.

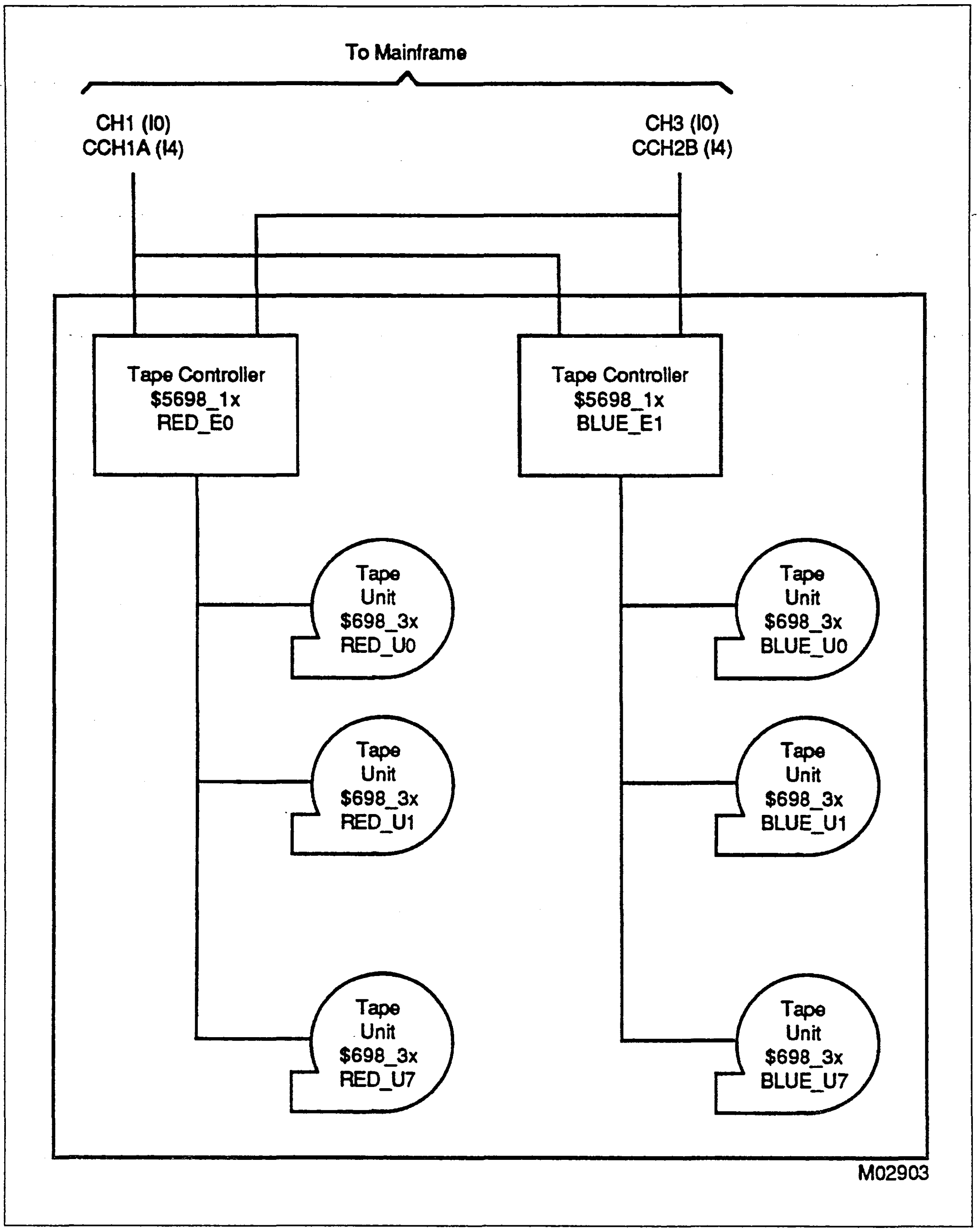


Figure 2-13. 5698-1x Daisy-Chain/Redundant Path/Single-Access Configuration Example

In the following portion of the configuration file, channel CH1 is actively configured, and channel CH3 is used for redundancy. Both controllers (RED_E0 and BLUE_E1) are accessible only from channel CH1.

```
"Define each controller.

define_element element=red_e0 ..
               element_identification=$5698_1x ..
               serial_number=990 ..
               iou_connection=((ch1 0) (ch3 0))

define_element element=blue_e1 ..
               element_identification=$5698_1x ..
               serial_number=991 ..
               iou_connection=((ch1 1) (ch3 1))

"Define each tape unit.

define_element element=red_u0 ..
               element_identification=$698_3x ..
               serial_number=2010 ..
               peripheral_connection=((red_e0 0))

define_element element=red_u1 ..
               element_identification=$698_3x ..
               serial_number=2011 ..
               peripheral_connection=((red_e0 1))
     .
     .
     .
define_element element=red_u7 ..
               element_identification=$698_3x ..
               serial_number=2017 ..
               peripheral_connection=((red_e0 7))

define_element element=blue_u0 ..
               element_identification=$698_3x ..
               serial_number=2020 ..
               peripheral_connection=((blue_e1 0))

define_element element=blue_u1 ..
               element_identification=$698_3x ..
               serial_number=2021 ..
               peripheral_connection=((blue_e1 1))
     .
     .
     .
define_element element=blue_u7 ..
               element_identification=$698_3x ..
               serial_number=2027 ..
               peripheral_connection=((blue_e1 7))
```

In the following alternative controller configuration, both channels are actively configured (it is assumed that both channels are set to ON). Controller RED_E0 is accessed only from channel CH1, and controller BLUE_E1 is accessed only from channel CH3. Each channel provides redundant access to a controller primarily accessed by another channel.

```
"Define each controller

define_element element=red_e0 ..
               element_identification=$5698_1x ..
               serial_number=990 ..
               iou_connection=((ch1 0) (ch3 0))

define_element element=blue_e1 ..
               element_identification=$5698_1x ..
               serial_number=991 ..
               iou_connection=((ch3 1) (ch1 1))

"Define each tape unit.

define_element element=red_u0 ..
               element_identification=$698_3x ..
               serial_number=2010 ..
               peripheral_connection=((red_e0 0))

define_element element=red_u1 ..
               element_identification=$698_3x ..
               serial_number=2011 ..
               peripheral_connection=((red_e0 1))
      .
      .
      .
define_element element=red_u7 ..
               element_identification=$698_3x ..
               serial_number=2017 ..
               peripheral_connection=((red_e0 7))

define_element element=blue_u0 ..
               element_identification=$698_3x ..
               serial_number=2020 ..
               peripheral_connection=((blue_e1 0))

define_element element=blue_u1 ..
               element_identification=$698_3x ..
               serial_number=2021 ..
               peripheral_connection=((blue_e1 1))
      .
      .
      .
define_element element=blue_u7 ..
               element_identification=$698_3x ..
               serial_number=2027 ..
               peripheral_connection=((blue_e1 7))
```

Example 14: 5680-11: Single-Access Configuration

The example illustrated in figure 2-14 is part of a configuration file showing the interconnection of a $5680_11 tape controller and $5682_12 cartridge tape units in a single-access configuration. Only one tape unit can be accessed at any one time.

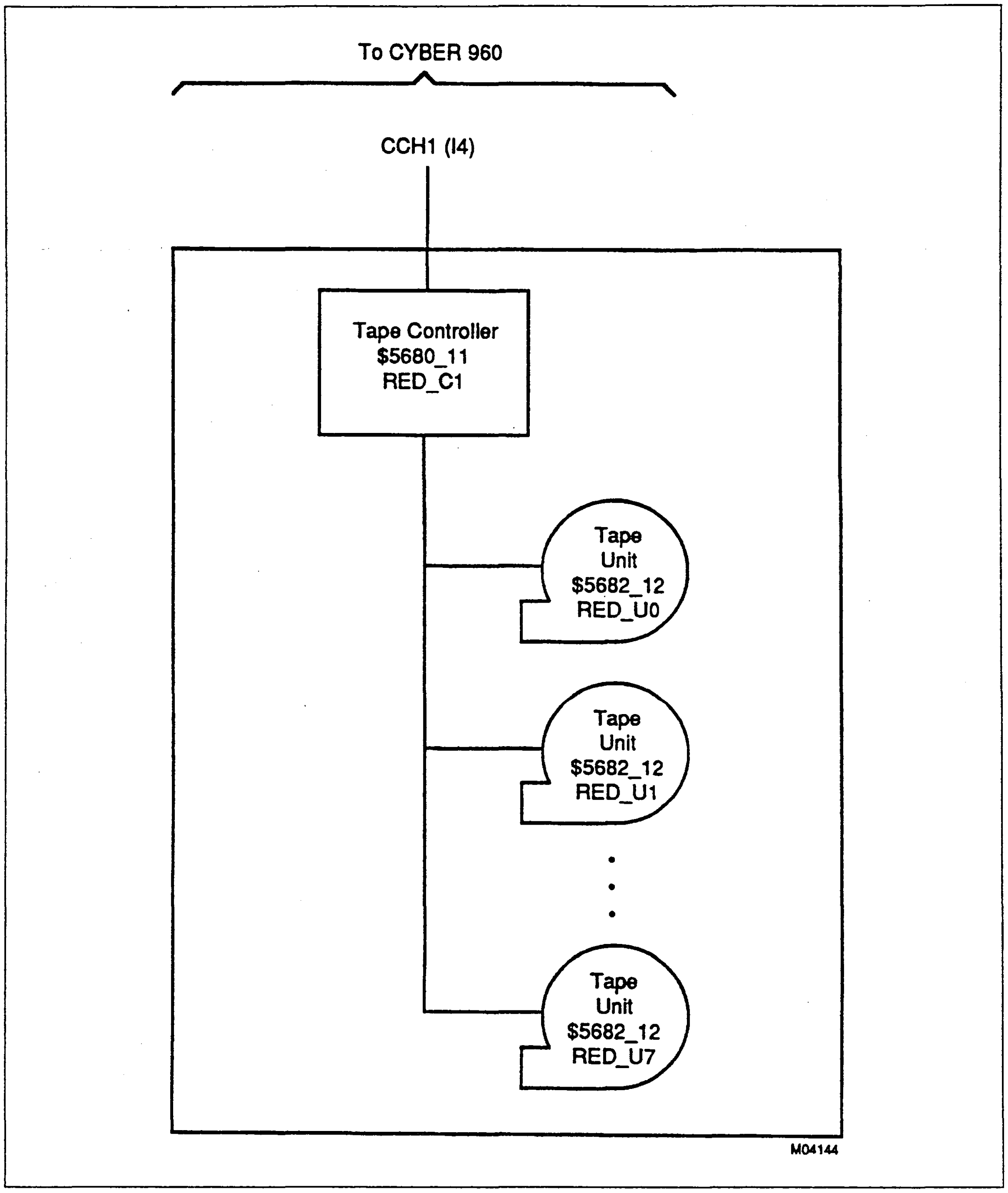


Figure 2-14. 5680-11 Single-Access Configuration Example

```
"Define the controller connection to a CYBER 960 (I4C).

define_element element=red_c1 ..
               element_identification=$5680_11 ..
               serial_number=998 ..
               iou_connection=((cch1 0))

"Define each tape unit.

define_element element=red_u0 ..
               element_identification=$5682_12 ..
               serial_number=2010 ..
               peripheral_connection=((red_c1 0))

define_element element=red_u1 ..
               element_identification=$5682_12 ..
               serial_number=2011 ..
               peripheral_connection=((red_c1 1))
      .
      .
      .
define_element element=red_u7 ..
               element_identification=$5682_12 ..
               serial_number=2017 ..
               peripheral_connection=((red_c1 7))
```

This page intentionally left blank.

Example 15: 5680-11 Redundant-Access Configuration

The example illustrated in figure 2-15 is part of a configuration file showing the interconnection of $5680_11 tape controllers and $5682_12 cartridge tape units in a redundant-access configuration. In this case, two tape units can be accessed concurrently, one per controller. However, only one controller is capable of accessing a particular tape unit. Either channel may be an NIO or a CIO channel. A CIO configuration is shown in the example.

The primary controller for a $5682_12 tape unit is the first controller identified in the DEFINE_ELEMENT subcommand's PERIPHERAL_CONNECTION parameter for the tape unit.

NOTE

For optimum performance, we recommend the following:

- Configure the odd-numbered units to one controller and the even-numbered units to the second controller.
- Number the tape units sequentially from 0 to 7.
- Distribute tape mounts equally among the two controllers' tape units.

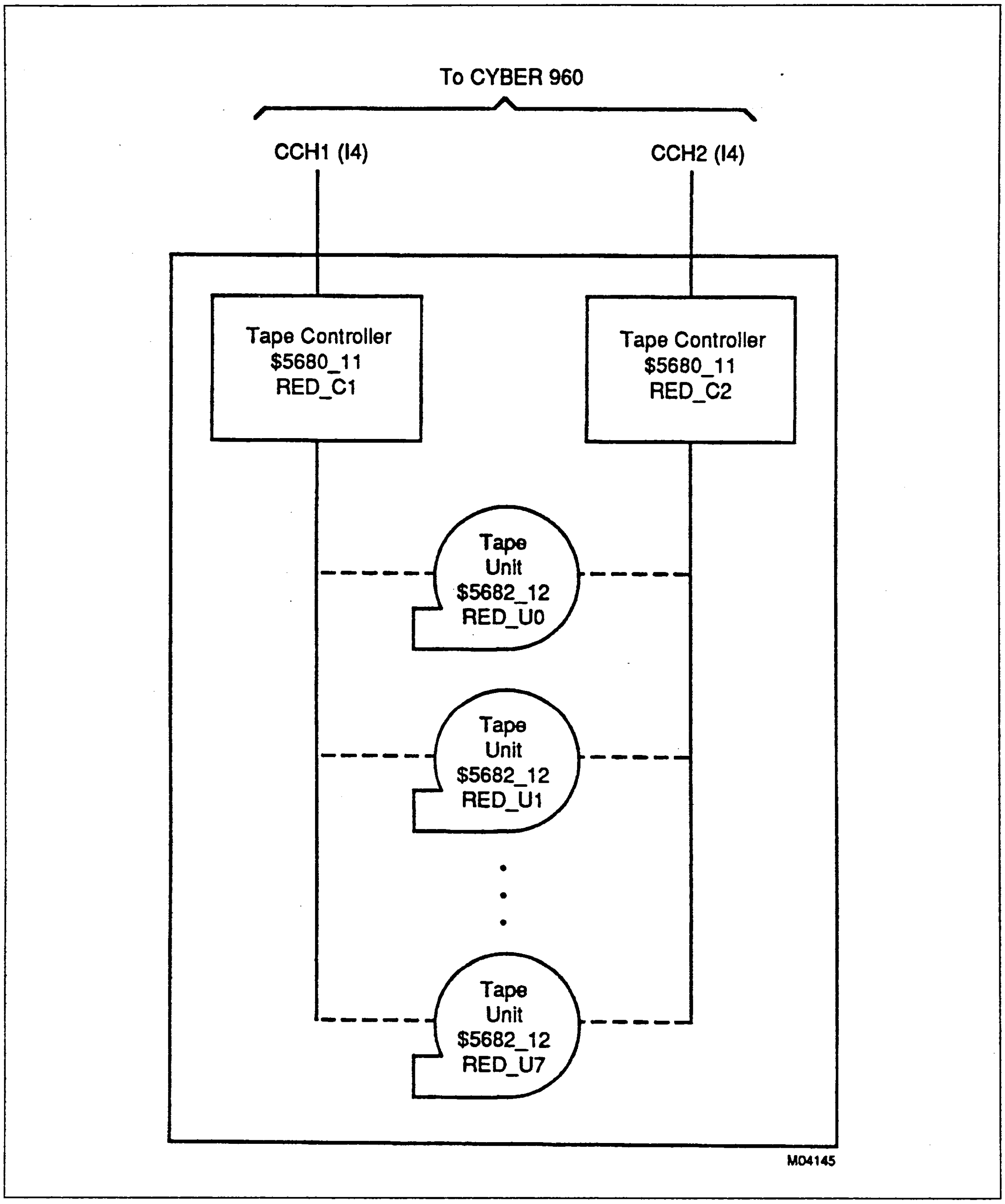

To CYBER 960
CCH1 (I4)
CCH2 (I4)
Tape Controller $5680_11 RED_C1
Tape Controller $5680_11 RED_C2
Tape Unit $5682_12 RED_U0
Tape Unit $5682_12 RED_U1
Tape Unit $5682_12 RED_U7
M04145

Figure 2-15. 5680-11 Redundant-Access Configuration Example

```
"Define the controller connection to a CYBER 960 (I4C).

define_element element=red_c1 ..
               element_identification=$5680_11 ..
               serial_number=998 ..
               iou_connection=((cch1 0))

define_element element=red_c2 ..
               element_identification=$5680_11 ..
               serial_number=999 ..
               iou_connection=((cch2 0))

"Define each tape unit.

define_element element=red_u0 ..
               element_identification=$5682_12 ..
               serial_number=2010 ..
               peripheral_connection=((red_c1 0) ..
                                      (red_c2 0))

define_element element=red_u1 ..
               element_identification=$5682_12 ..
               serial_number=2011 ..
               peripheral_connection=((red_c2 1) ..
                                      (red_c1 1))

define_element element=red_u2 ..
               element_identification=$5682_12 ..
               serial_number=2012 ..
               peripheral_connection=((red_c1 2) ..
                                      (red_c2 2))

define_element element=red_u3 ..
               element_identification=$5682_12 ..
               serial_number=2013 ..
               peripheral_connection=((red_c2 3) ..
                                      (red_c1 3))

define_element element=red_u4 ..
               element_identification=$5682_12 ..
               serial_number=2014 ..
               peripheral_connection=((red_c1 4) ..
                                      (red_c2 4))

define_element element=red_u5 ..
               element_identification=$5682_12 ..
               serial_number=2015 ..
               peripheral_connection=((red_c2 5) ..
                                      (red_c1 5))

define_element element=red_u6 ..
               element_identification=$5682_12 ..
               serial_number=2016 ..
               peripheral_connection=((red_c1 6) ..
                                      (red_c2 6))
```

```
define_element element=red_u7 ..
               element_identification=$5682_12 ..
               serial_number=2017 ..
               peripheral_connection=((red_c2 7) ..
                                      (red_c1 7))
```

Logical Configuration Utility 3

Logical Configuration Utility 3

Logical Configuration Utility (LCU) activities can be divided into two general categories:

- Defining logical characteristics of peripheral devices that were defined (using the PCU) in the mainframe's physical configuration.
- Creating and installing the network configuration file for the mainframe.

The Logical Configuration Utility can only be initiated as part of a System Operator Utility session. While in the LCU, the `LCU/` prompt informs you that you may enter an LCU subcommand or use an LCU function. A QUIT subcommand terminates the utility. For more information about entering the LCU and using its subcommands, see Validation Capabilities and Access Controls later in this chapter.

The status variable returned by the LOGICAL_CONFIGURATION_UTILITY command is the status of the entire utility. This allows you to program the execution of the utility using standard SCL procedure, data, and structured statement capabilities.

Each LCU subcommand has a status variable. The same general rules for command disposition that apply to all commands are applicable to the LCU subcommands. Specifically, if the status parameter is supplied on a subcommand that terminates abnormally, the reason for the termination is placed in the status variable and the utility continues processing.

The Logical Configuration Utility also provides several functions that make configuration-related material available to writers of SCL procedures.

Summary of LCU Subcommands and Functions

Table 3-1 provides a brief description of each of the LCU subcommands. Detailed descriptions are provided in the LCU Command and Subcommands section.

Table 3-1. Summary of LCU Subcommands

LCU Subcommand	Description
Network definition subcommands:	
DEFINE_HOST_NETWORK	Defines a network identifier by which the NOS/VE host is known in the network.
DEFINE_NETWORK_CONNECTION	Defines a network to which NOS/VE is connected.
Network configuration subcommands:	
DISPLAY_NETWORK_CONFIGURATION	Displays the active network connections defined for NOS/VE.
INSTALL_NETWORK_CONFIGURATION	Verifies and installs a file containing network definition subcommands.
VERIFY_NETWORK_CONFIGURATION	Performs the same verification testing as the INSTALL_ NETWORK_CONFIGURATION subcommand.
Mass storage device management subcommands:	
ADD_VOLUME_TO_SET	Adds a volume to a mass storage set.
CHANGE_MS_CLASS	Changes the mass storage class membership of a disk volume.
CHANGE_MS_VOLUME	Changes the default values for allocation size and transfer size for a disk volume.
CREATE_SET	Creates a new mass storage set (other than the system set).
DISPLAY_MS_CLASS	Displays the class membership assignments of a disk volume.

(Continued)

Table 3-1. Summary of LCU Subcommands *(Continued)*

LCU Subcommand	Description
Mass storage device management subcommands:	
DISPLAY_MS_VOLUME	Displays the current default values for allocation size and transfer size for a disk volume.
INITIALIZE_MS_VOLUME	Erases a disk volume and performs initialization functions.
Mass storage defect management subcommands:	
DEFINE_MS_FLAW	Flaws a physical location of a mass storage volume in response to a media failure.
DISPLAY_MS_FLAWS	Displays the list of media flaws for a volume.
REMOVE_MS_FLAW	Removes media flaws previously defined automatically by NOS/VE or by a DEFINE_MS_FLAW subcommand.
Other subcommands:	
CHANGE_ELEMENT_STATE	Changes the logical state (ON, OFF, or DOWN) of a peripheral element or channel.
DISPLAY_MAINFRAME_CONFIGURATION	Displays information about one or more elements in a mainframe's active configuration.
QUIT	Terminates the Logical Configuration Utility.

Table 3-2 supplies a brief description of each of the LCU functions. Detailed descriptions are provided in the LCU Functions section.

Table 3-2. Summary of LCU Functions

LCU Function	Description
$ACTIVE_SETS	Returns a list of the names of all the active mass storage sets on the requesting mainframe.
$ACTIVE_SET_FAMILIES	Returns a list of the names of families assigned to the active mass storage sets.
$ACTIVE_SET_MEMBERS	Returns the element name and the recorded VSN for each disk volume that is a member of the specified set.
$CHANNEL_PORT	Returns the name of the channel port connecting a channel to a peripheral element.
$ELEMENT	Returns the value of an element attribute.
$MASS_STORAGE_CLASS_ MEMBERS	Returns the element name and the recorded VSN for each member of a mass storage class.
$PHYSICAL_ADDRESS	Returns an integer value representing the physical address from an upline element to a downline element.
$STORAGE_DEVICE_NAME	Returns the name of the disk or tape unit element on which a volume is mounted.
$SYSTEM_SET_NAME	Returns the name of the system set.

Validation Capabilities and Access Controls

This section supplies information about entering an LCU session, as well as restrictions on the use of LCU subcommands once you begin a session.

Starting an LCU Session

You can enter the LOGICAL_CONFIGURATION_UTILITY command and the utility's subcommands only from within a System Operation Utility session, described in detail in the NOS/VE Operations manual. In addition, you must have the proper validation capabilities as described in the next section.

To start a System Operator Utility session, enter:

```
/system_operator_utility
sou/
```

After the prompt, enter the LOGICAL_CONFIGURATION_UTILITY command to begin an LCU session:

```
sou/logical_configuration_utility
LCU/
```

The LCU/ prompt indicates that you can now enter LCU subcommands.

Validation Capabilities

To use LCU subcommands, you need special validation capabilities. The System Operator Utility uses validation capabilities to control access to subsets of the operator commands, as well as to system and user permanent files. Validation capabilities are described in detail in the NOS/VE Operations manual.

LCU subcommands are under the control of one or more of the following validation capabilities:

Validation Capability	Description
CONFIGURATION_ ADMINISTRATION	Provides access to subcommands that configure system options and characteristics.
REMOVABLE_MEDIA_ OPERATION	Provides access to subcommands related to the operation of removable media storage devices such as magnetic tape units.
SYSTEM_OPERATION	Provides access to subcommands related to the operation of disk devices and network devices.
SYSTEM_DISPLAYS	Provides access to subcommands that display system and job information.

With the exception of the CHANGE_ELEMENT_STATE subcommand, all subcommands that control the logical configuration are controlled by the CONFIGURATION_ADMINISTRATION validation capability. The Remarks section for each LCU subcommand description indicates the capability for which you must be validated in order to use the subcommand.

Access Controls

The Logical Configuration Utility prevents more than one user from modifying the logical configuration at the same time. With the exception of the CHANGE_ELEMENT_STATE subcommand, only the LCU subcommands of the user who accesses the utility first are actually executed. If another user tries to execute one of the affected LCU subcommands during this time, the utility returns an error. Another user can execute LCU subcommands only after the first user enters the LCU subcommand QUIT. In addition, NOS/VE cannot automatically set any elements in the configuration to DOWN until after the privileged user enters a QUIT subcommand.

However, two users can execute CHANGE_ELEMENT_STATE subcommands concurrently under the following conditions:

- One user executes with a REMOVABLE_MEDIA_OPERATION capability while changing the state of magnetic tape elements.
- The other user executes with a CONFIGURATION_ADMINISTRATION or SYSTEM_OPERATION capability while changing the state of elements not associated with magnetic tape.

We recommend that you promptly exit the LCU after executing whatever subcommands are required. This allows others with the same System Operator Utility capability to use the LCU.

Defining Device Logical Characteristics

Logical device operations or characteristics controlled by the LCU include:

- Initialization of disk volumes
- Initialization of tape volumes (described in the NOS/VE Operations manual)
- Mass storage volume class membership
- Changing the state (ON, OFF, or DOWN) of peripheral elements defined by the PCU

Initializing Disk Volumes

You must initialize and add a disk volume to a mass storage set before you can place files or catalogs on the volume. The system disk volume is automatically initialized and added to the system set during an installation deadstart. You must manually initialize all other disk volumes using the INITIALIZE_MS_VOLUME subcommand. Do not attempt to reinitialize the system disk volume with this subcommand. You can use the INITIALIZE_MS_VOLUME subcommand to initialize a volume either during a continuation deadstart or while the system is running. However, you cannot initialize an active volume without first performing a continuation deadstart. You can initialize a volume either at the first LCU intervention during deadstart or during the LCU session in which the volume's disk unit is first turned ON following a deadstart.

A volume that has been initialized but not yet added to a set is only accessible to NOS/VE or to a maintenance job.

To prepare a volume for use, we recommend the following sequence of LCU subcommands:

```
INITIALIZE_MS_VOLUME
CHANGE_MS_CLASS
CHANGE_MS_VOLUME
CREATE_SET (if required)
ADD_VOLUME_TO_SET
DEFINE_MS_FLAW (if required)
```

Refer to the descriptions of these subcommands for specific information about how to use them.

Before initializing a volume, you should be familiar with the conditions under which mass storage (disk) subcommands can be used and what effects they have on the state of the volume. Tables 3-3, 3-4, and 3-5 summarize these conditions. These tables use the following terms to describe volume states:

State	Description
ON	The disk unit on which the volume is mounted is in the ON state.
Avail	The disk unit is operational and accessible.
Init	The volume is initialized.
Online	System files on the volume are intact and accessible. System access to system files on the volume is permitted if the disk unit is ON and available, and the volume is online.
Active	The volume is online, is a member of a set, and is mounted on a disk unit whose state was ON during or since the last deadstart. User access to files or catalogs on a volume is permitted if the disk unit is ON and available and the volume is active.

Table 3-3 identifies the required state of the disk unit and the volume mounted on the disk unit for successfully executing the various LCU subcommands that operate on mass storage volumes:

Table 3-3. Volume State Requirements

Subcommand	ON	Avail	Init	Online	Active
ADD_VOLUME_TO_SET	X	X	X		
CHANGE_MS_CLASS	X	X	X	X	
CHANGE_MS_VOLUME	X	X	X	X	
CREATE_SET[1]	X	X	X		
DEFINE_MS_FLAW					
DISPLAY_MS_CLASS	X	X	X	X	
DISPLAY_MS_FLAW	X	X	X	X	
DISPLAY_MS_VOLUME	X	X	X	X	
INITIALIZE_MS_VOLUME	X	X			
REMOVE_MS_FLAW	X	X	X	X	X

1. The volume state requirements shown are for the master volume of the set.

Tables 3-4 and 3-5 identify the effect that a mass storage subcommand has on the state of a disk unit and the volume mounted on it. Table 3-4 shows the state of a volume before the subcommand is issued; table 3-5 shows the state of a volume after the subcommand is issued. Subcommands not listed here have no effect on volume or disk unit states.

Table 3-4. Volume State Changes Before Issuing LCU Subcommand

Subcommand	ON	Avail	Init	Online	Active
ADD_VOLUME_TO_SET	X	X	X		
CHANGE_ELEMENT_STATE (DOWN/OFF to ON)					
CHANGE_ELEMENT_STATE (ON to DOWN/OFF)	X	X	X	X	X
CREATE_SET[1]	X	X	X		
INITIALIZE_MS_VOLUME	X	X			
QUIT	X	X	X		

Table 3-5. Volume State Changes After Issuing LCU Subcommand

Subcommand	ON	Avail	Init	Online	Active
ADD_VOLUME_TO_SET	X	X	X	X	X
CHANGE_ELEMENT_STATE (DOWN/OFF to ON)	X	X			
CHANGE_ELEMENT_STATE (ON to DOWN/OFF)			X	X	X
CREATE_SET[1]	X	X	X	X	X
INITIALIZE_MS_VOLUME	X	X	X	X	
QUIT				X[2]	X[3]

1. The volume state requirements shown are for the master volume of the set.

2. If you issue an LCU QUIT subcommand following a CHANGE_ELEMENT_STATE subcommand that changes the state of a disk unit from DOWN or OFF to ON, the volume is brought online if previously initialized.

3. If you issue an LCU QUIT subcommand following a CHANGE_ELEMENT_STATE subcommand that changes the state of a disk unit from DOWN or OFF to ON, the volume is activated if it can be brought online and is a member of a set.

Mass Storage Volume Class Membership

NOS/VE supports 26 mass storage classes, each named by a single alphabetic character (A to Z). The class or classes to which a volume belongs determines the type of files or catalogs that can be stored on that volume. For example, volumes belonging to class M contain permanent files, while volumes belonging to class C contain system swap files. Classes U to Z are definable by site analysts.

Default class assignments for a volume are made when the volume is initialized with the INITIALIZE_MS_VOLUME subcommand. By default, a volume belongs to all 26 mass storage classes. In most cases, it is advisable to alter the default class memberships to provide the optimum balance between maximum performance and maximum availability for catalogs and permanent files. We recommend that you change the default assignments (using the CHANGE_MS_CLASS subcommand) before adding the volume to a mass storage set (using the ADD_VOLUME_TO_SET subcommand)

Suggestions about how to assign mass storage classes appear in the CHANGE_MS_ CLASS subcommand description and in Part IV of this manual, Fault Tolerance Configuration and Failure Analysis.

Table 3-6 describes the 26 mass storage classes and specifies whether a class needs to be assigned to a volume in the system mass storage set or a nonsystem mass storage set. Mass storage sets are described in the next section.

Table 3-6. Relationship Between Mass Storage Classes and Sets

Class	Class Description and Set Requirements
A and B	Reserved for Control Data.
C	Swap files. These files are only recorded on a system set volume because they belong to user :$SYSTEM.$SYSTEM. The system set must have at least one volume assigned to class C. If mass storage class C is assigned to a volume in a nonsystem set, it is ignored.
D to I	Reserved for Control Data.
J	$SYSTEM catalogs. Catalogs created by the user $SYSTEM are assigned to the set associated with the login family. Since each family has a $SYSTEM user, each set must have at least one volume assigned to class J.
K	Input queue files, output queue files, service-critical files, and $SYSTEM permanent files. Every set must have at least one volume assigned to class K. *Input queue files* are assigned to either the system set or the set associated with the login-family of the submitting job. Consequently, every set must have at least one volume assigned to class K.

(Continued)

Table 3-6. Relationship Between Mass Storage Classes and Sets *(Continued)*

Class	Class Description and Set Requirements
	An input queue file is placed in one of the following catalogs: :$SYSTEM.$SYSTEM.$JOB_INPUT_QUEUE Input queue files are assigned to this catalog if the JOB_ DESTINATION_USAGE parameter specifies either VE_LOCAL or VE. VE is used when the login family is on a server mainframe but the family is not defined within the File Server as a leveled family. :$SYSTEM.$SYSTEM.$SF_JOB_INPUT_QUEUE Input queue files are assigned to this catalog when the JOB_ DESTINATION_USAGE parameter has any value except VE or VE_ LOCAL. :login_family.$SYSTEM.$JOB_INPUT_QUEUE Batch input queue jobs are assigned to this catalog when the JOB_ DESTINATION_USAGE parameter specifies VE and the login family is either directly accessible to the mainframe or is available as a leveled family through the File Server. *Output queue files* belong to user :$SYSTEM.$SYSTEM and are assigned to a volume in the system set. *Service-critical files* belong to user :$SYSTEM.$SYSTEM and are assigned to a volume in the system set. *$SYSTEM permanent files* belong to user $SYSTEM in the login family. These files are assigned to the set associated with the login family.
L	User catalogs. Catalogs created by any user except $SYSTEM are assigned to the set associated with the login family. Therefore, each nonsystem set must have at least one volume assigned to class L. If families other than $SYSTEM belong to the system set, the system set must have at least one volume assigned to class L.
M	User permanent files. Files created by any user except $SYSTEM are assigned to the set associated with the login family. NOS/VE installs products that are not service-critical to class M volumes in the system set. Therefore, all sets must have at least one volume assigned to class M.
N	User temporary files. Temporary files created by any job except the system job are assigned to any volume belonging to class N. The login family is ignored for temporary file assignment. The files are written on any set that has a volume assigned to class N. Therefore, you may choose to assign class N to volumes of any set. The only requirement is that at least one volume in the configuration must belong to class N.
O and P	Reserved for Control Data.

(Continued)

Table 3-6. Relationship Between Mass Storage Classes and Sets *(Continued)*

Class	Class Description and Set Requirements
Q	System-critical temporary files. Temporary files created by the system job are assigned to any volume belonging to class Q. The login family is ignored for temporary file assignment. The files are written on any set that has a volume assigned to class Q. Therefore, you may choose to assign class Q to volumes of any set. The only requirement is that the system device must belong to class Q. We strongly recommend that the system device be the only volume that is a member of class Q.
R to T	Reserved for Control Data.
U to Z	Reserved for site analysts.

Mass Storage Sets

A mass storage set is a logical grouping of physical disk volumes that are reserved for one or more families. A mass storage set, therefore, determines the amount of permanent file storage available to the assigned families.

On each mainframe, NOS/VE requires at least one mass storage set, the system set, and supports multiple mass storage sets. The system set is automatically created during an installation deadstart. The name of the system set is specified on the INITIALIZE_SYSTEM_DEVICE system core command described in chapter 4, System Core Commands.

You can create additional sets, called nonsystem sets, with the LCU subcommand CREATE_SET. Organizing files and catalogs into multiple sets has a number of advantages. For example, if you define sets on an individual family basis, it guarantees a fixed amount of capacity for each family. At sites where system and file availability is of primary importance, multiple sets can be used to provide backup permanent file storage. If a device failure occurs in one set, the alternate set is immediately available. In a multiple mainframe environment, multiple sets also provide a convenient way for one mainframe to acquire and access the files of another mainframe. This capability is useful for providing access to files on a downed mainframe.

You assign a family to a set when you create the family using the CREATE_FAMILY command. By default, a new family is associated with the system set. For details about the CREATE_FAMILY command, see chapter 8, Permanent File Maintenance.

Each set must have its volumes assigned to the appropriate mass storage classes. The classes define the volumes as critical devices, swap file devices, user file devices, and so on. Table 3-6 contains a description of the mass storage classes and the class requirements for volumes in system and nonsystem sets.

NOS/VE handles mass storage sets according to the following rules:

- Every mainframe must have a system set. All permanent files and catalogs created on the $SYSTEM family reside within the set. The system device is the master volume of this set.

- A mainframe can have one or more sets active at any time. Each set defined by one or more mass storage devices whose state is ON at deadstart is activated during deadstart, provided the master volume of the set is accessible and active. A set's master volume must be active for users to gain access to permanent files on that set.
- A family can belong to only one set. All permanent files and catalogs created on that family reside within the set.
- Every set except the system set must contain volumes that belong to classes J, K, L, and M. See table 3-6 for more information.
- The system set must contain volumes that belong to classes C, J, K, and M. See table 3-6 for more information.
- Temporary files (classes Q and N) are allocated on volumes without regard to set. Only mass storage class controls temporary file residence. You may allow or disallow temporary file residence on a given set by adjusting the mass storage class membership for volumes in the set. At least one volume in one of the sets must belong to class Q, and at least one volume in one of the sets must belong to class N. We strongly recommend that the system device be the only volume that is a member of class Q.
- You can activate a set after the system is deadstarted using the ACTIVATE_SET command. This allows one mainframe to recover a set from another (failed) mainframe and provide access to that set without requiring an interrupt of the recovering mainframe. The ACTIVATE_SET command is described in the NOS/VE Operations manual.
- Set names must be unique on a single mainframe. If you want to move sets between mainframes, the set names must be unique between mainframes as well. To change the name of a nonsystem set, you must initialize the set's master and member volumes and restore any missing permanent files and catalogs. For additional information, refer to chapter 12, Repair Solutions.
- A site may upgrade to a NOS/VE version that supports multiple sets without requiring an installation deadstart. New sets may be defined at any time.
- Other than standard NOS/VE security measures, there is no restriction placed on file or catalog references among family members in different sets.

For configuration recommendations for mass storage sets, see chapter 11, Fault Tolerance.

Changing Element States

Devices defined as elements of a system's physical configuration can be placed in any of three states: ON, OFF, or DOWN. You can change an element's current state using the LCU subcommand CHANGE_ELEMENT_STATE.

You can enter the CHANGE_ELEMENT_STATE subcommand during deadstart or while the system is running. The resulting state change takes place immediately. See the description of the CHANGE_ELEMENT_STATE subcommand later in this chapter for the rules and restrictions that apply.

Toggling Devices Between NOS/VE and NOS or NOS/BE

In a dual-state system, you can use the CHANGE_ELEMENT_STATE subcommand to toggle the ownership of devices between NOS/VE and the NOS or NOS/BE partner. This capability is available only for certain devices, and you must define a device to be toggled in the physical configuration for both NOS/VE and NOS or NOS/BE. See tables E-2 and E-3 (NOS EQPDECK Entries for Peripherals and NOS/BE CMR Entries for Peripherals, respectively) for information on which devices can be toggled. The CHANGE_ELEMENT_STATE subcommand description provides more information on the effects of toggling a device in this manner.

Creating and Installing a Network Configuration File

If your NOS/VE system is connected to a CDCNET network, you must create and install a network configuration file. The network configuration file defines the characteristics of the NOS/VE system's connections to one or more CDCNET networks. These characteristics include the type of connection, as well as the CDCNET network identifier of the network to which NOS/VE is attached.

Defining a Network Configuration

Within a CDCNET configuration, a network is defined as a physical media that connects two or more NOS/VE systems or CDCNET DIs. Each network must be assigned a network identifier. Examples of networks that may be included in a CDCNET configuration include Ethernet networks, HDLC networks, and X.25 networks.

To define a network configuration, two types of configuration files are required:

- For each CDCNET DI or ICA-II, a configuration file must be installed in the catalog $SYSTEM.CDCNET.SITE_CONTROLLED. This configuration file defines a DI's or an ICA-II's connections to CDCNET networks. The CDCNET Configuration Guide explains how to create and install CDCNET configuration files.

- For each NOS/VE system in the network, a network configuration file must be installed. This file defines a NOS/VE system's connections to one or more networks within the CDCNET configuration. Use the LCU subcommands described in this chapter to create and install this file.

Use the following LCU subcommands to define a NOS/VE system's network connections:

- DEFINE_NETWORK_CONNECTION (defines NOS/VE connections to any network; the connection can be provided by any CDCNET communications device (MDI, MTI, or ICA-II).)

- DEFINE_HOST_NETWORK (defines a unique network identifier by which the NOS/VE host is known in the network)

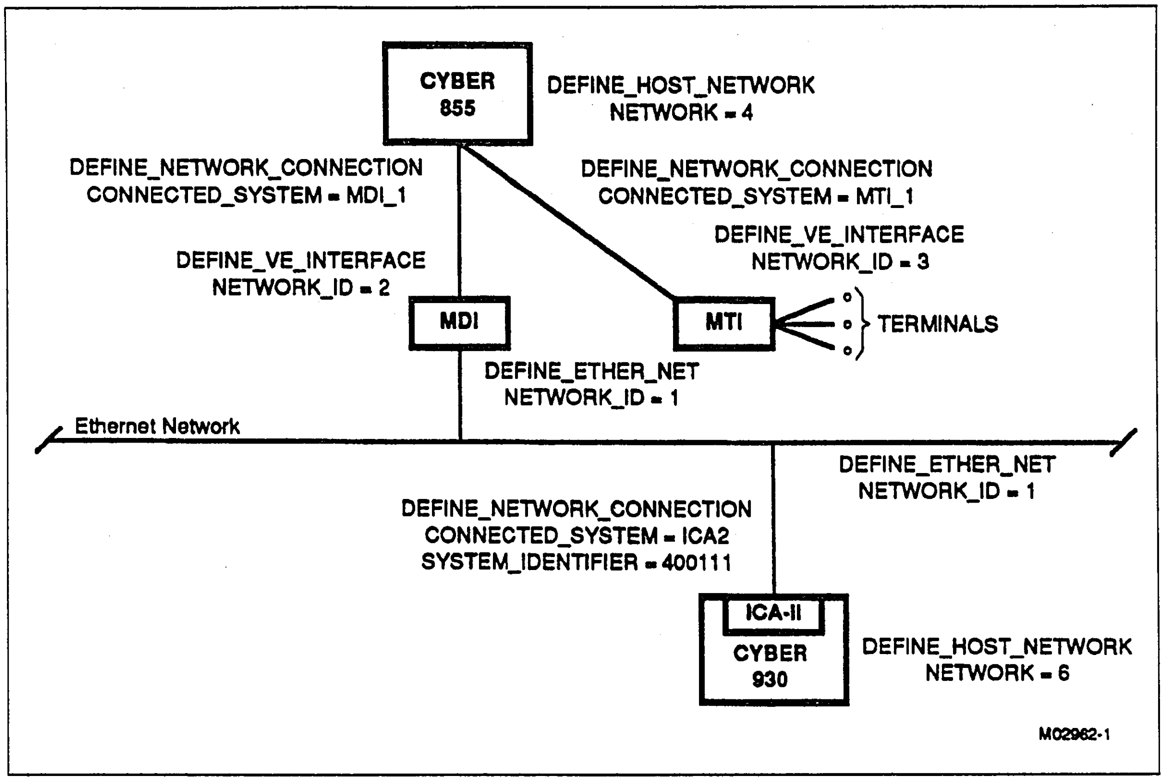


Figure 3-1. Sample Network Definition Commands

Figure 3-1 illustrates how LCU and CDCNET configuration commands define a network configuration involving a CYBER 930 system. DEFINE_ETHER_NET and DEFINE_VE_INTERFACE are CDCNET configuration commands that must be included in the configuration files for the MDI and ICA-II. They define the MDI's and ICA-II's respective connections to Ethernet and to the mainframe.

The LCU subcommand DEFINE_HOST_NETWORK specifies the unique network identifier that is assigned to each host system (4 for the CYBER 855 and 6 for the CYBER 930). The LCU subcommand DEFINE_NETWORK_CONNECTION defines the NOS/VE connections provided by an MDI (MDI_1), an MTI (MTI_1), and an ICA-II (ICA2). The CDCNET commands DEFINE_ETHER_NET and DEFINE_VE_INTERFACE must be included in the configuration file for the MDI, MTI, and ICA-II. They define the MDI's, MTI's, and ICA-II's respective connections to Ethernet and to the mainframe.

NOTE

All the commands and subcommands shown in figure 3-1 have other parameters, some of which may be required. The purpose of the figure is simply to show the correlation between the network identifier values as specified on the LCU subcommands and the CDCNET configuration commands.

Installing a Network Configuration File

You can use the EDIT_FILE utility to create a network configuration file. Include in the file all the DEFINE_NETWORK_CONNECTION and DEFINE_HOST_NETWORK subcommands required to define the network connections for your NOS/VE system. You then install the file using the LCU subcommand INSTALL_NETWORK_CONFIGURATION. This subcommand verifies the file for syntactical correctness before it installs the file. If errors are found, you must correct the file and then enter another INSTALL_NETWORK_CONFIGURATION subcommand.

A network configuration file installed by the INSTALL_NETWORK_CONFIGURATION subcommand is not immediately activated by NOS/VE. Rather, NOS/VE waits until the next time NAM/VE is activated, at which time the new network configuration is also activated. The activation is automatic; no further operator intervention is required.

If you want to verify a new network configuration without installing and activating it, you can use the VERIFY_NETWORK_CONFIGURATION subcommand, which is also described later in this section. In contrast to the INSTALL_NETWORK_CONFIGURATION subcommand, which you must enter from the system console, you can enter the VERIFY_NETWORK_CONFIGURATION subcommand from a user terminal.

LCU Command and Subcommands

In this section the LOGICAL_CONFIGURATION_UTILITY command is described first. The LCU subcommand descriptions, listed below, are presented in alphabetical order.

LOGICAL_CONFIGURATION_UTILITY command

- ADD_VOLUME_TO_SET subcommand
- CHANGE_ELEMENT_STATE subcommand
- CHANGE_MS_CLASS subcommand
- CHANGE_MS_VOLUME subcommand
- CREATE_SET subcommand
- DEFINE_CHANNEL_NETWORK subcommand
- DEFINE_HOST_NETWORK subcommand
- DEFINE_MS_FLAW subcommand
- DEFINE_NETWORK_ACCESS subcommand
- DEFINE_NETWORK_CONNECTION subcommand
- DISPLAY_MAINFRAME_CONFIGURATION subcommand
- DISPLAY_MS_CLASS subcommand
- DISPLAY_MS_FLAWS subcommand
- DISPLAY_MS_VOLUME subcommand
- DISPLAY_NETWORK_CONFIGURATION subcommand
- INITIALIZE_MS_VOLUME subcommand
- INSTALL_NETWORK_CONFIGURATION subcommand
- QUIT subcommand
- REMOVE_MS_FLAW subcommand
- VERIFY_NETWORK_CONFIGURATION subcommand

LOGICAL_CONFIGURATION_UTILITY Command

Purpose Initiates the Logical Configuration Utility.

Format LOGICAL_CONFIGURATION_UTILITY or
LCU
  STATUS=status variable

Parameters STATUS

Returns the completion status for the entire utility.

Remarks The Logical Configuration Utility can only be initiated as part of a System Operator Utility session. See Validation Capabilities and Access Controls earlier in this chapter.

ADD_VOLUME_TO_SET Subcommand

Purpose Adds a disk volume to a set.

Format **ADD_VOLUME_TO_SET** or
ADDVTS
  RECORDED_VSN=name
  SET_NAME=name
  STATUS=status variable

Parameters **RECORDED_VSN** or **MEMBER_VSN** or **MEMVSN** or **MV** or **RV**

Specifies the 6-character volume serial number (VSN) of the volume being added to a mass storage set. The VSN must be the same VSN specified on the INITIALIZE_MS_VOLUME subcommand. This parameter is required.

SET_NAME or *SN*

Specifies the name of the mass storage set associated with the disk volume being added. The default is the name of the current system set.

STATUS

Returns the completion status for this subcommand.

Remarks

- To use this subcommand, you must have a CONFIGURATION_ADMINISTRATION validation capability and be in a System Operator Utility session. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

- You can enter this subcommand anytime the LCU is executed, either during deadstart or while the system is running.

- A mass storage volume must be a member of a set before space on the volume can be assigned to user files or catalogs.

- When you issue this subcommand, the disk unit on which the volume is mounted must be in the ON state in the active configuration.

- The volume must have already been initialized by the INITIALIZE_MS_VOLUME subcommand. However, you need not initialize a volume each time you add it to the set.

- If the system device is initialized, NOS/VE implicitly recreates the system set. However, only those volumes whose disk unit is ON during the deadstart that initializes the system device are implicitly added back to the system set at that time.

  If a volume that was previously a member of the system set was OFF or DOWN at the time of the system device initialization, that volume must be explicitly added back to the set using the ADD_VOLUME_TO_SET subcommand. This may be done either during or after deadstart. Add a member back to the set without first initializing the volume only when you are certain that the volume has not been damaged.

Examples See the example under the INITIALIZE_MS_VOLUME subcommand.

CHANGE_ELEMENT_STATE Subcommand

Purpose

Changes the state (ON, OFF, or DOWN) of peripheral elements or channels connected to a mainframe. Refer to Remarks below for restrictions on the use of this subcommand.

Format

CHANGE_ELEMENT_STATE or
CHAES
  ELEMENT=name
  STATE=keyword
  IOU=name
  STATUS=status variable

Parameters

ELEMENT or E

Specifies the name of the element whose state is to be changed. This parameter is required.

STATE or S

Defines the desired state of the defined element. This parameter is required. You can specify one of the following keywords:

ON

Indicates that the element is operational.

OFF

Indicates that the element is not available for normal operations or maintenance.

DOWN

Indicates that the device is only available for maintenance purposes.

IOU

Specifies the name of the IOU to which a channel element is connected. Values you can specify for this parameter are IOU0 and IOU1. The default is IOU0.

STATUS

Returns the completion status for this subcommand.

Remarks

- To use this subcommand, you must have special validation and be in a System Operator Utility session.
  - If the specified element is a tape channel, tape controller, or tape unit, you must be validated for at least one of the following capabilities:

    CONFIGURATION_ADMINISTRATION
    REMOVABLE_MEDIA_OPERATION
    SYSTEM_OPERATION

  - For any other type of element, you must be validated for either the CONFIGURATION_ADMINISTRATION or SYSTEM_OPERATION capability.

  For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

- A central processor is denoted by CPn. The n value can be either 0 or 1.
- Restrictions on the use of this subcommand:
  - You cannot change the state of a disk unit whose volume is a member of class Q (system-critical) to OFF or DOWN after deadstart has completed.
  - You cannot change the state of a disk unit whose volume is a member of class J ($SYSTEM catalogs) in the system set to OFF or DOWN after deadstart has completed.
  - You cannot change a mass storage channel controller to OFF or DOWN if it provides the only access to a class Q or class J disk volume. However, a state change to DOWN or OFF is permitted if a redundant channel or controller exists, is in the ON state, and provides access to the critical storage device.
  - No state change of a tape channel, controller, or unit is permitted if access will be denied to a job that has a tape unit assigned to it or if it is currently the object of maintenance access.
  - NOS/VE disables access to a disk unit if it is faulty or cannot be accessed. However, the state of the volume in the logical configuration is not modified. The state of the faulty element should be manually changed to DOWN before repair is attempted. To reinstate the volume, change the state back to ON.
  - You cannot use this subcommand to change the state of a NAD communications element while RHFAM/VE is active.
  - You can change the state of a CPU (denoted by central processor element CPn) from ON to DOWN or from DOWN to ON. You cannot place a CPU in an OFF state.
  - You cannot set a CPU to the DOWN state if it is the only CPU turned on.
  - You cannot change the state of a dual-state CPU in a multiple mainframe system if the MANDATORY_DUALSTATE system attribute is set to 1.
  - You cannot change the state of a CPU that has been set to DOWN because of hardware problems. This situation requires a NOS/VE deadstart.
- A state change requested by this subcommand becomes effective immediately, regardless of whether you enter it during deadstart or during normal system operations.
- State changes made by this subcommand are preserved across continuation deadstarts.

- Any resources acquired by NOS/VE from NOS or NOS/BE are returned to those systems either when you issue the LCU subcommand CHANGE_ELEMENT_STATE or when NOS/VE terminates.

  If the element is a PP:

  - When the state of a channel is changed to OFF (from ON or DOWN), NOS/VE reverts ownership of any PPs configured to the channel back to the NOS or NOS/BE system. However, this action does not apply to PPs that belong to the second IOU (IOU1) in a multiple-IOU mainframe. NOS and NOS/BE can only reference PPs and channels belonging to IOU0.

  If the element is a mass storage element:

  - Once acquired from NOS or NOS/BE, a NOS/VE mass storage channel, controller, or storage device remains assigned to NOS/VE until NOS/VE terminates.

  If the element is not a mass storage element:

  - In a dual-state environment, changing the state of nonmass storage elements may cause ownership of the element to revert from NOS/VE to NOS or NOS/BE. To use such an element in the NOS or NOS/BE system, you must configure it on both NOS/VE and NOS or NOS/BE. See tables E-2 and E-3 (NOS EQPDECK Entries for Peripherals and NOS/BE CMR Entries for Peripherals, respectively) for more information about toggling element ownership.

  If the element is a tape channel:

  - If this is the only channel to a tape subsystem, the entire subsystem is returned to the NOS or NOS/BE system. This includes the PP(s), the controller (if applicable), and the associated tape unit.

  - If another channel to the tape subsystem is defined in the NOS/VE active configuration and that channel is in the ON state, only the PP(s) and the channel affected by the state change are returned to the NOS or NOS/BE system.

  If the element is a tape controller:

  - All channels and units that are only connected to the controller and that are in the ON or DOWN state are returned to the NOS or NOS/BE system, provided that none of the storage devices are reserved or assigned to a NOS/VE job.

  If the element is a tape unit:

  - The element is logically disabled so that the associated NOS/VE driver or drivers process no further requests for the element. The element is also no longer a candidate for reservation or assignment to a NOS/VE job. The tape unit is returned to the NOS or NOS/BE system; however, NOS must use its own channel and controller to access the tape unit.

When you change the state of a channel or a controller from ON to OFF, other elements may be returned to the NOS or NOS/BE system even though their state remains ON in the NOS/VE configuration. Therefore, to reverse the effect of changing the state of a channel or controller to OFF, you need only return that element to the ON state. NOS/VE automatically acquires from NOS or NOS/BE the necessary subsystem elements; these subsystem elements are still in the ON state in the NOS/VE configuration.

For example, an operator changes the state of a controller from ON to OFF, causing its associated channel, PP, and tape units to be returned to the NOS or NOS/BE system. Later, when the operator returns the controller to the ON state, the previously mentioned resources are reacquired. In this example, if the operator wants to reacquire all but one of the tape units, the operator must change the state of the unwanted element to OFF before returning the state of the controller to ON.

- The effects of changing the state of an element from OFF to ON are the reverse of changing the state of an element from ON to OFF. However, the following exception applies: when a disk unit is turned ON by the CHANGE_ELEMENT_STATE subcommand, the volume mounted on the unit is not immediately available to the system for file access and allocation. However, set members are implicitly activated when the operator terminates the LCU session that caused the state change. Thus, you can turn ON an existing set member's unit, initialize the volume, and add the volume to the set. However, you must do this in one LCU session. NOS/VE does not allow an existing set member to be initialized after deadstart unless this is the first time that its disk unit has been turned ON since the last deadstart.

- When the state of a channel is changed to DOWN or OFF, NOS/VE automatically reconfigures to use a redundant channel to any of the elements connected to the original channel. When the channel is set to ON again, the original configuration is reinstated.

- When the state of a $FA7B5_A (9836 or 9853) disk controller is changed to DOWN or OFF, NOS/VE automatically reconfigures to use a redundant controller to any of the units connected to the original controller. When the controller is set to ON again, the original configuration is reinstated.

- If the first record of a tape volume cannot be read due to a hardware problem, NOS/VE automatically downs the associated tape unit. The tape unit is inaccessible to all jobs except maintenance jobs.

- NOS/VE automatically downs a network communications device (MDI, MTI, or ICA) that experiences a hardware failure.

Examples

- This example removes control of a 639 tape subsystem (channel CH6, tape unit equipment number 52) from NOS/VE and returns it to NOS. Enter the following commands at the NOS/VE system console:

```
logical_configuration_utility
  change_element_state element=ch6 state=off
  quit
```

  The channel and the 639-1 tape unit are returned to NOS. Enter the following commands at the NOS system console to make the tape unit available:

```
UP,CH6.
ON,EQ=52.
IDLE,MAG.
MAG.
```

  For 639 and 698 tape subsystems, MAGNET should be idled and restarted in order to force the loading of the NOS conversion tables. (This is not required for the 679 tape subsystem.)

- This example moves control of a 679 tape subsystem (channel CH33, EST ordinal 42) to NOS/VE from NOS/BE. Enter the following commands at the NOS/BE console:

```
OFF,42.
DOWN,CH33.
```

  Enter the following commands at the NOS/VE system console:

```
logical_configuration_utility
  change_element_state element=ch33 state=on
  quit
```

  NOS/VE may now use the tape unit.

CHANGE_MS_CLASS Subcommand

Purpose

Changes the mass storage class or classes assigned to an online mass storage volume. By default, a volume is assigned to all classes (A to Z). Refer to Remarks below for a description of the mass storage classes.

Format

CHANGE_MS_CLASS or
CHANGE_MS_CLASSES or
CHAMSC or
CHAMC
  RECORDED_VSN=list of name or **keyword**
  DELETE_CLASS=list of name or *keyword*
  ADD_CLASS=list of name or *keyword*
  STATUS=status variable

Parameters

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

Specifies the 6-character volume serial number (VSN) of the mass storage volume or volumes to be changed. The VSN is the same name defined for the volume on the INITIALIZE_MS_VOLUME subcommand. For this parameter you can specify a list of VSNs or the keyword ALL. This parameter is required. You cannot specify ALL for both the RECORDED_VSN parameter and for the DELETE_CLASS parameter. If any volumes identified by this parameter are inaccessible, an abnormal status is returned.

DELETE_CLASS or *DELETE_CLASSES* or *DC*

Specifies a mass storage class or classes to be deleted from the list of classes associated with the specified mass storage volumes. The name or names specified must be single alphabetic characters from B to Z (you cannot delete mass storage class A). No message is issued if this mass storage class is not currently associated with any of the specified volumes.

The keyword ALL deletes all mass storage classes (except the required class A) associated with the specified volumes; however, the keyword ALL must not be specified for both the DELETE_CLASS parameter and the RECORDED_VSN parameter. By default, no mass storage classes are deleted from the volumes.

ADD_CLASS or *ADD_CLASSES* or *AC*

Specifies a mass storage class or classes to be added to the list of classes associated with the specified mass storage volumes. The name or names specified must be single alphabetic characters from A to Z. No message is issued if this mass storage class is already associated with any of the specified volumes. You can specify the keyword ALL to add all 26 classes. By default, no mass storage classes are added.

STATUS

Returns the completion status for this subcommand.

Remarks

- To use this subcommand, you must have a CONFIGURATION_ ADMINISTRATION validation capability and be in a System Operator Utility session. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

- If you specify both the DELETE_CLASS and ADD_CLASS parameters, the DELETE_CLASS parameter is processed first.

- NOS/VE supports up to 26 mass storage classes. Each class is identified by a single letter of the alphabet. The classes have meanings that are predefined by NOS/VE. The class definitions are described in Mass Storage Volume Class Membership earlier in this chapter.

  You can assign files to these classes explicitly using the REQUEST_ MASS_STORAGE command. Refer to appendix D, Assigning Files to a Specific Device, for more information about file assignment.

  If a file is to be assigned to class Q and no member volumes have space available, NOS/VE automatically adds another device to class Q. For all other classes, if no member volumes have space available, NOS/VE suspends the task requiring disk space until adequate space becomes available.

- For more information about how to assign mass storage classes for optimum fault tolerance characteristics, see Part IV of this manual, Fault Tolerance Configuration and Failure Analysis.

- This subcommand performs the following validation before putting the changes into effect. An error is returned if all these conditions are not met:

  - Each volume specified on this subcommand must be mounted on elements that are in the ON state in the active configuration.

  - Each volume specified must be online; that is, it must be initialized and accessible to the system.

  - The system device must belong to classes A and Q.

  - Every volume must belong to class A.

- The DELETE_CLASSES=ALL parameter causes the volume to be removed from membership in all classes except class A.

- You must first initialize a volume (using INITIALIZE_MS_VOLUME or INITIALIZE_SYSTEM_DEVICE) before you can change its mass storage class membership. If you plan to change a volume's class membership to something other than the default classes (A to Z), we recommend you do so before adding the volume to a mass storage set (using the ADD_VOLUME_TO_SET subcommand).

- Once a file is assigned to a volume, the file may obtain additional space from the volume even though the volume may no longer belong to the class that caused the original file assignment.

- Removing a volume from a class prevents new files or catalogs of that class from being created on the volume and prevents other files from overflowing to the volume.

- Use the REQUEST_MASS_STORAGE command or the RMP$REQUEST_MASS_STORAGE program interface to explicitly assign files to classes U to Z (reserved for site analyst use). For more information, see appendix D, Assigning Files to a Specific Device.

- Class membership is a logical attribute of the mass storage volume and is reinstated after a system interrupt. Class membership moves with the volume if the volume is moved from one disk unit to another (while the system is offline).

Examples

- The following sequence of subcommands initializes a volume to be used for user permanent and temporary files, and for classes M and N, respectively. The volume remains a member of classes A, D to I, and O to P for potential use in subsequent versions of NOS/VE.

```
initialize_ms_volume element=blue_disk_42 recorded_vsn=vsn042
change_ms_class recorded_vsn=vsn042 delete_classes=(C,J,K,L,Q)
add_volume_to_set member_vsn=vsn042
```

- The following subcommand removes classes C, J, K, L, M, and N from the system disk volume. In addition, the system disk volume remains a member of class Q and classes A, D to I, O to P, and R to Z. It remains a member of classes A, D to I, and O to P for potential use in future versions of NOS/VE.

```
change_ms_class recorded_vsn=vsn001 ..
  delete_classes = (C,J,K,L,M,N)
```

CHANGE_MS_VOLUME Subcommand

Purpose

Changes the default values for allocation size and transfer size for a disk volume.

Format

CHANGE_MS_VOLUME or
CHANGE_MS_VOLUME or
CHAMV or
CHAMSV
  RECORDED_VSN=list of name or **keyword**
  ALLOCATION_SIZE=keyword
  TRANSFER_SIZE=keyword
  STATUS=status variable

Parameters

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

Specifies the 6-character volume serial number (VSN) of the mass storage volume or volumes to be changed. The VSN is the same name defined for the volume on the INITIALIZE_MS_VOLUME subcommand. You must specify a list of VSNs or the keyword ALL. This parameter is required.

ALLOCATION_SIZE or *AS*

Specifies the allocation size to be used as the new volume default. Allocation size is the amount of contiguous mass storage space, in bytes, to be assigned to the file each time additional space is needed. Files assigned to the volume that do not request a particular allocation size are given the volume default. You can specify one of the following keywords; the default is $16K (16,384 bytes):

$16K
$32K
$64K
$128K
$256K
$512K
CYLINDER or C

The keyword CYLINDER implies the number of bytes on one cylinder of the volume. For more information about allocation size, see the description of the INITIALIZE_MS_VOLUME subcommand later in this chapter.

TRANSFER_SIZE or *TS*

Specifies the transfer size to be used as the new volume default. Files assigned to the volume that do not request a particular transfer size are given the volume default. Transfer size for a file determines the number of bytes that are read during prestreaming or streaming modes of operation. For more information about page streaming, see the NOS/VE System Performance and Maintenance manual, Volume 1.

You can specify one of the following keywords:

$16K
$32K
$64K
$128K
$256K
$512K
CYLINDER or C

The keyword CYLINDER implies the number of bytes on one cylinder of the volume. All disk units except the model 887 have a default transfer size of $16K (16,384 bytes). The default for the 887 disk unit is $32K (32,768 bytes).

For more information about transfer size, see the description of the INITIALIZE_MS_VOLUME subcommand later in this chapter.

STATUS

Returns the completion status for this subcommand.

Remarks

- To use this subcommand, you must have a CONFIGURATION_ADMINISTRATION validation capability and be in a System Operator Utility session. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- This subcommand does not affect the allocation size and transfer size values of existing files. It affects only future files.

Examples

The following example changes the default allocation size to 32,768 bytes for the volume having a recorded VSN of NVE100:

```
change_ms_volume recorded_vsn=nve100 allocation_size=$32K
```

CREATE_SET Subcommand

Purpose Creates a new mass storage set other than the system set.

Format

```
CREATE_SET or
CRES
   SET_NAME=name
   MASTER_VSN=name
   RECOVER_SET=boolean
   STATUS=status variable
```

Parameters

SET_NAME or SN

Specifies the name of the set to be created. This parameter is required.

MASTER_VSN or MVSN or MV

Specifies the volume serial number (VSN) of the master volume of the set being created. This parameter is required.

RECOVER_SET or RS

Specifies whether recovery is to be performed on the set. Recovery is necessary when the master volume has been initialized. The default is FALSE.

TRUE

The set must be recovered using the permanent file maintenance command RESTORE_UNRECONCILED_CATALOGS.

FALSE

The master volume is not reloaded, and the other set members are not recovered.

Remarks

- To use this subcommand, you must have a CONFIGURATION_ADMINISTRATION validation capability and be in a System Operator Utility session. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- The system set is implicitly defined by the INITIALIZE_SYSTEM_DEVICE system core command.
- A mass storage set consists of a master volume and any number of member volumes.
- The master volume must already be initialized before it can be specified by this subcommand.
- The set specified by the SET_NAME parameter must not already be active on the mainframe.
- Recovery of a set using the RECOVER_SET=TRUE parameter takes place prior to restoring missing catalogs for the set.
- For more information about mass storage sets, see Mass Storage Sets earlier in this chapter.

DEFINE_HOST_NETWORK Subcommand

Purpose Defines a unique network identifier by which the NOS/VE host is known in the network.

NOTE

The DEFINE_HOST_NETWORK subcommand is a network definition subcommand. It must be included in a network configuration file (that is, the input file to an INSTALL_NETWORK_CONFIGURATION or VERIFY_NETWORK_CONFIGURATION subcommand). You cannot enter it interactively.

Format

DEFINE_HOST_NETWORK or
DEFHN
  NETWORK=integer

Parameters NETWORK or N

Specifies the network identifier by which the NOS/VE host is known in the network. The value specified must be an integer between 1 and 65,535. This parameter is required.

Remarks

- You can enter this subcommand only from the system console. You cannot enter it from an interactive terminal.
- You must include this subcommand in your network configuration file. You may specify it only once.
- For a more detailed discussion of CDCNET network identifiers, see Creating and Installing a Network Configuration File earlier in this chapter.

DEFINE_MS_FLAW Subcommand

Purpose Records a physical location or a range of locations on a mass storage volume as flawed.

Format **DEFINE_MS_FLAW** or
DEFMSF
  RECORDED_VSN=name
  CYLINDER=integer
  TRACK=integer
  SECTOR=integer
  STATUS=status variable

Parameters **RECORDED_VSN** or **RVSN** or **RV**

Specifies the 6-character volume serial number (VSN) of the mass storage volume on which the flaw is to be recorded. The VSN is the same name defined for the volume on the INITIALIZE_MS_VOLUME subcommand. This parameter is required.

CYLINDER or **C**

Specifies a cylinder number. If you omit the track and sector parameters, the entire cylinder specified is recorded as flawed. This parameter is required.

TRACK or *T*

Specifies a track number. If you omit the SECTOR parameter, an entire track is recorded as flawed. Otherwise, only one sector is recorded as flawed.

SECTOR or *S*

Specifies a sector number. If you specify this parameter, both the CYLINDER and TRACK parameters are required.

STATUS

Returns the completion status for this subcommand.

Remarks

- To use this subcommand, you must have a CONFIGURATION_ADMINISTRATION validation capability and be in a System Operator Utility session. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

- The term *flaw* refers to the mass storage space surrounding the volume defect. NOS/VE treats this space as a device allocation unit (DAU). The effect of setting the flaw is that NOS/VE no longer assigns the defective space to any new files. In addition, NOS/VE may relocate the data in an existing file to avoid loss of data. For more information, refer to Volume Defect Management Recommendations in appendix E, Supported Hardware Products.

- If you need to manually flaw an uncorrectable media defect, we recommend you issue the DEFINE_MS_FLAW subcommand before you initialize the volume. This ensures that NOS/VE does not allocate the defective location for system files created by the initialization process. If the volume is already initialized, NOS/VE automatically flaws an uncorrectable media defect.

- If the affected space is already assigned to a file and the file to which it is assigned is deleted, this subcommand prevents the flawed space from being assigned to another file.

- This subcommand only alters a NOS/VE data structure on the volume. It does not physically relocate sectors on the volume.

- The affected space remains flawed until a corresponding REMOVE_MS_FLAW subcommand is issued. In addition, you can remove all flaws during volume initialization by setting the RETAIN_MS_FLAWS parameter to FALSE on the INITIALIZE_MS_VOLUME command.

- NOS/VE does not allow you to define a flaw in the following places:

  - Designated maintenance cylinder

  - Confidence test cylinder

  - Manufacturing cylinder

  - Cylinder reserved for CIP (CYBER Initialization Package), CTI (Common Test and Initialization), HIVS (Hardware Initialization and Verification Software), MSL (Maintenance Software Library), or the common disk area

- You can define a flaw for a volume at any time. If the volume identified by the RECORDED_VSN parameter has not yet been initialized or has been inactive since the last deadstart, the following restrictions apply:

  - The defined flaw is saved in memory and put into effect when the volume is initialized or activated. However, the pending flaw is not saved across a system interruption or a deadstart.

  - Since the identity of the volume and its physical characteristics are unknown until the volume is reinstated, the CYLINDER, TRACK, and SECTOR parameters are not validated during execution of this subcommand. Therefore, if these parameters are erroneous, no indication of this condition is given at the time the volume is reinstated.

  - To ensure that the deferred flaw is defined, issue the DISPLAY_MS_FLAW subcommand after the volume is reinstated. If the flaw is not defined, reissue the DEFINE_MS_FLAW subcommand.

- Use the DISPLAY_MS_FLAW subcommand to display a list of media defects for a mass storage volume.

- A CM0201 statistic is emitted whenever a mass storage flaw is set either manually or automatically.

DEFINE_NETWORK_CONNECTION Subcommand

Purpose

Defines NOS/VE connections to any network.

NOTE

The DEFINE_NETWORK_CONNECTION subcommand is a network definition subcommand. It must be included in a network configuration file (that is, the input file to an INSTALL_NETWORK_CONFIGURATION or VERIFY_NETWORK_CONFIGURATION subcommand).

Format

DEFINE_NETWORK_CONNECTION or
DEFNC
  CONNECTED_SYSTEM=name
  SYSTEM_IDENTIFIER=integer

Parameters

CONNECTED_SYSTEM or **CS**

Specifies the element name (from the physical configuration file) of the CDCNET communications device that provides access to the network. This parameter is required.

SYSTEM_IDENTIFIER or *SI*

Specifies an integer indicating the lower six hexadecimal digits of the ICA-II's 12-digit system identifier. The upper six hexadecimal digits of the system ID are always 080025 and are supplied by the command processor. The unique system identifier appears on a label attached to the mainframe. This parameter is required when you define an ICA-II.

Remarks

For a more detailed discussion of CDCNET network identifiers, see the section called Creating and Installing a Network Configuration File earlier in this chapter.

DISPLAY_MAINFRAME_CONFIGURATION Subcommand

Purpose Displays information about one or more elements in the mainframe's active configuration.

Format **DISPLAY_MAINFRAME_CONFIGURATION** or
DISMC
ELEMENT=list of name or *keyword*
DISPLAY_OPTION=list of keyword
IOU=name
OUTPUT=file
STATUS=status variable

Parameters *ELEMENT* or *ELEMENTS* or *E*

Specifies either the names of the elements or the class of elements for which the definition is to be displayed. The default is ALL, which means that the definitions of all elements in the active configuration are displayed. If you specify a class of elements, the definitions for all elements within that class are displayed. The following are names of the classes:

$CHANNEL
$CHANNEL_ADAPTOR or $CA
$COMMUNICATIONS_ELEMENT or $CE
$CONTROLLER
$EXTERNAL_PROCESSOR or $EP
$STORAGE_DEVICE or $SD

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Specifies the information to be displayed for each element specified by the ELEMENT parameter. You can specify the following keywords for this parameter; the default is ALL:

ACTIVE_PATHS or AP

Displays the active physical paths for each element specified by the ELEMENT parameter. A path is active if all elements in the path are in the ON state and the path is connected to an actively configured channel. The names and physical addresses of all the elements in each path are displayed.

ALL

Displays all information from all display option keywords.

APPLICATION_INFORMATION or AI

Displays information to be used by an application that logically configures the element.

CONNECTION_STATUS or CS

Displays the status of connections between elements specified by the ELEMENT parameter. Each connection between pairs of elements in a physical path is displayed on a separate line, along with the status of the connection. An ACTIVE connection status means that both of the elements in the connection are in the ON state. An INACTIVE connection status means that one or both of the elements in the connection are not in the ON state. The element that is logically closer to the mainframe (the upline connection) is always displayed first.

ELEMENT_IDENTIFICATION or EI

Displays all identification information for each element specified by the ELEMENT parameter. This information is the value specified for the ELEMENT_IDENTIFICATION parameter of the element's DEFINE_ELEMENT subcommand.

INACTIVE_PATHS or IP

Displays all inactive physical paths for each element specified by the ELEMENT parameter. A path is inactive if it contains one or more elements that are in an OFF or DOWN state, or if the channel is not actively configured. The names and physical addresses of all the elements in each path are displayed.

IOU_PROGRAM_NAME or IPN

Displays the IOU program name associated with each element specified by the ELEMENT parameter.

MS_CLASS or MSC or MC

Displays the mass storage class of the volume mounted on the element or elements specified by the ELEMENT parameter.

PHYSICAL_CONNECTIONS or PC

Displays the names of the elements defined by the CENTRAL_MEMORY_CONNECTIONS, IOU_CONNECTIONS, and PERIPHERAL_CONNECTIONS parameters of the PCU subcommand DEFINE_ELEMENT. Only the names of the elements that are directly connected to the element are displayed.

PHYSICAL_PATHS or PP

Displays all physical paths that include the element specified by the ELEMENT parameter, regardless of status. This is equivalent to the sum of the displays produced by the ACTIVE_PATHS and INACTIVE_PATHS options. The names and physical addresses of all elements in each path are displayed.

SERIAL_NUMBER or SN

Displays the serial number associated with each element specified by the ELEMENT parameter. The serial number is the same as specified for the SERIAL_NUMBER parameter of the element's DEFINE_ELEMENT subcommand.

SITE_INFORMATION or SI

Displays information provided and maintained by the site.

STATE or S

Displays information indicating whether the specified element is ON or OFF.

IOU

Specifies the name of the IOU associated with the channel to be displayed. This parameter is ignored unless you name a specific channel in the ELEMENT parameter of this subcommand and there are multiple IOUs on the mainframe executing this subcommand. Values you can specify for this parameter are IOU0 and IOU1. The default is IOU0.

OUTPUT or *O*

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

STATUS

Returns the completion status for this subcommand.

Remarks

To use this subcommand, you must be validated for at least one of the following capabilities and be in a System Operator Utility session:

CONFIGURATION_ADMINISTRATION
REMOVABLE_MEDIA_OPERATION
SYSTEM_DISPLAYS
SYSTEM_OPERATION

For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

Examples

- The following example lists information about a hypothetical tape controller, PINK_698_C1:

```
display_mainframe_configuration element=pink_698_c1

CONTROLLER ELEMENT: PINK_698_C1
    PRODUCT IDENTIFICATION:   $698_10
    SERIAL NUMBER: 283
    IOU PROGRAM NAME: E1C7021
    STATE: OFF
    EQUIPMENT NUMBER:      0(10)
    CHANNEL CONNECTIONS:   IOU0/CH26
    STORAGE DEVICE CONNECTIONS:   P60  P61  P62  P63
    CONNECTION STATUS:
      IOU0/CH26.PINK_698_C1                           INACTIVE
      PINK_698_C1.P60                                 INACTIVE
      PINK_698_C1.P61                                 INACTIVE
      PINK_698_C1.P62                                 INACTIVE
      PINK_698_C1.P63                                 INACTIVE
    ACTIVE PATHS:
      NONE
    INACTIVE PATHS:
      IOU0.CH26.PINK_698_C1.P60                       IOU0.CH26.C0.U0
      IOU0.CH26.PINK_698_C1.P61                       IOU0.CH26.C0.U1
      IOU0.CH26.PINK_698_C1.P62                       IOU0.CH26.C0.U2
      IOU0.CH26.PINK_698_C1.P63                       IOU0.CH26.C0.U3
```

- The following example lists information about a hypothetical disk unit, PINK_9853_1:

```
display_mainframe_configuration element=pink_9853_1

STORAGE DEVICE ELEMENT: PINK_9853_1
    PRODUCT IDENTIFICATION:  $9853_1
    SERIAL NUMBER: 413
    STATE: ON
    MASS STORAGE CLASS:   A  B  D  E  F  G  H  I  M  O  P  R  S  T  U  V  W  X
         Y  Z
    UNIT NUMBER:       1(10)
    CONTROLLER CONNECTIONS:   PINK_FA7B5_1
    CONNECTION STATUS:
      IOU0/CCH4A.PINK_FA7B5_1                              ACTIVE
      PINK_FA7B5_1.PINK_9853_1                             ACTIVE
    ACTIVE PATHS:
      IOU0.CCH4A.PINK_FA7B5_1.PINK_9853_1                  IOU0.CCH4A.C0.U1
    INACTIVE PATHS:
      NONE
```

DISPLAY_MS_CLASS Subcommand

Purpose Displays the mass storage classes to which the specified volumes belong.

Format **DISPLAY_MS_CLASS** or
DISPLAY_MS_CLASSES or
DISMSC
RECORDED_VSN=keyword or *list of name*
OUTPUT=file
STATUS=status variable

Parameters *RECORDED_VSN* or *RECORDED_VSNS* or *RVSN* or *RV*

Specifies the 6-character volume serial number (VSN) of the mass storage volume or volumes whose class membership is to be displayed. You can enter the keyword ALL to display the class membership of all volumes. The default is ALL.

OUTPUT or *O*

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

STATUS

Returns the completion status for this subcommand.

Remarks To use this subcommand, you must be validated for at least one of the following capabilities and be in a System Operator Utility session:

CONFIGURATION_ADMINISTRATION
REMOVABLE_MEDIA_OPERATION
SYSTEM_DISPLAYS
SYSTEM_OPERATION

For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

DISPLAY_MS_FLAW Subcommand

Purpose Displays the list of media flaws for a mass storage volume.

Format **DISPLAY_MS_FLAW** or
DISPLAY_MS_FLAWS or
DISMSF
RECORDED_VSN=keyword or *list of name*
DISPLAY_OPTION=keyword
OUTPUT=file
STATUS=status variable

Parameters *RECORDED_VSN* or *RECORDED_VSNS* or *RVSN* or *RV*

Specifies the 6-character volume serial number (VSN) of the mass storage volume or volumes whose flaw lists are to be displayed. The VSN is the same name defined for the volume on the INITIALIZE_MS_VOLUME subcommand. You can specify a list of VSNs or the keyword ALL for this parameter. The default is ALL.

DISPLAY_OPTION or *DO*

Specifies the content of the display to be provided. You can specify one of the following keywords; the default is EFFECT:

EFFECT

Displays the effect of each flaw in terms of a range of cylinders and tracks or sectors involved. In addition, the type of flaw is shown. For each flaw, the physical address is displayed. If an entire cylinder or track is flawed, only one line is displayed. The value RESERVED in a line identifies mass storage space reserved for use by Control Data or the manufacturer of the storage device. For a list of reserved areas, see the Remarks section for this subcommand. Lines not containing the value RESERVED were flawed either automatically by NOS/VE when an unrecovered media defect occurred, or manually by a DEFINE_MS_FLAW subcommand.

SOURCE

Displays the cylinder, track, and sector information that you can use to reproduce each flaw (by the DEFINE_MS_FLAW subcommand) in case the flaw information recorded on the storage medium is lost or damaged. Only the flaws set by the DEFINE_MS_FLAW subcommand or those automatically set by NOS/VE are displayed.

OUTPUT or *O*

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

STATUS

Returns the completion status for this subcommand.

Remarks

- To use this subcommand, you must be validated for at least one of the following capabilities and be in a System Operator Utility session:

  CONFIGURATION_ADMINISTRATION
  REMOVABLE_MEDIA_OPERATION
  SYSTEM_DISPLAYS
  SYSTEM_OPERATION

  For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

- The following are reserved areas on a mass storage volume:

  - Cylinders reserved for offline maintenance software such as CIP (CYBER Initialization Package), CTI (Common Test and Initialization), HIVS (Hardware Initialization and Verification Software), MSL (Maintenance Software Library), or the common disk area.
  - Cylinders reserved for online maintenance confidence testing.
  - Cylinders reserved to NOS/VE for confidence testing.
  - Flaws identified in the device's utility map or factory flaw map.

- This subcommand displays device allocation unit (DAU) addresses in terms of their physical locations. The starting address displayed may not match exactly what you defined if the cylinder/track/sector you flawed appears in the middle of a DAU. That is, the address displayed is the beginning of the DAU.

Examples

The following example displays media flaws for hypothetical mass storage volume V87H01:

```
LCU/display_ms_flaw rvn=v87h01 do=effect

         MEDIA FLAWS FOR VSN - V87H01

    CYL  TRK SEC  THROUGH  CYL  TRK SEC  TYPE      FIRST DAU  LAST DAU

    881    0  0            883    3  37  RESERVED     33478     33591
LCU/
```

DISPLAY_MS_VOLUME Subcommand

Purpose

Displays the current default values for allocation size and transfer size for a disk volume.

Format

DISPLAY_MS_VOLUME or
DISPLAY_MS_VOLUME or
DISMV or
DISMSV
RECORDED_VSN=list of name or *keyword*
DISPLAY_OPTIONS=list of name or keyword
OUTPUT=file
STATUS=status variable

Parameters

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

Specifies the 6-character volume serial number (VSN) of the mass storage volume or volumes whose default attributes are to be displayed. You can specify a list of VSNs or the keyword ALL. The default is ALL.

DISPLAY_OPTIONS or *DO*

Specifies the content of the display to be provided. You can specify one of the following options; the default is ALL:

ALLOCATION_SIZE or AS
TRANSFER_SIZE or TS
ALL

OUTPUT or *O*

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

STATUS

Returns the completion status for this subcommand.

Remarks

To use this subcommand, you must be validated for at least one of the following capabilities and be in a System Operator Utility session:

CONFIGURATION_ADMINISTRATION
REMOVABLE_MEDIA_OPERATION
SYSTEM_DISPLAYS
SYSTEM_OPERATION

For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

Examples

The following example displays the default allocation size and transfer size values for the volume having a recorded VSN of NVE100:

```
display_ms_volume recorded_vsn=nve100 display_option=all

  RVSN       ALLOCATION SIZE     TRANSFER SIZE

  NVE100     16384               16384
```

DISPLAY_NETWORK_CONFIGURATION Subcommand

Purpose Displays the networks to which NOS/VE is connected.

Format **DISPLAY_NETWORK_CONFIGURATION** or
DISNC
DISPLAY_OPTIONS=keyword
OUTPUT=file
STATUS=status variable

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

Specifies which options are to be displayed. By default, all options are displayed. You can specify the following keywords:

NETWORKS or NETWORK or N

Displays the networks to which this NOS/VE system is connected.

ALL

All options are displayed.

OUTPUT or *O*

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

STATUS

Returns the completion status for this subcommand.

Remarks

- To use this subcommand, you must be validated for at least one of the following capabilities and be in a System Operator Utility session:

  CONFIGURATION_ADMINISTRATION
  REMOVABLE_MEDIA_OPERATION
  SYSTEM_DISPLAYS
  SYSTEM_OPERATION

  For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

- This command displays information for the current active configuration only. If a network configuration has been installed but not activated (that is, if NAM/VE has not been activated since the network was installed), this command does not display information for the installed configuration.

Examples The following example displays the networks to which a sample NOS/VE system is connected:

```
display_network_configuration display_option=networks

  HOST NETWORK: 41988(10)

  CONNECTED SYSTEM: CA_2629_105
  SYSTEM IDENTIFIER: 80025400148(16)
```

INITIALIZE_MS_VOLUME Subcommand

Purpose Initializes the volume that is mounted on a disk unit.[1] Before a volume can be used by NOS/VE, it must be initialized.[2] A volume can be initialized only if it is not active. A deadstart is required to initialize an active volume.

CAUTION

Initializing a volume that is a member of a set destroys all files recorded in whole or in part on the volume. If the volume contains catalogs that describe files or catalogs on other volumes, those files and catalogs are also lost.

Format **INITIALIZE_MS_VOLUME** or
INIMV or
INIMSV
  ELEMENT=name
  RECORDED_VSN=name
  RETAIN_DEVICE_FLAWS=boolean
  ALLOCATION_SIZE=keyword
  TRANSFER_SIZE=keyword
  STATUS=status variable

Parameters ELEMENT or E

Specifies the element name of the disk unit on which the volume is mounted. This element must be in the ON state in the active configuration and must not already be active. This parameter is required.

RECORDED_VSN or RVSN

Specifies the 6-character volume serial number (VSN) to be written on the mass storage volume's label. Since the system recognizes a volume by its VSN, it is important that each volume at a site have a unique VSN. This parameter is required.

RETAIN DEVICE_FLAWS or *RDF*

Specifies whether device flaws that are currently defined for the volume should be retained. The default is TRUE. The flaws recorded in the utility map of a 844-4 or 885-1 device are not affected by this parameter; they are retained in any case.

TRUE

Device flaws are retained if the system is able to read an existing, valid NOS/VE label and the flaws can be read correctly from the device. If the flaws can be retained on a 895-2 device, the device will not be soft-sectored. If flaws cannot be retained, a message is sent to the system console and the volume initialization continues.

1. Before a volume can be initialized, it must be formatted. NOS/VE does not format volumes (except the 895-2 disk unit). Volumes manufactured by Control Data are formatted at the factory. For more information about formatting volumes, contact your Control Data customer engineer.

2. If you want to use the 895-2 disk unit as a CIP device, you should have your CE initialize the maintenance and diagnostic sectors (using the Disk Format Utility (DFU)) before you enter the INITIALIZE_MS_VOLUME subcommand. Executing the DFU after you initialize the disk erases the contents of the disk, since DFU initializes the entire disk.

FALSE

Device flaws are not retained. 895-2 devices will be soft-sectored to ensure that, if CIP has been removed, the cylinders previously reserved for CIP are formatted in the NOS/VE sector size.

ALLOCATION_SIZE or *AS*

Specifies the allocation size to be used as the volume default. Allocation size is the amount of contiguous mass storage space, in bytes, to be assigned to the file each time additional space is needed. Files assigned to the volume that do not request a particular allocation size are given the volume default. You can specify one of the following keywords; the default is $16K (16,384 bytes):

$16K
$32K
$64K
$128K
$256K
$512K
CYLINDER or C

The keyword CYLINDER implies the number of bytes on one cylinder of the volume. For more information about allocation size, see Remarks.

TRANSFER_SIZE or *TS*

Specifies the transfer size to be used as the volume default. Files created without specification of transfer size are given the volume default. Transfer size for a file determines the number of bytes that are read during prestreaming or streaming modes of operation. For more information about page streaming, see the NOS/VE System Performance and Maintenance manual, Volume 1.

You can specify one of the following keywords:

$16K
$32K
$64K
$128K
$256K
$512K
CYLINDER or C

The keyword CYLINDER implies the number of bytes on one cylinder of the volume. All disk units except the model 887 have a default transfer size of $16K (16,384 bytes). The default for the 887 disk unit is $32K (32,768 bytes).

For more information about transfer size, see Remarks.

STATUS

Returns the completion status for this subcommand.

Remarks

- To use this subcommand, you must have a CONFIGURATION_ADMINISTRATION validation capability and be in a System Operator Utility session. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

- If a file is created (REQUEST_MASS_STORAGE command) and no allocation size is requested, the file is given the volume default for the initial volume to which it is assigned. If an allocation size is requested when the file is created, the request is honored but restricted by the value of the MAXIMUM_ALLOCATION_SIZE system attribute. This attribute is described in the NOS/VE System Performance and Maintenance manual, Volume 1.

- We recommend you specify an allocation size greater than 16K, provided the disk unit has a fast transfer rate (for example, a model 887 disk unit) or if the files on the volume are accessed sequentially. Using a larger allocation size ensures that more data can be written sequentially on the volume without incurring further seek time and latency time (that is, the file can be streamed).

  Another advantage of specifying a larger allocation size is that the CPU overhead resulting from I/O requests is reduced when the file is accessed sequentially. When the allocation size is doubled, half as many I/O requests are required to access the file sequentially as with the smaller allocation size. This assumes a transfer size that is greater than or equal to the allocation size.

  However, the larger the allocation size, the greater the potential for wasted capacity. Since the length of a file is generally not an exact multiple of the allocation size, half of the last allocation unit of a file tends to be wasted space. In addition, disk volumes do not always have a cylinder size that is an exact multiple of allocation sizes greater than 16K. In general, a default allocation size of 16K is the most efficient way to utilize the volume's capacity.

- Whenever a file must overflow to a new volume, NOS/VE allows the file to overflow only to a volume that supports the file's allocation size. If the allocation size of a file was determined by a REQUEST_MASS_STORAGE command, the file may overflow to any disk unit that supports an allocation size equal to or greater than the file's allocation size. If the allocation size of a file was determined by its initial volume's default and the volume default was a cylinder, the file may overflow only to volumes having the same cylinder capacity as the initial volume.

- We recommend you specify a transfer size greater than 16K if the system is CPU-bound or is configured with a disk unit having a fast transfer rate (such as a model 887 disk unit). Applications in this environment require more buffering of read data because the application's processing rate is slower than the disk unit. In an I/O-bound environment, selecting a transfer size of 16K is sufficient for most types of disk units.

- If the specified transfer size exceeds the allocation size, NOS/VE issues simultaneous multiple read requests; each read request is for an amount of data equal to the allocation size. If the transfer size is less than or equal to the allocation size, NOS/VE issues a single read request.

- The following flaws are affected by the RETAIN_DEVICE_FLAWS parameter:

  - Flaws that were automatically defined by NOS/VE to recognize the presence of the CYBER Initialization Package (CIP) on a preceding initialization of this volume. During installation or removal of the CIP on a NOS/VE volume, the NOS/VE information is lost. Accordingly, Common Test and Initialization (CTI) must be used to correctly install or remove the CIP. After that, you must initialize the volume using the INITIALIZE_MS_VOLUME subcommand with the RETAIN_DEVICE_FLAWS parameter set to FALSE.

  - Flaws that were automatically defined by NOS/VE due to a detected but uncorrected device failure.

  - Flaws that were defined with a DEFINE_MS_FLAW subcommand.

- Depending on the characteristics of the device, the process of initializing a volume includes one or more of the following activities:

  - The NOS/VE system label recorded on the volume is validated. If a valid system label is not found, processing continues. (For further information, see the description of the RETAIN_DEVICE_FLAWS parameter.) If a valid system label is found, the current recorded VSN is displayed and you are asked to confirm the initialization attempt.

  - If the volume is a member of the system set, the system asks you to confirm the initialization attempt. If you decide to proceed, the system automatically removes the volume from the set before the volume is initialized. This is true when a volume is initialized during or after deadstart. An existing set member can be initialized after deadstart only if:

    - The volume's storage device was not set to ON during the last deadstart.

    - The volume is initialized in the first LCU session that sets the storage device back to ON.

  - A unique internal volume identifier (VSN) is generated and stored in the volume label.

  - The site-supplied recorded VSN is stored in the volume label.

  - If the volume requires the initialization of sector addresses (such as the 895-2 storage device), the entire volume is prewritten except for any cylinders reserved to the CTI process and the initialization performed by CIP.

  - The volume is assigned to all mass storage classes A to Z.

  - A CM0100 statistic is placed in the Engineering Log whenever a mass storage volume is initialized.

- Initializing a volume erases the previous contents of the volume.

- You can enter this subcommand anytime the LCU is executed, either during deadstart or during normal system operations. The system executes the INITIALIZE_MS_VOLUME subcommand as soon as you enter it.
- If a volume is physically removable and has been initialized, you can mount it on any active disk unit (that is, in the ON state) before you deadstart NOS/VE. The system reads the volume label and recognizes the volume by its VSN. All NOS/VE volumes must be mounted, the START switch must be pressed, and the volume reach operating speed before NOS/VE can be deadstarted. You cannot move disk volumes during system operation.
- For the 895-2 mass storage unit only, this subcommand causes the disk unit to be formatted prior to initalization. This process requires approximately 5 minutes per disk unit.

Examples Assume you have two disks (with VSNs of B100 and R102) that are not members of the system set. One of the disks has not been initialized. You instruct the system operator to add both of them to the system set and (by default) to retain device flaws. The system operator enters the following commands from the NOS/VE system console:

```
logical_configuration_utility
  initialize_ms_volume element=blue0 recorded_vsn=b100
  add_volume_to_set member_vsn=b100
  add_volume_to_set member_vsn=r102
  quit
```

INSTALL_NETWORK_CONFIGURATION Subcommand

Purpose Verifies and installs a network configuration file. A network configuration file defines a NOS/VE system's connections to one or more CDCNET networks. Verification tests the validity of the network configuration defined in the input file. Installation creates a permanent file that contains the network configuration.

Format **INSTALL_NETWORK_CONFIGURATION** or
INSNC
  INPUT=file
  ERRORS=file
  RETAIN_HIGH_CYCLES=keyword or *integer*
  STATUS=status variable

Parameters INPUT or I

Specifies the name of the file containing the network configuration to be installed. This file consists of a sequence of network definition subcommands. The file may also include SCL control statements, but no other NOS/VE commands are allowed. This parameter is required.

ERRORS or *E*

Specifies the name of the file to which error messages are to be written. The default is $ERRORS.

If a subcommand has a syntax error that can be detected in isolation from other subcommands, the subcommand is echoed to the file followed by an error message. Messages describing errors that cannot be isolated to a particular subcommand are written to the end of the file.

RETAIN_HIGH_CYCLES or *RETAIN_HIGH_CYCLE* or *RHC*

Specifies the number of cycles of the file $SYSTEM.NETWORK. CONFIGURATION (including the one being installed) to be retained by the system. All cycles prior to those specified are deleted. Values that can be specified for this parameter are integer values from 1 to 999 or the keyword ALL. The default is 2.

STATUS

Returns the completion status for this subcommand.

Remarks

- To use this subcommand, you must have a CONFIGURATION_ ADMINISTRATION validation capability and be in a System Operator Utility session. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

- The network definition commands you can include in the input file for this subcommand are DEFINE_NETWORK_CONNECTION and DEFINE_HOST_NETWORK.

- The network configuration file installed by this subcommand is automatically activated the next time NAM/VE is activated. This is true regardless of whether NAM/VE is active at the time the configuration file is installed. The installation of a configuration file has no effect on the active network configuration if NAM/VE is currently active.

QUIT Subcommand

Purpose Terminates the Logical Configuration Utility.

Format **QUIT** or
QUI

Remarks When a user with a CONFIGURATION_ADMINISTRATION or SYSTEM_OPERATION validation capability enters a QUIT subcommand to terminate an LCU session, the utility performs the following validation functions:

1. Any disk volume that is a member of the system set and is mounted on a unit that is set to ON is activated. This enables users to access files or catalogs on the volume.

2. Once all candidate volumes are activated, the utility verifies that every mass storage class has at least one active member volume.

3. If a mass storage class has no active members, an error message is displayed. The operator must resolve the problem before being allowed to exit the utility.

In addition, entering the QUIT subcommand allows another user with the same validation capability within a System Operator Utility session to use the LCU. For more information, refer to Access Controls earlier in this chapter.

REMOVE_MS_FLAW Subcommand

Purpose Removes, from an active volume, a media flaw previously defined automatically by NOS/VE or manually by a DEFINE_MS_FLAW subcommand.

Format **REMOVE_MS_FLAW** or
REMMSF
  RECORDED_VSN=name
  CYLINDER=integer
  TRACK=integer
  SECTOR=integer
  STATUS=status variable

Parameters *RECORDED_VSN* or *RECORDED_VSNS* or *RVSN* or *RV*

Specifies the volume serial number (VSN) of the mass storage volume on which the flaw is to be recorded. The VSN is the same name defined for the volume on the INITIALIZE_MS_VOLUME subcommand. This parameter is required.

CYLINDER or C

Specifies a cylinder number. If you omit the track and sector parameters, the entire cylinder specified is removed. This parameter is required.

TRACK or *T*

Specifies a track number. If you omit the SECTOR parameter, an entire track is removed. Otherwise, only one sector is removed.

SECTOR or *S*

Specifies a sector number. If you specify this parameter, both the CYLINDER and TRACK parameters are required.

STATUS

Returns the completion status for this subcommand.

Remarks

- To use this subcommand, you must have a CONFIGURATION_ ADMINISTRATION validation capability and be in a System Operator Utility session. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- Whenever an uncorrectable media failure is detected, NOS/VE automatically records the defects as flawed. (A media failure is one for which an uncorrectable address or data field checkword (ECC) error is reported by the storage device or storage controller.) Some failure modes of disk subsystems may cause NOS/VE to automatically record unfaulty media as flawed. Therefore, you can use this subcommand to reinstate flawed media after such a hardware failure is corrected.
- If the affected space is already assigned to a file, it remains assigned and the file to which it is assigned is unaffected by this subcommand.

- NOS/VE does not let you remove flaws from the following areas:
  - Designated maintenance cylinder
  - Confidence test cylinder
  - Manufacturing cylinder
  - Cylinder reserved for CIP, (CYBER Initialization Package) CTI (Common Test and Initialization), HIVS (Hardware Initialization and Verification Software), MSL (Maintenance Software Library), or the common disk area)

VERIFY_NETWORK_CONFIGURATION Subcommand

Purpose Verifies the validity of a network configuration. The verification performed by this subcommand is identical to that performed by the INSTALL_NETWORK_CONFIGURATION subcommand.

Format **VERIFY_NETWORK_CONFIGURATION** or
VERNC
  INPUT=file
  ERRORS=file
  STATUS=status variable

Parameters **INPUT or I**

Specifies the name of the file containing the network configuration to be verified. This file consists of a sequence of network definition subcommands. The file may also include SCL control statements, but no other NOS/VE commands are allowed. This parameter is required.

ERRORS or E

Specifies the name of the file to which error messages are to be written. The default is $ERRORS.

If a subcommand has a syntax error that can be detected in isolation of other subcommands, the subcommand is echoed to the file followed by an error message. Messages describing errors that cannot be isolated to a particular subcommand are written to the end of the file.

STATUS

Returns the completion status for this subcommand.

Remarks

- To use this subcommand, you must be validated for at least one of the following capabilities and be in a System Operator Utility session:

  CONFIGURATION_ADMINISTRATION
  REMOVABLE_MEDIA_OPERATION
  SYSTEM_DISPLAYS
  SYSTEM_OPERATION

  For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

- The network definition commands that you can include in the input file for this subcommand are DEFINE_NETWORK_CONNECTION and DEFINE_HOST_NETWORK.

LCU Functions

This section contains descriptions of the following LCU functions:

$ACTIVE_SETS
$ACTIVE_SET_FAMILIES
$ACTIVE_SET_MEMBERS
$CHANNEL_PORT
$ELEMENT
$MASS_STORAGE_CLASS_MEMBERS
$PHYSICAL_ADDRESS
$STORAGE_DEVICE_NAME
$SYSTEM_SET_NAME

To use these functions, you must be validated for at least one of the following capabilities and be in a System Operator Utility session

CONFIGURATION_ADMINISTRATION
REMOVABLE_MEDIA_OPERATION
SYSTEM_DISPLAYS
SYSTEM_OPERATION

For more information about these capabilities and the System Operator Utility, see Validation Capabilities and Access Controls earlier in this chapter.

$ACTIVE_SETS Function

Purpose Returns a list of the names of all the active mass storage sets on the requesting mainframe. The name of the system set is included.

Format **$ACTIVE_SETS**

Parameters None.

Examples The following example returns the names of the active mass storage sets on a hypothetical mainframe:

```
display_value $active_sets
GOLD_SYSTEM
GOLD_USER
```

$ACTIVE_SET_FAMILIES Function

Purpose Returns a list of the names of families assigned to the active mass storage set.

Format **$ACTIVE_SET_FAMILIES**
(set)

Parameters **set**

Specifies the name of the mass storage set whose family names are to be returned. This parameter is required.

Examples The following example returns the names of the families on active mass storage set GOLD_SYSTEM:

```
display_value $active_set_families(gold_system)
$SYSTEM
FAMILY_1
FAMILY_2
```

$ACTIVE_SET_MEMBERS Function

Purpose Returns the element name and the recorded VSN for each disk volume that is a member of the specified set.

Format **$ACTIVE_SET_MEMBERS**
(set)

Parameters **set**

Specifies the name of the mass storage set whose members are to be returned. This parameter is required.

Examples The following example returns the element names and recorded VSNs for active set GOLD_USER:

```
display_value $active_set_members(gold_user)
DISK_9853_1001
VSN001
DISK_9853_1002
VSN002
DISK_9853_1003
VSN003
```

$CHANNEL_PORT Function

Purpose Returns the string value of the channel port that connects a channel to a peripheral element.

Format **$CHANNEL_PORT** or **$CP**
(channel,
downline_element,
iou)

Parameters **channel**

Specifies the name of the channel whose port identification to the downline element is requested. This parameter is required.

downline_element

Specifies the name of the element that can be accessed using the channel port whose identification is requested. This parameter is required.

iou

Specifies the IOU associated with the upline element whose name is specified. This parameter is required if the system has multiple IOUs. The name of the channel is not unique unless qualified by the IOU identification. If you specify this parameter in a single IOU system, its value must be IOU0.

Remarks The value returned by this function is either A or B for a channel that supports a dual-port option, and an empty string for a channel that does not support this option.

Examples

- In the following example, $CHANNEL_PORT returns an empty string value because channel CH4 does not support a dual-port option:

```
display_value $channel_port(ch4,blue_disk_controller)
"returns an empty string"
```

- In the following example, $CHANNEL_PORT returns a string value of A because CCH4 supports a dual-port option:

```
display_value $channel_port(cch4,blue_disk_controller)
A
```

$ELEMENT Function

Purpose Returns the value of an element attribute.

Format **$ELEMENT**
(element,
attribute,
iou)

Parameters **element**

Specifies the name of the element whose attribute value you want returned. This parameter is required.

attribute

Specifies the name of the element attribute whose value you want returned. This parameter is required. You can specify the following attributes:

APPLICATION_INFORMATION or AI
ASSIGNED_JOB_IDENTIFICATION or AJI
ASSIGNED_TO_JOB or ATJ
CONCURRENT_MAINTENANCE_COUNT or CMC
DEDICATED_MAINTENANCE_ACTIVE or DMA
DENSITIES_IN_OPERATION or DIO
DEVICE_ALLOCATION_UNIT_SIZE or DAUS
DEVICE_CLASS or DC
ELEMENT_CAPABILITY or EC
ELEMENT_EXISTS or EE
ELEMENT_IDENTIFICATION or EI
ELEMENT_STATE or ES
ELEMENT_TYPE or ET
EXTERNAL_VSN or EVSN
IOU_PROGRAM_NAME or IOUPN or IPN
MASS_STORAGE_AVAILABLE or MSA
MASS_STORAGE_CAPACITY or MSC
MASS_STORAGE_SET_NAME or MSSN
MASS_STORAGE_VOLUME_ACTIVE or MSVA
MASS_STORAGE_VOLUME_CLASSES or MSVC
MASS_STORAGE_VOLUME_ONLINE or MSVO
RECORDED_VSN or RVSN
REPAIR_ACTION_REQUIRED or RAR
REPAIR_ATTEMPTED or RA
RESERVABLE_ELEMENT or RE
RESERVED_TO_JOB or RTJ
RESERVING_JOB_IDENTIFICATION or RJI
SERIAL_NUMBER or SN
SITE_INFORMATION or SI
SYSTEM_CRITICAL_ELEMENT or SCE

iou

Specifies the IOU associated with the element whose name is specified.

Remarks

- This function is designed to be used in logical expressions.
- Only elements in the active configuration can be interrogated.
- If the attribute does not pertain to the element named, the function returns an error.

Examples

In the following example, the value of the EXTERNAL_VSN element attribute is returned for element TAPE_0 and is used in an expression:

```
our_check_will_arrive_soon= ..
  ($element(tape_0,external_vsn)='PAYROL')
```

$MASS_STORAGE_CLASS_MEMBERS Function

Purpose Returns the element name and the recorded VSN for each member of a mass storage class.

Format **$MASS_STORAGE_CLASS_MEMBERS**
(class)

Parameters **class**

Specifies the name of the mass storage class whose members are to be returned. This parameter is required.

Examples The following example returns the element names and recorded VSNs for mass storage class K:

```
display_value $mass_storage_class_members(k)
DISK_9853_1001
VSN001
DISK_9853_1002
VSN002
DISK_9853_1003
VSN003
DISK_9853_1004
VSN004
```

$PHYSICAL_ADDRESS Function

Purpose Returns an integer value representing the physical address from an upline element to a downline element.

Format **$PHYSICAL_ADDRESS**
(upline_element,
downline_element,
iou)

Parameters **upline_element**

Specifies the name of the element whose physical address to the downline element is requested. This parameter is required.

downline_element

Specifies the name of the element that can be accessed by the physical address from the upline element. This parameter is required.

iou

Specifies the IOU associated with the upline element whose name is specified. This parameter is required if the system has multiple IOUs and a channel name is specified. The name of the channel is not unique unless qualified by the IOU identification. If you specify this parameter in a single IOU system, its value must be IOU0.

Remarks

- A value of 1 is returned if the two elements are not connected.
- The meaning of the physical address depends on the type of the two elements:

Upline Element Type	Downline Element Type	Meaning of Physical Address Returned
Central memory	External processor	Memory port number
Central memory	Central processor	Memory port number
Central memory	IOU	Memory port number
Channel	Channel adapter	Equipment number
Channel	Communications element	Equipment number
Channel	Peripheral	Equipment number
Channel adapter	Peripheral	Address from channel adapter to peripheral

Upline Element Type	Downline Element Type	Meaning of Physical Address Returned
IOU	PP	PP number
Mainframe	CP	CP number
Mainframe	IOU	IOU number
Peripheral	Peripheral	Address from upline element to downline element
PP	Channel	Channel number

For additional information, refer to the description of the PCU subcommand DEFINE_ELEMENT in chapter 2, Physical Configuration Utility.

Examples In the following example, the physical address from upline element CH4 to downline element BLUE_DISK_CONTROLLER is 0:

```
display_value $physical_address(ch4,blue_disk_controller)
0
```

$STORAGE _DEVICE _NAME Function

Purpose Returns a string containing the element name for the disk unit or tape unit on which a specified volume is mounted.

Format **$STORAGE _DEVICE _NAME**
(recorded _vsn)

Parameters **recorded _vsn**

Specifies the 6-character volume serial number (VSN) of the disk or tape volume mounted on the element whose name you want returned. This parameter is required.

Remarks If the recorded VSN is not defined, this function returns a null string.

Examples The following example returns element name for the disk unit on which the volume indicated by recorded VSN V01003 is mounted:

```
display_value $storage_device_name(v01003)
DISK_9853_1003
```

$SYSTEM_SET_NAME Function

Purpose Returns the name of the system set.

Format **$SYSTEM_SET_NAME**

Parameters None.

Examples The following example returns the name of the system set:

```
display_value $system_set_name
GOLD_SYSTEM
```

Part II: Maintenance Features and Utilities

System Core Commands 4

System Core Commands 4

The system core commands are a group of commands that you can enter only during deadstart. Table 4-1 provides a brief description of the core commands described in this chapter. This chapter also describes the $SECURITY_OPTION SCL function, which tests for values set by the SET_SECURITY_OPTION system core command.

You can enter any of the system core commands interactively during deadstart. To enter core commands interactively, you must initiate a deadstart with operator pause. During deadstart, the system prompts you to begin entering core commands as follows:

```
Enter system core commands
```

You respond by entering the appropriate core commands. The last core command you enter is either the AUTO or GO command. These commands terminate core command processing.

NOTE

The parameters of the system core commands are specified positionally and not by parameter name; that is, you must specify the parameter values shown in the command format descriptions.

You can include some of the system core commands (specifically, USE_CONFIGURATION_PROLOG, USE_INSTALLED_CONFIGURATION, SET_SYSTEM_ATTRIBUTES, and SET_SECURITY_OPTION) in the deadstart command file (DCFILE). In this way, you can avoid having to enter these commands from the system console. For information on how to do this, see the DCFILE description in chapter 5, Deadstart File Software.

Table 4-1. Summary of System Core Commands and Functions

Command	Description
AUTO	Terminates core command processing and inhibits further operator pauses for the remainder of the deadstart (unless an error condition occurs).
DEFINE_MS_FLAW	Defines the location of a mass storage media defect.
DISPLAY_SYSTEM_ATTRIBUTE	Displays the value of a system performance attribute. (This command is described in the NOS/VE System Performance and Maintenance manual, Volume 1.)
DISPLAY_TIME_ZONE	Displays time zone information relative to universal time coordinated.
GO	Terminates core command processing; deadstart continues until the next operator pause.
INITIALIZE_SYSTEM_DEVICE	Initiates an installation deadstart or recovery of the system set by initializing the system device.

(Continued)

Table 4-1. Summary of System Core Commands and Functions *(Continued)*

Command	Description
$SECURITY_OPTION	Indicates whether or not a specific security option is currently active.
SET_SYSTEM_ATTRIBUTE	Changes the current value of a system performance attribute. (This command is described in the NOS/VE System Performance and Maintenance manual, Volume 1.)
SET_INSTALLATION_TAPE	Specifies the name of the file that will contain the packing list of software products or corrections for a subsequent software installation. This command also specifies the external and recorded volume serial numbers and tape type of the tape that contains the packing list.
SET_SECURITY_OPTION	Activates or deactivates optional NOS/VE security features.
SET_TIME_ZONE	Sets time zone information (relative to universal time coordinated) to be saved across deadstarts.
USE_CONFIGURATION_ PROLOG	Specifies which configuration prolog (on the deadstart file) is to be used.
USE_INSTALLED_ CONFIGURATION	Specifies the physical configuration to be installed: either the current, active configuration or the configuration read from the deadstart file.

AUTO Command

Purpose Terminates the system core command processor and causes the deadstart process to continue to completion; no further operator pauses occur except for error conditions. (The AUTO and GO commands perform the same function except that the GO command causes the deadstart process to advance only to the next operator pause.)

Format **AUTO**

Parameters None.

Remarks

- You can enter the AUTO command only from the system console. Do not place an AUTO command in a DCFILE file.
- Operator pauses affected by this command include the installation options display and the PCU and LCU prompts. When you terminate core command processing with the AUTO command, these prompts do not appear unless an exception condition occurs that requires operator action.

DEFINE_MS_FLAW Command

Purpose Defines to NOS/VE the location of a mass storage media defect.

Format

DEFINE_MS_FLAW or
DEFMSF
  vsn
  cylinder
  track
  sector

Parameters

vsn

Specifies the volume serial number (VSN) of the mass storage volume on which the flaw is to be defined. The VSN is the same name defined for the volume on the INITIALIZE_MS_VOLUME subcommand. This parameter is required.

cylinder

Specifies a cylinder number identifying the location of the defect. This parameter is required.

track

Specifies a track number identifying the location of the defect. This parameter is required.

sector

Specifies a sector number identifying the location of the defect. This parameter is required.

Remarks

- The parameters for this command are specified positionally; you must enter them in the order shown in the command description.
- The term *flaw* refers to the mass storage space surrounding the volume defect. NOS/VE treats this space as a unit of allocation. The effect of setting the flaw is that NOS/VE no longer assigns the defective space to any new files. In addition, NOS/VE may relocate the data in an existing file to avoid loss of data.
- This command is primarily used to define a flaw on the NOS/VE system device prior to its initialization or activation during deadstart. Issuing this command prevents NOS/VE from using the defective space for volume-resident system files created by the initialization process.
- You can also use this command to define a flaw for any other volume either prior to, or after, the volume is initialized. Its effect is identical to that of the DEFINE_MS_FLAW LCU subcommand. However, the latter command is preferred for volumes other than the system device.
- For further information about the effect of this command, see the description of the DEFINE_MS_FLAW LCU subcommand in chapter 3, Logical Configuration Utility.

DISPLAY_TIME_ZONE Command

Purpose	Displays time zone information entered by the SET_TIME_ZONE command, as well as data entered by the CHANGE_TIME_ZONE operator command.
Format	**DISPLAY_TIME_ZONE** or **DISTZ**
Parameters	None.
Remarks	The SET_TIME_ZONE command is described later in this chapter. The CHANGE_TIME_ZONE command is described in the NOS/VE Operations manual.

GO Command

Purpose Terminates the system core command processor and causes the deadstart process to continue until the next operator pause. (The AUTO and GO commands perform the same function except that the AUTO command causes the deadstart to continue to completion.)

Format GO

Parameters None.

Remarks

- You can enter the GO command only from the system console. Do not place a GO command in a DCFILE file.
- Operator pauses affected by this command include the installation options display and the PCU and LCU prompts. When you terminate core command processing with the GO command, the system pauses at each of these points during the deadstart process and asks if you want to enter input.

INITIALIZE_SYSTEM_DEVICE Command

Purpose Initializes the volume on the deadstart disk unit[1] and assigns a VSN to the volume.

Format **INITIALIZE_SYSTEM_DEVICE** or
INISD
  vsn
  boolean
  RECOVER_SYSTEM_SET
  set_name

Parameters **vsn**

Specifies the 6-character volume serial number (VSN) written on the device label. This parameter is required.

boolean

Specifies whether device flaws currently defined for the volume are to be retained. The default is TRUE.

TRUE

Flaws are retained if the system is able to read an existing, valid NOS/VE label and the flaws can be read correctly from the device. If the flaws can be retained on a 895-2 device, the device will not be soft-sectored. If you specify TRUE (or omit the parameter) and the system determines that flaws cannot be retained, the system reinitializes the volume as if FALSE had been specified.

FALSE

The volume's existing flaws are discarded. 895-2 devices will be soft-sectored to ensure that, if the CYBER Initialization Package (CIP) has been removed, the cylinders previously reserved for the CIP are formatted in the NOS/VE sector size.

RECOVER_SYSTEM_SET

Specifies whether the system set is to be recovered. RECOVER_SYSTEM_SET is the only name you can specify for this parameter. Omit this parameter if you want the system set to be initialized. When the system set is initialized, all files on those volumes are lost or must be restored from a backup. After the system set is initially installed, specify RECOVER_SYSTEM_SET to recreate the system device and to recover the remaining members of the system set.

set_name

Specifies the name to be assigned to the system set. The default is NVESET.

1. Before a volume can be initialized, it must be formatted. Except for 895 disks, NOS/VE does not format volumes. Volumes manufactured by Control Data are formatted at the factory. To format a volume, you must contact a customer engineer.

Remarks

- The parameters for this command are specified positionally; you must enter them in the order shown in the command description.
- This command tells the system to perform an installation deadstart or a system device initialization. You must enter it from the system console. This command cannot be executed from a DCFILE.
- Initializing an 895 disk causes NOS/VE to automatically format (soft sector) the disk. This process takes approximately five minutes.

$SECURITY_OPTION SCL Function

Purpose Returns a boolean value indicating whether or not a specific security option is currently active.

Format **$SECURITY_OPTION(name, option)**
name
option

Parameters **name**

Specifies the name of the security option. This parameter is required. You can specify the following keywords:

CONSOLE_OPERATION_ONLY
SECURE_ANALYSIS
SECURITY_AUDIT

option

Specify the following keyword value:

ACTIVE

Indicates whether or not the specified security option is active.

Examples

```
/DISPLAY_VALUE $SECURITY_OPTION(SECURITY_AUDIT,ACTIVE)
/TRUE
```

SET_INSTALLATION_TAPE Command

Purpose

Specifies the volume serial numbers and tape type of the tape containing the packing list and the name of the file in which to store the packing list. The system uses this information later when you install software using the INSTALL_SOFTWARE utility. The INSTALL_SOFTWARE utility is described in chapter 7, Software Installation.

Format

SET_INSTALLATION_TAPE or
SETIT
  packing_list
  external_vsn
  recorded_vsn
  type

Parameters

packing_list

Specifies the 1- to 16-character name of the file into which the packing list is to be copied. This file resides in the installation data base catalog which, by default, is $SYSTEM.SOFTWARE_MAINTENANCE.INSTALLATION_DATABASE. This parameter is required.

external_vsn

Specifies a 1- to 6-character string representing the external volume serial number of the tape containing the packing list to be used in a subsequent software installation. If you omit this value, you must specify the recorded_vsn value. You can also specify both external_vsn and recorded_vsn parameter values.

recorded_vsn

Specifies a 1- to 6-character string representing the recorded volume serial number of the tape containing the packing list to be used in a subsequent software installation. If you omit this value, you must specify the external_vsn value. You can also specify both recorded_vsn and external_vsn parameter values.

type

Specifies the type of tape containing the packing list. The following keyword values can be specified; the default value is MT9$6250.

MT9$1600

9-track, 1600-cpi density.

MT9$6250

9-track, 6250-cpi density.

MT18$38000

18-track, 38000-cpi density.

Remarks

- The parameters for this command are specified positionally; you must enter them in the order shown in the command description.
- If you enter both external_vsn and recorded_vsn parameter values, the values should be identical. Therefore, when you load the packing list, the system can assign the tape automatically. Otherwise, you must assign the tape using the ASSIGN_DEVICE command. The ASSIGN_DEVICE command is described in the NOS/VE Operations manual.
- The information this command provides applies to a subsequent INSTALL_SOFTWARE utility session when one of the following conditions is satisfied:
  - An installation deadstart is being performed and an AUTO command ends system core command processing.
  - An installation deadstart is being performed, a GO command ends system core command processing, and Install Installation Tape is selected from the menu during the system initiation phase of the deadstart.
  - A continuation deadstart is being performed, a GO command ends system core command processing, there are no jobs to recover, and Install Installation Tape is selected from the menu during the system initiation phase of the deadstart.

Examples

To load installation tape VE0001 and save the packing list on file PACKING_LIST, enter:

```
set_installation_tape packing_list 'VE0001'
```

SET_SECURITY_OPTION Command

Purpose Activates or deactivates a security option.

Format **SET_SECURITY_OPTION** or
SETSO
  option
  value

Parameters **option**

Specifies the name of the security option to be activated or deactivated. This parameter is required. You can specify the following keywords:

CONSOLE_OPERATION_ONLY

Restricts System Operator Utility activities to the system console.

SECURE_ANALYSIS

Deactivates system analysis tools.

SECURITY_AUDIT

Enables use of the Audit utility.

ALL

Selects all three security options.

value

Specifies whether or not the security option specified on the option parameter is to be activated or deactivated. This parameter is required. You can specify the following keywords:

ON

Activates the specified option(s).

OFF

Deactivates the specified option(s).

Remarks

- Each security option can be set only once during each deadstart of NOS/VE.
- For the SECURE_ANALYSIS option, from the time that deadstart is initiated to the time this command is entered to change the SECURE_ANALYSIS option, NOS/VE continues to honor the value that was set at the previous deadstart.
- To ensure that the security options are set appropriately at every deadstart, you can place the SET_SECURITY_OPTION system core commands in the DCFILE file on the deadstart file. Because the security options can be set only once per deadstart, placing these commands on the deadstart file ensures that the options cannot be changed interactively during deadstart.
- Refer to the NOS/VE Security Administration manual for information about the optional security features.

SET_TIME_ZONE Command

Purpose Sets time zone information relative to universal time coordinated.

Format **SET_TIME_ZONE** or
SETTZ
  hours
  minutes
  daylight

Parameters **hours**

Specifies the hour. Values vary from -12 to 12. A zero denotes universal time coordinated. A minus value denotes the number of hours before universal time coordinated. A positive value denotes the number of hours after universal time coordinated. This parameter is required.

minutes

Specifies the minute. Values vary from -30 to 30. A zero denotes universal time coordinated. A minus value denotes the number of minutes before universal time coordinated. A positive value denotes the number of minutes after universal time coordinated. The default is 0.

daylight

Specifies whether the time denotes daylight saving time or standard time. TRUE denotes daylight saving time; FALSE denotes standard time. The default is FALSE.

Remarks

- The parameters for this command are specified positionally; you must enter them in the order shown in the command description.
- The time zone information is saved across deadstarts in the common disk area on the CIP device.
- To display time zone information set by this command, use the DISPLAY_TIME_ZONE command described earlier in this chapter.
- Universal time coordinated is sometimes called Greenwich mean time.

Examples To set the time zone for Minneapolis, Minnesota relative to Greenwich mean time, enter:

```
set_time_zone -6 0 TRUE
```

USE_CONFIGURATION_PROLOG Command

Purpose Specifies which configuration prolog is to be used for an installation deadstart.

Format

```
USE_CONFIGURATION_PROLOG or
USECP
   name
```

Parameters

name

Specifies the name of the configuration prolog that contains the PCU and LCU commands and subcommands needed to establish the equipment configuration. This parameter is required.

Remarks

- This system core command is used only during an installation deadstart.
- Configuration prologs reside in the object library file PROLOG_LIBRARY in the catalog $SYSTEM.SITE_OS_MAINTENANCE.DEADSTART_COMMANDS. For more information on the PROLOG_LIBRARY file, refer to chapter 5, Deadstart File Software.
- Use the MAINTAIN_DEADSTART_SOFTWARE subcommand CREATE_PROLOG to create a configuration prolog. The MAINTAIN_DEADSTART_SOFTWARE utility is described in chapter 5, Deadstart File Software.

USE_INSTALLED_CONFIGURATION Command

Purpose For a NOS/VE continuation deadstart, this command specifies which physical configuration file is to be installed.

Format **USE_INSTALLED_CONFIGURATION** or
USEIC
boolean

Parameters *boolean*

Specifies which physical configuration is to be installed. The default value is TRUE.

TRUE

Specifies that the currently installed physical configuration is to be reinstalled.

FALSE

Specifies that the physical configuration defined on the deadstart file is to be installed. The physical configuration defined on the deadstart file is maintained in file PHYSICAL_CONFIG. The PHYSICAL_CONFIG file is described in chapter 5, Deadstart File Software.

Remarks

- Regardless of which physical configuration file this core command selects, you may edit the file during deadstart (that is, before the physical configuration is installed) using the EDIT_PHYSICAL_CONFIGURATION subutility of the Physical Configuration utility. For more information, see Changing the Configuration at Deadstart in chapter 13, Repair Solutions.

- If you specify the value FALSE on this command, you must take steps to ensure that the number of elements and their states are correct. Whenever the state of an element is changed, the new state is updated in the installed configuration file on the system device. If the PHYSICAL_CONFIG file is out of date and you replace the installed configuration, you may lose information about the state of an element.

Deadstart File Software 5

Deadstart File Software 5

This chapter provides information in the following three areas related to deadstart maintenance:

- The data files and catalogs used in creating and maintaining deadstart files.
- The processes used to create or modify a deadstart file and to place the deadstart file on magnetic tape or disk.
- The MAINTAIN_DEADSTART_SOFTWARE utility subcommands used to create and maintain deadstart file software.

Deadstart-Related Files and Catalogs

This section describes the input and output catalogs and files associated with the creation of a NOS/VE deadstart file. This section also gives an overview of how the following MAINTAIN_DEADSTART_SOFTWARE utility subcommands are used to create deadstart files and catalogs:

- CREATE_VE_DEADSTART_TAPE
- ESTABLISH_DISK_BASED_SYSTEM
- REPLACE_DS_TAPE_CONFIGURATIONS
- GENERATE_VE_DEADSTART_CATALOG
- CREATE_VE_DEADSTART_CATALOG

The term *deadstart file* refers to the tape file or disk file used to deadstart the system. The data files placed in the deadstart file contain programs, procedures, and configuration data required to initialize system software and define mainframe-specific software and hardware configurations. For sites that have multiple mainframes, a single deadstart file can contain the configuration data for all mainframes.

The term *deadstart catalog* refers to a NOS/VE catalog that contains the files to be written to a deadstart file as part of the process of creating a new deadstart file. Generally, the process of creating a new deadstart file requires that you first create a new deadstart catalog, and then write the files in that catalog to tape or disk.

The MAINTAIN_DEADSTART_SOFTWARE utility subcommand GENERATE_VE_DEADSTART_CATALOG generates a deadstart catalog. For your convenience, NOS/VE also provides a subcommand that reads the files from an existing deadstart tape and places them in a catalog suitable for subsequent use as a deadstart catalog. This subcommand is CREATE_VE_DEADSTART_CATALOG.

The MAINTAIN_DEADSTART_SOFTWARE utility subcommand CREATE_VE_DEADSTART_TAPE creates a new deadstart file on tape. The ESTABLISH_DISK_BASED_SYSTEM subcommand creates a deadstart file and places it on the system disk. Both of these subcommands initiate a process that generates an entire deadstart file from a previously created deadstart catalog. The default deadstart catalog referenced by MAINTAIN_DEADSTART_SOFTWARE utility subcommands is SITE_OS_MAINTENANCE.Lxxx.DEADSTART_CATALOG,[1] shown in figure 5-1. You can, however, specify any appropriate catalog as the deadstart catalog.

The MAINTAIN_DEADSTART_SOFTWARE utility subcommand REPLACE_DS_TAPE_CONFIGURATIONS replaces the deadstart command files that define mainframe software and hardware configurations and incorporates these new files in the creation of a new deadstart tape. In comparison, the CREATE_VE_DEADSTART_TAPE subcommand simply copies a deadstart catalog to tape. The deadstart command files are described later in this chapter.

1. The xxx variable represents the operating system PSR level.

Deadstart Maintenance Catalogs

Figure 5-1 shows the NOS/VE catalog structure of deadstart-related files. Two $SYSTEM subcatalogs contain deadstart-related files: NOSVE_MAINTENANCE and SITE_OS_MAINTENANCE.

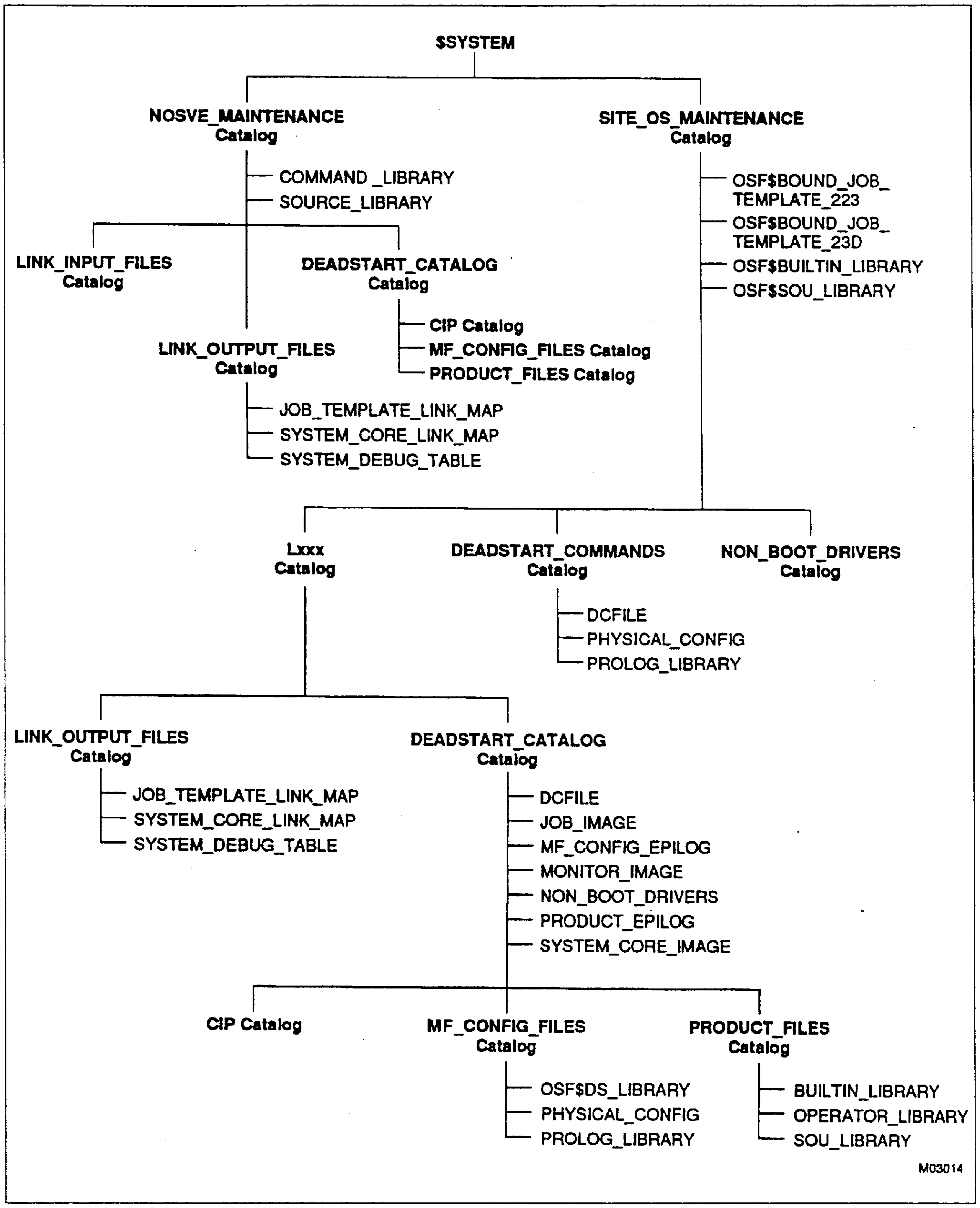


Figure 5-1. Deadstart Maintenance Catalogs

NOSVE_MAINTENANCE Catalog

The NOSVE_MAINTENANCE catalog contains all deadstart-related files released with NOS/VE. The files in the NOSVE_MAINTENANCE catalog are used as input to the process of creating a new deadstart catalog.

Although you cannot modify files in the NOSVE_MAINTENANCE catalog directly, you can replace some of these files with site-defined versions residing in the SITE_OS_MAINTENANCE catalog. You can create your own versions of the following files:

```
LINK_INPUT_FILES.OSF$BOUND_JOB_TEMPLATE_223
LINK_INPUT_FILES.OSF$BOUND_JOB_TEMPLATE_23D
DEADSTART_CATALOG.DCFILE
DEADSTART_CATALOG.NON_BOOT_DRIVERS
DEADSTART_CATALOG.PRODUCT_FILES.OSF$BUILTIN_LIBRARY
DEADSTART_CATALOG.PRODUCT_FILE.OSF$SOU_LIBRARY
```

In general, the GENERATE_VE_DEADSTART_CATALOG subcommand first searches the SITE_OS_MAINTENANCE catalog for site-defined versions of these files. In each case, the system uses the SITE_OS_MAINTENANCE version if it finds one; otherwise, the released version of the file in the NOSVE_MAINTENANCE catalog is used.

The OSF$BOUND_JOB_TEMPLATE_223 file contains, among other things, the site-modifiable system module responsible for performing SRU calculations. Similarly, the OSF$BOUND_JOB_TEMPLATE_23D file contains the site-modifiable modules for password encryption, password validation, and magnetic tape validation. Chapter 6, Site Tailoring, describes the process of modifying system modules.

For information about how to modify the DCFILE file, see DCFILE File later in this chapter.

Modification of permanent file maintenance procedures that reside in the OSF$SOU_ LIBRARY file is described in chapter 6, Site Tailoring.

SITE_OS_MAINTENANCE Catalog

In contrast to the NOSVE_MAINTENANCE catalog, which contains the released versions of deadstart-related input files, the SITE_OS_MAINTENANCE catalog is used to store all site-defined files related to deadstart.

The SITE_OS_MAINTENANCE catalog has three subcatalogs. Two of these subcatalogs, DEADSTART_COMMANDS and NON_BOOT_DRIVERS, contain files used as input for the creation of a new deadstart catalog. The DEADSTART_COMMANDS subcatalog contains the deadstart command files that define mainframe specific software and hardware configurations. These files are described in Deadstart Command Files later in this chapter. The NON_BOOT_DRIVERS subcatalog is used to store site-written peripheral processor (PP) drivers.

The SITE_OS_MAINTENANCE.Lxxx[2] catalog contains the output files created when a GENERATE_VE_DEADSTART_CATALOG subcommand is executed. These output files consist of the new deadstart catalog and the link output files. The deadstart catalog resides in the DEADSTART_CATALOG catalog and serves as input to the CREATE_VE_DEADSTART_TAPE subcommand or the ESTABLISH_DISK_BASED_SYSTEM subcommand. The link output files reside in the LINK_OUTPUT_FILES catalog and contain information about the software in the new deadstart catalog.

The system automatically names the SITE_OS_MAINTENANCE.Lxxx subcatalog according to the current version of NOS/VE, as defined in the file NOSVE_MAINTENANCE.LINK_INPUT_FILES.OS_VERSION. For example, the name of this subcatalog for the NOS/VE Version 1.5.1 L739AA batch corrective update is L739AA.

Deadstart Command Files

The term *deadstart command files* refers to the files contained in catalog $SYSTEM.SITE_OS_MAINTENANCE.DEADSTART_COMMANDS. In general, these files are used to define the software and hardware configurations for a specific mainframe or mainframes.

The following deadstart command files can be placed in this catalog:

File Name	Description
DCFILE	Defines one or more sets of system core commands to be executed at deadstart.
PHYSICAL_CONFIG	Defines the physical configuration of one or more mainframes.
PROLOG_LIBRARY	Contains configuration prologs to modify the physical, logical, or network configuration of one or more mainframes.

DCFILE File

The DCFILE file contains one or more sets of system core commands that execute during each deadstart. The DCFILE makes it possible to automatically execute system core commands at deadstart, rather than pausing during the deadstart and entering them at the system console. Refer to chapter 4, System Core Commands, for more information about system core commands.

As figure 5-2 shows, system core command sets are named consecutively: DCF01, DCF02,..., DCFnn, where nn is an octal integer. You can include as many sets of system core commands as you require. If your configuration includes more than one mainframe, you will probably want to include a set for each mainframe.

The first line of a DCFILE file is the DCF01 header. The DCF01 header is then followed by the system core commands to be included in the first set. The system recognizes the end of the first set of system core commands when the next header (DCF02) is encountered or when end-of-file is reached. The same rule applies for each subsequent set of system core commands.

2. The xxx variable represents the operating system PSR level.

The following is an example of a DCFILE file:

```
DCF01
"This is the DCFILE.
"
"Your system analyst can replace this text with system
"core commands to be processed at every deadstart.
"
"For further information on system core commands, refer
"to the NOS/VE System Performance and Maintenance manual, Volume 1
"(system attribute commands).
"
DCF02
"Unused
"
DCF03
"Unused
"
DCF04
"For C180-855 S/N 002.
 use_configuration_prolog SN002_CLOSED_SHOP
 set_system_attribute job_recovery_option 0
 set_system_attribute maximum_segment_length 300000000
"
DCF05
"For C180-830 S/N 631
set_system_attribute maximum_active_jobs 75
set_system_attribute job_recovery_option 0
use_configuration_prolog SN631_ISD1_2X4
```

Figure 5-2. Example DCFILE File

The core command set ordinal is the octal integer portion of the appropriate DCFxx header. For example, 04 is the command set ordinal for DCF04. The DCFILE is selected on the NOS/VE Boot display, option 3.

PHYSICAL_CONFIG File

The PHYSICAL_CONFIG file contains the PCU subcommand entries DEFINE_ELEMENT and DEFINE_WORKING_MAINFRAME. These subcommands describe your peripheral device configuration.

Chapter 2, Physical Configuration Utility, describes how to create the PHYSICAL_CONFIG file. Chapter 2 also contains an example of a configuration file that defines multiple mainframes. Refer also to the NOS/VE Site Analyst Examples online manual.

On a continuation deadstart you have a choice: you can install the new physical configuration from the PHYSICAL_CONFIG file on the deadstart file, or you can use the physical configuration that was installed at the last deadstart. If you install the PHYSICAL_CONFIG file from the deadstart file, the system copies the PHYSICAL_CONFIG file from the deadstart file to the $LOCAL.PHYSICAL_CONFIGURATION file. This is true for both installation and continuation deadstarts. If you use the current configuration instead, the system copies the currently installed configuration to the $LOCAL.PHYSICAL_CONFIGURATION file. The system automatically installs the contents of this file unless you request an opportunity to modify it. Use the USE_INSTALLED_CONFIGURATION system core command to select the physical configuration you want to install. There are two ways to execute the USE_INSTALLED_CONFIGURATION command: you can place the command in the DCFILE prior to deadstart or you can initiate a deadstart with operator pause and enter the USE_INSTALLED_CONFIGURATION command interactively during the pause for system core commands. The USE_INSTALLED_CONFIGURATION command is described in chapter 4, System Core Commands.

If you need to modify the physical configuration during deadstart, use the PCU subcommands EDIT_PHYSICAL_CONFIGURATION and INSTALL_PHYSICAL_CONFIGURATION to edit and install the $LOCAL.PHYSICAL_CONFIGURATION file. Refer to chapter 2, Physical Configuration Utility, for more information about the PCU subcommands. Refer to chapter 13, Repair Solutions, for more information about changing the configuration at deadstart.

When the deadstart completes, the system copies the installed physical configuration to the $SYSTEM.MAINFRAME.CONFIGURATION file. You can use this file to create a new PHYSICAL_CONFIG file.

NOTE

NOS/VE updates the installed configuration each time an element state change is made using the LCU subcommand CHANGE_ELEMENT_STATE. However, these changes are only reflected in the $SYSTEM.MAINFRAME.CONFIGURATION file after the next deadstart. Therefore, if you replace the previously installed configuration with a new PHYSICAL_CONFIG file, be sure that the states (ON, OFF, or DOWN) of all peripheral elements in the new configuration are correct.

PROLOG_LIBRARY File

The PROLOG_LIBRARY file is a site-defined object library containing one or more configuration prologs. A configuration prolog is a procedure that specifies modifications to the logical configuration, the physical configuration, or the network configuration. Refer to Configuration Prologs later in this chapter for information about how to create a configuration prolog. The size of the PROLOG_LIBRARY file is limited to 512,000 bytes to ensure that the system has enough space on the system device to recover itself. Configuration prologs are optional. Two reasons for using configuration prologs are:

- To automate the preparation of mass storage volumes. If you have a large disk configuration, it would be inconvenient to enter the required INITIALIZE_MS_VOLUME, CHANGE_MS_CLASS, and ADD_VOLUME_TO_SET subcommands manually.

- To tailor the NOS/VE configuration for a particular instance of deadstart. For example, if you use different NOS/VE configurations on weekdays and weekends, you might want to create a separate configuration prolog to automatically define each configuration.

On an installation deadstart, you can use the USE_CONFIGURATION_PROLOG system core command to select the configuration prolog to be executed. The USE_CONFIGURATION_PROLOG command is described in chapter 4, System Core Commands. Only one configuration prolog can be executed per deadstart. If you do not include a USE_CONFIGURATION_PROLOG command, the system looks for the prolog with the current mainframe ID name (for example, $SYSTEM_0990_0102). On a continuation deadstart, the system ignores the USE_CONFIGURATION_PROLOG command.

Configuration Prologs

Each configuration prolog consists of one or more of the following: a set of LCU mainframe subcommands, a set of LCU network subcommands, and a set of PCU subcommands. In the prolog, each set of subcommands serves as input to a separate COLLECT_TEXT command entry. Each COLLECT_TEXT command directs its text to a corresponding output file. The names of these files are as follows:

```
$LOCAL.LCU_MAINFRAME_SUBCOMMANDS
$LOCAL.LCU_NETWORK_SUBCOMMANDS
$LOCAL.PCU_SUBCOMMANDS
```

During deadstart, the configuration prolog executes to create these files. The system then executes each file at the appropriate time during the deadstart.

We recommend that you use the CREATE_PROLOG subcommand of the MAINTAIN_DEADSTART_SOFTWARE utility to create your configuration prologs. This ensures that your prologs follow standard NOS/VE conventions and that you can use any upgrade conversion tools that might be required for a future version of NOS/VE.

Example: Creating a Configuration Prolog

The following example creates a configuration prolog named EXAMPLE in the $SYSTEM.SITE_OS_MAINTENANCE.DEADSTART_COMMANDS.PROLOG_LIBRARY command library:

```
/maintain_deadstart_software
maids/create_prolog prolog_name=example file=pcu_subcommands
ct?edit_physical_configuration
ct?change_element_definition blue_34 state=off
ct?quit
**

maids/create_prolog prolog_name=example file=lcu_mainframe_subcommands
ct?initialize_ms_volume element=blue_33 recorded_vsn=vsn002
ct?add_volume_to_set member_vsn=vsn002
ct?**

maids/create_prolog prolog_name=example file=lcu_network_subcommands
ct?define_channel_network network=120 connected_system=blue_di_10006b..
ct? relay_restricted=false
ct?define_host_network network=54
ct?**

 maids/quit
```

In expanded form, this produces the following configuration prolog:

```
PROCEDURE EXAMPLE (
  status)

COLLECT_TEXT $local.pcu_subcommands until='end_prolog_file'
EDIT_PHYSICAL_CONFIGURATION
  change_element_definition blue_34 state=off
QUIT
end_prolog_file
COLLECT_TEXT $local.lcu_mainframe_subcommands until='end_prolog_file'
initialize_ms_volume element=blue_33 recorded_vsn=vsn002
add_volume_to_set member_vsn=vsn002
end_prolog_file
COLLECT_TEXT $local.lcu_network_subcommands until='end_prolog_file'
define_channel_network network=120 connected_system=blue_di_10006b..
  relay_restricted=false
define_host_network network=54
end_prolog_file

PROCEND EXAMPLE
```

Example: Designing a Configuration Prolog

This example illustrates several things:

- How to create, update, and replace a configuration prolog in the PROLOG_LIBRARY.
- How to structure the lcu_mainframe_subcommands part of a configuration prolog.

As explained previouly, the configuration prolog selected by the USE_CONFIGURATION_PROLOG system core command is automatically executed during an installation deadstart. However, a configuration prolog is not executed automatically during a continuation deadstart because it is not normally desirable to initialize mass storage volumes at that time.

The PROLOG_LIBRARY is copied from the deadstart file and added to the command list during a continuation deadstart. If you deadstart with an operator pause to perform a reconfiguration, you may execute the procedures that you provided in the PROLOG_LIBRARY. For example, the structured design of the prolog below allows you to initialize the volume VSN023 by executing the following command:

```
initialize_vsn023
```

It is recommended that you not create files within your procedures. This is to ensure the ability of the system to complete deadstart. Remember that you cannot reference permanent files until after the completion of deadstart. Refer to Chapter 13, Repair Solutions, for more information about reconfiguration.

Follow these steps to create, modify, and replace a configuration prolog:

1. Create the prolog.

```
/maintain_deadstart_software
maids/create_prolog prolog_name=example2  file=lcu_mainframe_subcommands
ct?display_value 'initializing red_836_2 as vsn023'
ct?initialize_ms_volume element=red_836_2 recorded_vsn=vsn023
ct?chamsc vsn023 dc=(y z)
ct?display_value 'adding vsn023 to set'
ct?add_volume_to_set member_vsn=vsn023

ct?display_value 'initializing red_836_3 as vsn060'
ct?initialize_ms_volume element=red_836_3 recorded_vsn=vsn060
ct?chamsc vsn060 dc=(x y)
ct?display_value 'adding vsn060 to set'
ct?add_volume_to_set member_vsn=vsn060

ct?display_value 'initializing red_836_4 as vsn071'
ct?initialize_ms_volume element=red_836_4 recorded_vsn=vsn071
ct?chamsc vsn071 dc=(x z)
ct?display_value 'adding vsn071 to set'
ct?add_volume_to_set member_vsn=vsn071
ct?**
maids/quit
```

2. Copy the prolog out of the prolog libraries.

```
/set_working_catalog $system.site_os_maintenance.deadstart_commands
/create_object_library
COL/add_module module=example2 library=prolog_library
COL/generate_library library=$local.prolog format=scl_proc
COL/quit
```

The following procedure file is the configuration prolog source, which now resides in the $LOCAL.PROLOG file:

```
PROCEDURE EXAMPLE2 (
  status)

COLLECT_TEXT $local.pcu_subcommands until='end_prolog_file'
end_prolog_file
COLLECT_TEXT $local.lcu_mainframe_subcommands until='end_prolog_file'
  display_value 'initializing red_836_2 as vsn023'
  initialize_ms_volume element=red_836_2 recorded_vsn=vsn023
  chamsc vsn023 dc=(y z)
  display_value 'adding vsn023 to set'
  add_volume_to_set member_vsn=vsn023

  display_value 'initializing red_836_3 as vsn060'
  initialize_ms_volume element=red_836_3 recorded_vsn=vsn060
  chamsc vsn060 dc=(x y)
  display_value 'adding vsn060 to set'
  add_volume_to_set member_vsn=vsn060

  display_value 'initializing red_836_4 as vsn071'
  initialize_ms_volume element=red_836_4 recorded_vsn=vsn071
  chamsc vsn071 dc=(x z)
  display_value 'adding vsn071 to set'
  add_volume_to_set member_vsn=vsn071
end_prolog_file
COLLECT_TEXT $local.lcu_network_subcommands until='end_prolog_file'
end_prolog_file

PROCEND EXAMPLE2
```

3. Restructure the prolog:

```
PROCEDURE EXAMPLE2 (
  status)

COLLECT_TEXT $local.pcu_subcommands until='end_prolog_file'
end_prolog_file
COLLECT_TEXT $local.lcu_mainframe_subcommands until='end_prolog_file'
  initialize_vsn023
  initialize_vsn060
  initialize_vsn071
end_prolog_file
COLLECT_TEXT $local.lcu_network_subcommands until='end_prolog_file'
end_prolog_file

PROCEND EXAMPLE2
```

```
    PROCEDURE initialize_vsn023

      display_value 'initializing red_836_2 as vsn023'
      initialize_ms_volume element=red_836_2 recorded_vsn=vsn023
      chamsc vsn023 dc=(y z)
      display_value 'adding vsn023 to set'
      add_volume_to_set member_vsn=vsn023

    PROCEND initialize_vsn023

    PROCEDURE initialize_vsn060

      display_value 'initializing red_836_3 as vsn060'
      initialize_ms_volume element=red_836_3 recorded_vsn=vsn060
      chamsc vsn060 dc=(x y)
      display_value 'adding vsn060 to set'
      add_volume_to_set member_vsn=vsn060

    PROCEND initialize_vsn060

    PROCEDURE initialize_vsn071

      display_value 'initializing red_836_4 as vsn071'
      initialize_ms_volume element=red_836_4 recorded_vsn=vsn071
      chamsc vsn071 dc=(x z)
      display_value 'adding vsn071 to set'
      add_volume_to_set member_vsn=vsn071

    PROCEND initialize_vsn071
```

4. Enter the following commands to add the INITIALIZE_VSNxxx procedures and the modified configuration prolog to the PROLOG_LIBRARY file:

```
/create_object_library
COL/add_module library=prolog_library
COL/replace_module l=$local.prolog
COL/generate_library library=prolog_library.$next
COL/quit
```

Creating or Modifying a Deadstart File

This section contains the following sets of instructions:

- How to create a deadstart file.
- How to replace deadstart command files on a deadstart file with deadstart command files from the SITE_OS_MAINTENANCE.DEADSTART_COMMANDS catalog.
- How to install a deadstart file or catalog.

Creating a Deadstart Tape

1. Make any necessary changes to the deadstart command files in the $SYSTEM.SITE_OS_MAINTENANCE.DEADSTART_COMMANDS catalog.

2. Perform any site tailoring that affects the deadstart file software. Chapter 6, Site Tailoring, describes the site tailoring options available to you.

3. At the system console, initiate a MAINTAIN_DEADSTART_SOFTWARE utility session.

4. Enter the GENERATE_VE_DEADSTART_CATALOG subcommand to create a new deadstart catalog.

5. Enter the CREATE_VE_DEADSTART_TAPE subcommand to copy to tape the deadstart catalog created in step 4.

6. Enter the QUIT subcommand to end the MAINTAIN_DEADSTART_SOFTWARE utility session.

7. Enter the TERMINATE_SYSTEM command to terminate the system.

8. Perform a continuation deadstart using the new deadstart tape created in step 5. This ensures that the deadstart file executes properly and that the deadstart tape can serve as a reliable backup after you copy the deadstart file to the system disk.

9. At the system console, initiate a MAINTAIN_DEADSTART_SOFTWARE utility session.

10. Enter the COLLECT_DUMP_MATERIALS subcommand to dump the link output files to tape. This is necessary only if you made site tailoring changes to the deadstart file in step 2.

11. Enter the ESTABLISH_DISK_BASED_SYSTEM subcommand to copy the new deadstart tape file to the system disk.

12. Enter the COMMIT_NEW_SYSTEM subcommand to select the new system disk deadstart file for use during the next deadstart.

13. Enter the QUIT subcommand to end the MAINTAIN_DEADSTART_SOFTWARE utility session.

14. When it is convenient to do so, enter the TERMINATE_SYSTEM command to terminate the old system and perform a continuation deadstart using the new deadstart file on the system disk. This completes the process begun by the COMMIT_NEW_SYSTEM subcommand that enables you to deadstart the system from disk using the newly installed deadstart file. The system is not committed until a TERMINATE_SYSTEM command is executed successfully.

Replacing the Deadstart Command Files

1. Make any necessary changes to the deadstart command files in the $SYSTEM.SITE_OS_MAINTENANCE.DEADSTART_COMMANDS catalog.
2. At the system console or from an interactive terminal with $SYSTEM.OSF$BUILTIN_LIBRARY added to the command list, initiate a MAINTAIN_DEADSTART_SOFTWARE utility session.
3. Enter the REPLACE_DS_TAPE_CONFIGURATIONS subcommand to replace the deadstart command files on the old deadstart tape with the new deadstart command files you created in step 1. This produces a new deadstart tape.
4. Enter the QUIT subcommand to end the MAINTAIN_DEADSTART_SOFTWARE utility session.

Installing a Deadstart File to Disk

1. At the system console, initiate a MAINTAIN_DEADSTART_SOFTWARE utility session.
2. If you have made any site tailoring changes to the deadstart file, enter the COLLECT_DUMP_MATERIALS subcommand to copy the link output files to tape.
3. Enter the ESTABLISH_DISK_BASED_SYSTEM subcommand to copy the new deadstart file or catalog to the system disk.
4. Enter the COMMIT_NEW_SYSTEM subcommand to select the new system disk deadstart file for use during the next deadstart from disk.
5. Enter the QUIT subcommand to end the MAINTAIN_DEADSTART_SOFTWARE utility session.
6. When it is convenient to do so, enter the TERMINATE_SYSTEM command to terminate the system and perform a continuation deadstart using the new deadstart file on the system disk. The COMMIT_NEW_SYSTEM subcommand does not take effect until the system terminates successfully.

MAINTAIN_DEADSTART_SOFTWARE Command and Subcommands

This section describes the MAINTAIN_DEADSTART_SOFTWARE utility and its subcommands. The MAINTAIN_DEADSTART_SOFTWARE command is described first, followed by the subcommands in alphabetical order.

Table 5-1 summarizes the MAINTAIN_DEADSTART_SOFTWARE subcommands described in this section.

Table 5-1. Summary of MAINTAIN_DEADSTART_SOFTWARE Subcommands

Subcommand	Description
COLLECT_DUMP_MATERIALS	Copies link output files to tape.
COMMIT_NEW_SYSTEM	Selects the deadstart file on the system disk for use during the next deadstart.
CREATE_PROLOG	Creates a configuration prolog.
CREATE_VE_DEADSTART_CATALOG	Copies all files on a deadstart tape to a deadstart catalog.
CREATE_VE_DEADSTART_TAPE	Copies a deadstart catalog to tape.
ESTABLISH_DISK_BASED_SYSTEM	Creates a deadstart file on the system disk.
GENERATE_VE_DEADSTART_ CATALOG	Generates a new deadstart catalog from component files.
QUIT	Terminates a MAINTAIN_DEADSTART_ SOFTWARE session.
REPLACE_DS_TAPE_ CONFIGURATIONS	Replaces the deadstart command files on an existing deadstart tape.

MAINTAIN_DEADSTART_SOFTWARE Command

Purpose Initiates a MAINTAIN_DEADSTART_SOFTWARE utility session.

Format **MAINTAIN_DEADSTART_SOFTWARE** or
MAIDS
STATUS=status variable

Parameters *STATUS*

Returns the completion status for this utility.

Remarks

- You can enter most of the MAINTAIN_DEADSTART_SOFTWARE utility subcommands at an interactive terminal, assuming you have added the $SYSTEM.OSF$BUILTIN_LIBRARY library to your command list and you have EXECUTE permission for the file. Other subcommands, as noted in the Remarks sections, must be entered at the system console.

- After you enter the MAINTAIN_DEADSTART_SOFTWARE command, the following prompt appears:

```
maids/
```

COLLECT_DUMP_MATERIALS Subcommand

Purpose Submits a batch job to copy link output files to tape. The link output files provide information about the contents of the deadstart file. This information is used to analyze a central memory dump.

Format COLLECT_DUMP_MATERIALS or
COLDM
EXTERNAL_VSN=string
RECORDED_VSN=string
TYPE=keyword
LINK_OUTPUT_CATALOG=file
STATUS=status variable

Parameters *EXTERNAL_VSN* or *EVSN* or *EV*

Specifies the external volume serial number of the tape to which link output files are to be copied. You must enter a value for this parameter or the RECORDED_VSN parameter.

RECORDED_VSN or *RVSN* or *RV*

Specifies the recorded volume serial number of the tape to which the link output files are to be copied. You must enter a value for this parameter or the EXTERNAL_VSN parameter.

TYPE or *T*

Specifies the type of tape unit required. The following keyword values can be specified; the default is MT9$6250:

MT9$1600

9-track, 1600-cpi density.

MT9$6250

9-track, 6250-cpi density.

MT18$38000

18-track, 38000-cpi density.

LINK_OUTPUT_CATALOG or *LOC*

Specifies the name of the catalog in which the link output files reside. The default is $SYSTEM.SITE_OS_MAINTENANCE.Lxxx.LINK_OUTPUT_FILES, where Lxxx is the operating system PSR level as specified in file $SYSTEM.NOSVE_MAINTENANCE.LINK_INPUT_FILES.OS_VERSION.

STATUS

Returns the completion status for this subcommand.

Remarks

- Copy the link output files to tape after making changes to the deadstart software and creating a new deadstart tape. You should also dump link output files to tape after installing a correction package or BCU.
- You need not copy the link output files after making any of the following types of changes:
  - Changes to the deadstart command files, PHYSICAL_CONFIG, PROLOG_LIBRARY, or DCFILE.
  - Changes to the OSF$BUILTIN_LIBRARY file.
  - Changes to the OSF$SOU_LIBRARY file.
  - Additions to the NON_BOOT_DRIVERS catalog.
- If problems with the system occur, send a copy of the link output file dump tape along with any relevant NOS/VE environment dump tapes to Control Data for analysis.
- The batch job that creates the dump tape executes in the job class SYSTEM with the user job name COLDM.
- There are three link output files: JOB_TEMPLATE_LINK_MAP, SYSTEM_CORE_LINK_MAP, and SYSTEM_DEBUG_TABLE. Figure 5-1 shows the catalog residence of the link output files.

COMMIT_NEW_SYSTEM Subcommand

Purpose

Selects the new deadstart file on the system disk for use during the next deadstart. Successful execution of a subsequent TERMINATE_SYSTEM command performs the actual selection.

Format

COMMIT_NEW_SYSTEM or
COMNS
SET_FLAG=boolean
STATUS=status variable

Parameters

SET_FLAG or *SF*

Specifies a boolean value that indicates whether the new deadstart file should replace the current deadstart file. The default is TRUE.

TRUE

The new deadstart file should replace the current deadstart file.

FALSE

The new deadstart file should not replace the current deadstart file.

STATUS

Returns the completion status for this subcommand.

Remarks

- You must enter this command from the system console.
- Before you commit the new deadstart file, it must reside on the system disk. Enter the ESTABLISH_DISK_BASED_SYSTEM subcommand to place a deadstart file on the system disk.
- If NOS/VE terminates abnormally before the TERMINATE_SYSTEM command completes execution and recovery is required, you must perform the deadstart, terminate the system, and then perform a continuation deadstart. This must be done to complete the selection of the new deadstart file for use in future disk deadstarts.

CREATE_PROLOG Subcommand

Purpose Initiates a text entry session for the creation of a new configuration prolog or the modification of an existing configuration prolog. This subcommand stores new and modified configuration prologs in either a source or an object library file.

If the PROLOG_NAME parameter specifies the name of an existing configuration prolog, this command adds the file specified on the FILE parameter to the existing prolog. If the configuration prolog named in the PROLOG_NAME parameter does not exist, this command creates it.

Format **CREATE_PROLOG** or
CREP
PROLOG_NAME=name
FILE=keyword
UNTIL=string
INPUT=file
OUTPUT=file
OUTPUT_FORMAT=keyword
DECK_PREFIX=name
MODIFICATION_NAME=name
STATUS=status variable

Parameters *PROLOG_NAME* or *PN*

Specifies the name of the configuration prolog to be created or changed. The default is the name of the mainframe on which the prolog executes.

FILE or F

Specifies the type of configuration instructions that are to be entered in the configuration prolog during this text entry session. This parameter is required. You can specify one of the following keywords:

LCU_MAINFRAME_SUBCOMMANDS or LMS

Specifies that text entries consist of LCU subcommands that prepare mass storage volumes or change the states of channels and peripheral devices. This keyword specifies that these LCU subcommands are to be written to the $LOCAL.LCU_MAINFRAME_SUBCOMMANDS file.

LCU_NETWORK_SUBCOMMANDS or LNS

Specifies that text entries consist of LCU subcommands that configure the NOS/VE network connections. This keyword specifies that these LCU subcommands are to be written to the $LOCAL.LCU_NETWORK_SUBCOMMANDS file.

PCU_SUBCOMMANDS or PS

Specifies that text entries consist of PCU subcommands that modify the definitions of peripheral devices. This keyword specifies that these LCU subcommands are to be written to the $LOCAL.PCU_SUBCOMMANDS file.

UNTIL or *U*

Specifies the string that marks the end of the text entry session. The default string is two asterisks (**).

INPUT or *I*

Specifies the name of an input file from which configuration instructions are to be read into the configuration prolog. The default is $COMMAND, which specifies an interactive terminal session.

OUTPUT or *O*

Specifies the name of the library file in which the configuration prolog is to be stored. The default is $SYSTEM.SITE_OS_MAINTENANCE.DEADSTART_COMMANDS.PROLOG_LIBRARY.

OUTPUT_FORMAT or *OF*

Specifies the data format of the configuration prolog. This value must match the format of the library file specified on the OUTPUT parameter. You can specify one of the following keywords; the default is COMMAND_LIBRARY:

COMMAND_LIBRARY or CL

Formats the configuration prolog as an object module. The library file specified on the OUTPUT parameter must be an object libary.

SOURCE_LIBRARY or SL

Formats the configuration prolog as a source deck. The library specified on the OUTPUT parameter must be a source library.

DECK_PREFIX or *DP*

Specifies the first half of the configuraton prolog deck name when you place the prolog in a source library. The value for DECK_PREFIX is concatenated with the value specified on the PROLOG_NAME parameter to form the deck name. The default is a null string; thus, the default deck name is the value specified on the PROLOG_NAME parameter.

MODIFICATION_NAME or *MN*

Specifies the 1- to 9-chararacter name of the modification to be assigned to the text you add to the configuration prolog deck. This parameter applies only when you specify OUTPUT_FORMAT=SOURCE_LIBRARY. The default is the user name of the job executing this subcommand.

Remarks

- Each configuration prolog can contain command entries to create three files: $LOCAL.LCU_MAINFRAME_SUBCOMMANDS, $LOCAL.LCU_NETWORK_SUBCOMMANDS, and $LOCAL.PCU_SUBCOMMANDS. However, you can specify the command entries to create only one of these files during a CREATE_PROLOG subcommand session. Therefore, you must enter the CREATE_PROLOG subcommand three times to define all three files in a configuration prolog.

- If you attempt to redefine a file on an existing configuration prolog, the CREATE_PROLOG subcommand execution aborts and issues a warning message. The original file definition is retained.

- If you specify a value for the PROLOG_NAME parameter that matches the name of a procedure in the prolog library file that is not a configuration prolog, the new configuration prolog replaces the procedure of the same name.

- For more information about configuration prologs and how to create them, see Configuration Prologs earlier in this chapter.

CREATE_VE_DEADSTART_CATALOG Subcommand

Purpose Copies all files on a deadstart tape to a deadstart catalog.

Format **CREATE_VE_DEADSTART_CATALOG** or
CREVDC
  DEADSTART_CATALOG=file
  EXTERNAL_VSN=string
  RECORDED_VSN=string
  TYPE=keyword
  STATUS=status variable

Parameters **DEADSTART_CATALOG** or **DC**

Specifies the name of the catalog to which the contents of the deadstart tape are to be copied. This parameter is required.

EXTERNAL_VSN or *EVSN* or *EV*

Specifies the external volume serial number of the deadstart tape. You must enter a value for this parameter or the RECORDED_VSN parameter.

RECORDED_VSN or *RVSN* or *RV*

Specifies the recorded volume serial number of the deadstart tape. You must enter a value for this parameter or the EXTERNAL_VSN parameter.

TYPE or *T*

Specifies the type of tape unit required. The following keyword values can be specified; the default is MT9$6250:

MT9$1600

9-track, 1600-cpi density.

MT9$6250

9-track, 6250-cpi density.

MT18$38000

18-track, 38000-cpi density.

STATUS

Returns the completion status for this subcommand.

CREATE_VE_DEADSTART_TAPE Subcommand

Purpose Copies a deadstart catalog to tape, creating a NOS/VE deadstart tape.

Format **CREATE_VE_DEADSTART_TAPE** or
CREVDT
  DEADSTART_CATALOG=file
  EXTERNAL_VSN=string
  RECORDED_VSN=string
  TYPE=keyword
  STATUS=status variable

Parameters **DEADSTART_CATALOG** or **DC**

Specifies the name of the deadstart catalog to be copied to tape. This parameter is required.

EXTERNAL_VSN or *EVSN* or *EV*

Specifies the external volume serial number of the tape on which the deadstart catalog is to be written. You must enter a value for this parameter or the RECORDED_VSN parameter.

RECORDED_VSN or *RVSN* or *RV*

Specifies the recorded volume serial number of the tape on which the deadstart catalog is to be written. You must enter a value for this parameter or the EXTERNAL_VSN parameter.

TYPE or *T*

Specifies the type of tape unit required. The following keyword values can be specified; the default is MT9$6250:

MT9$1600

9-track, 1600-cpi density.

MT9$6250

9-track, 6250-cpi density.

MT18$38000

18-track, 38000-cpi density.

STATUS

Returns the completion status for this subcommand.

Remarks

- This subcommand copies only deadstart-related files from the deadstart catalog. Other files that may have been placed in the deadstart catalog are not copied to tape. Refer to Deadstart-Related Files and Catalogs earlier in this chapter.
- Because NOS/VE requires that a deadstart file reside on a single tape volume, use at least a 1200-foot tape at 1600 cpi or a 600-foot tape at 6250 cpi.

ESTABLISH_DISK_BASED_SYSTEM Subcommand

Purpose Copies the contents of a deadstart catalog or a deadstart tape to a deadstart file on the system disk.

Format **ESTABLISH_DISK_BASED_SYSTEM** or
ESTDBS
  DEADSTART_CATALOG=file
  EXTERNAL_VSN=string
  RECORDED_VSN=string
  TYPE=keyword
  STATUS=status variable

Parameters **DEADSTART_CATALOG** or **DC**

Specifies the deadstart catalog to use in creating the deadstart file. This parameter is required when creating the system disk deadstart file from a deadstart catalog.

EXTERNAL_VSN or *EVSN* or *EV*

Specifies the external volume serial number of the deadstart tape to use in creating a deadstart file on the system disk. You must enter a value for this parameter or the RECORDED_VSN parameter when creating the disk file from a deadstart tape.

RECORDED_VSN or *RVSN* or *RV*

Specifies the recorded volume serial number of the deadstart tape to use in creating a deadstart file on the system disk. You must enter a value for this parameter or the EXTERNAL_VSN parameter when creating the system disk from a deadstart tape.

TYPE or *T*

Specifies the type of tape unit required. The following keyword values can be specified; the default is MT9$6250:

MT9$1600

9-track, 1600-cpi density.

MT9$6250

9-track, 6250-cpi density.

MT18$38000

18-track, 38000-cpi density.

STATUS

Returns the completion status for this subcommand.

Remarks

- You must enter this subcommand from the system console.
- When creating a deadstart file on the system disk from a deadstart tape, the system issues a tape mount request. After you mount the labelled deadstart tape, the following messages appear:

  ```
  Establishing disk based system ...
  Please wait for completion message.
  ```

  After a few minutes, when the transfer of data from the deadstart tape to the system disk is complete, the following message appears:

  ```
  Disk based system complete.
  ```

- Two deadstart files can reside on the system disk at the same time. One file is currently used to deadstart the system; the other file, if it exists, is inactive and is simply stored on the system disk. Use the COMMIT_NEW_SYSTEM subcommand to select the inactive deadstart file as the one to use to deadstart the system.

GENERATE_VE_DEADSTART_CATALOG Subcommand

Purpose

Merges files from the $SYSTEM.NOSVE_MAINTENANCE catalog with those from the $SYSTEM.SITE_OS_MAINTENANCE catalog to create a new deadstart catalog.

Format

GENERATE_VE_DEADSTART_CATALOG or
GENVDC
CONFIGURATION_FILES_CATALOG=file
STATUS=status variable

Parameters

CONFIGURATION_FILES_CATALOG or *CFC*

Specifies the name of the catalog containing deadstart command files you want to include in the deadstart catalog. The default is $SYSTEM.SITE_OS_MAINTENANCE.DEADSTART_COMMANDS.

STATUS

Returns the completion status for this subcommand.

Remarks

- You must enter this command at the system console.
- This command creates a new deadstart catalog in the catalog $SYSTEM.SITE_OS_MAINTENANCE.Lxxx.DEADSTART_CATALOG. The value for Lxxx is the NOS/VE version level as defined in file $SYSTEM.NOSVE_MAINTENANCE.LINK_INPUT_FILES.OS_VERSION.
- The process of merging deadstart catalogs from the NOSVE_MAINTENANCE catalog and the SITE_OS_MAINTENANCE catalog produces the link output files. The link output files contain information about the deadstart software in the new deadstart catalog. The link output files reside in catalog $SYSTEM.SITE_OS_MAINTENANCE.Lxxx.LINK_OUTPUT_FILES. The value for Lxxx is the NOS/VE version level as defined in the file $SYSTEM.NOSVE_MAINTENANCE.LINK_INPUT_FILES.OS_VERSION. Use the COLLECT_DUMP_MATERIALS subcommand to dump the link output files to tape.
- The deadstart command files PROLOG_LIBRARY and PHYSICAL_CONFIG were formerly named PROLOG_FILE and PHYSICAL_CONFIGURATION. If the PROLOG_FILE file exists in the catalog given by the CONFIGURATION_FILES_CATALOG parameter, the system converts the file's contents and copies it to the new deadstart catalog as PROLOG_LIBRARY. If the PHYSICAL_CONFIGURATION file is found, the system copies the file into the new deadstart catalog with the new file name PHYSICAL_CONFIG.
- This command produces a deadstart catalog that is suitable as input to the CREATE_VE_DEADSTART_TAPE subcommand and the ESTABLISH_DISK_BASED_SYSTEM subcommand.

- This command merges the following deadstart software files from the $SYSTEM.SITE_OS_MAINTENANCE catalog with the same components from the $SYSTEM.NOSVE_MAINTENANCE catalog:
  - Any drivers in the NON_BOOT_DRIVERS catalog.
  - OSF$SOU_LIBRARY file. This is a command library containing the permanent file maintenance commands described in chapter 6, Site Tailoring.
  - OSF$BOUND_JOB_TEMPLATE_223 and OSF$BOUND_JOB_TEMPLATE_23D library files. These libraries are site-defined and contain the object code for SRU calculation, password encryption, password validation, and magnetic tape validation. Creating and modifying these libraries is described in chapter 6, Site Tailoring.
- If the PHYSICAL_CONFIG or PROLOG_LIBRARY files are missing from the catalog specified on the CONFIGURATION_FILES_CATALOG parameter, the deadstart catalog is created without them (that is, it is unconfigured). If the DCFILE file is missing, the released version found in the NOSVE_MAINTENANCE catalog is used.

QUIT Subcommand

Purpose Ends a MAINTAIN_DEADSTART_SOFTWARE utility session.

Format **QUIT** or
QUI
STATUS=status variable

Parameters *STATUS*

Returns the completion status for this subcommand.

REPLACE_DS_TAPE_CONFIGURATIONS Subcommand

Purpose Merges the contents of an existing deadstart tape and a set of deadstart command files to produce a new deadstart tape.

Format **REPLACE_DS_TAPE_CONFIGURATIONS** or
REPDTC
INPUT_EXTERNAL_VSN=string
INPUT_RECORDED_VSN=string
OUTPUT_EXTERNAL_VSN=string
OUTPUT_RECORDED_VSN=string
CONFIGURATION_FILES_CATALOG=file
TYPE=keyword
STATUS=status variable

Parameters *INPUT_EXTERNAL_VSN* or *IEVSN* or *IEV*

Specifies the external volume serial number of the existing deadstart tape. You must enter a value for this parameter or the INPUT_RECORDED_VSN parameter.

INPUT_RECORDED_VSN or *IRVSN* or *IRV*

Specifies the recorded volume serial number of the existing deadstart tape. You must enter a value for this parameter or the INPUT_EXTERNAL_VSN parameter.

OUTPUT_EXTERNAL_VSN or *OEVSN* or *OEV*

Specifies the external volume serial number of the tape on which to copy the new deadstart file. You must enter a value for this parameter or the OUTPUT_RECORDED_VSN parameter.

OUTPUT_RECORDED_VSN or *ORVSN* or *ORV*

Specifies the recorded volume serial number of the tape on which to copy the new deadstart file. You must enter a value for this parameter or the OUTPUT_EXTERNAL_VSN parameter.

CONFIGURATION_FILES_CATALOG or *CFC*

Specifies the name of the catalog containing the deadstart command files you want to include on the new deadstart tape. The default is $SYSTEM.SITE_OS_MAINTENANCE.DEADSTART_COMMANDS.

TYPE or *T*

Specifies the type of tape unit required. You can specify one of the following keywords: the default is MT9$6250:

MT9$1600

9-track, 1600-cpi density.

MT9$6250

9-track, 6250-cpi density.

MT18$38000

18-track, 38000-cpi density.

STATUS

Returns the completion status for this subcommand.

Remarks

- The deadstart command files PROLOG_LIBRARY and PHYSICAL_CONFIG were formerly named PROLOG_FILE and PHYSICAL_CONFIGURATION. If the PROLOG_FILE file exists in the catalog given by the CONFIGURATION_FILES_CATALOG parameter, the system converts the file's contents and copies it into the new deadstart catalog as PROLOG_LIBRARY. If the PHYSICAL_CONFIGURATION file is found, the system copies the file into the new deadstart catalog with the new file name PHYSICAL_CONFIG.
- If any of the deadstart command files do not exist in the catalog specified on the CONFIGURATION_FILES_CATALOG parameter, the corresponding files on the input deadstart tape are not replaced.
- You can enter this command from an interactive terminal under the following conditions:
  - The $SYSTEM.OSF$BUILTIN_LIBRARY library is in your command list.
  - You have the appropriate access permission to the catalog given by the CONFIGURATION_FILES_CATALOG parameter. If this catalog is $SYSTEM.SITE_OS_MAINTENANCE.DEADSTART_COMMANDS, you can use the CHANGE_CONFIG_FILE_ACCESS subcommand of the CREATE_INSTALLATION_ENVIRONMENT utility to acquire access permission. The CREATE_INSTALLATION_ENVIRONMENT utility is described in chapter 7, Software Installation.
- Because NOS/VE requires that a deadstart file reside on a single tape volume, use at least a 1200-foot tape at 1600 cpi or a 600-foot tape at 6250 cpi.
- This subcommand issues two tape mount requests. The first request is for the existing deadstart tape. After the job reads the deadstart tape and creates a new deadstart file, the job issues the second tape mount request. The second request is for the tape on which to write the new deadstart file. All deadstart tapes must be labelled tapes.

Site Tailoring 6

Site Tailoring 6

This chapter describes several ways in which you can tailor NOS/VE operations to the needs of your particular site.

Prologs and Epilogs

This section describes three types of prologs and epilogs:

- System deadstart prologs and epilogs, which you can use to execute commands or procedures at various points during deadstart.
- System prologs and epilogs, which are automatically executed whenever a user logs in to or logs out of the system.
- The system termination prolog, which executes after a TERMINATE_SYSTEM command has been entered.

All of the prologs and epilogs described in this chapter reside in the catalog $SYSTEM.PROLOGS_AND_EPILOGS. There are other types of prologs and epilogs that are executed by NOS/VE, specifically:

- Job class prologs and epilogs, which are described in the NOS/VE System Performance and Maintenance manual, Volume 1.
- Account, project, and user prologs and epilogs, which are described in the NOS/VE User Validation manual.

System Deadstart Prologs and Epilogs

The deadstart process, or system activation, consists of three nested phases: the system initiation phase, the production activation phase, and the network activation phase. Figure 6-1 outlines what happens during these phases and how they are nested. Each phase has an associated prolog file and epilog file. Because these prolog and epilog files execute at different times during the deadstart, each is suited for different types of system tailoring. You can modify these prolog and epilog files to tailor the system during the deadstart process.

```
*************************** System Activation Begins *************************

1. Execute CIP.
2. Enter system core commands.
3. Enter PCU subcommands.
4. Enter LCU subcommands.
5. Job and file recovery.

     *********************** System Initiation Begins ***********************

     a. The Accounting Analysis System records the time of the deadstart in
          the system log.
     b. The SYSTEM_INITIATION_PROLOG file executes.
     c. Install the installation tape for an upgrade, or install deferred files
          for a batch corrective update (BCU), as needed.

          ******************* Production Activation Begins ******************

          1. JOB_ACTIVATION_PROLOG file executes.
          2. Validation of tape access is enabled.
          3. Recovery of the job scheduling profile.
          4. HPA monitor job is initiated.
          5. Dual-state; remote host I/O tasks, and interactive tasks
               are activated.

               ****************** Network Activation Begins *****************

               a. NETWORK_ACTIVATION_PROLOG file executes.
               b. NAM/VE is activated.
               c. NETWORK_ACTIVATION_EPILOG file executes.

               ****************** Network Activation Ends *******************

          6. JOB_ACTIVATION_EPILOG file executes.

          ******************* Production Activation Ends ********************

     d. SYSTEM_INITIATION_EPILOG executes.

     ************************ System Initiation Ends ************************

************************* System Activation Complete ***********************
```

Figure 6-1. Phases of the Deadstart Process

Deadstart Phase	Description
System activation	System activation refers to the entire deadstart process consisting of the system initiation, production activation, and network activation phases. The system activation phase begins with the execution of CIP and the entry of system core commands, PCU subcommands, and LCU subcommands. If you enter AUTO to end system core command processing, the system automatically completes the system initiation and production activation phases of deadstart without prompting you for further input. When system activation is complete, the following message appears: `-----SYSTEM ACTIVATION COMPLETE-----`
System initiation	The system initiation phase is part of system activation, consisting of the production activation and network activation phases. If you enter GO to end system core command processing, the system prompts you during this phase to activate the system for production or to activate the system for console use only. If you do not activate the system for production, the system is available only through the system console. By choosing to activate the system for console use only, the process continues through the system initiation phase, but bypasses the production activation phase. You can activate the system for production after system activation is complete by entering the ACTIVATE_PRODUCTION_ENVIRONMENT command. This command is described in the NOS/VE Operations manual.
Production activation	The production activation phase makes the system available to users. The production activation phase may also include the network activation phase if NAM/VE is configured and if the NETWORK_ACTIVATION system attribute was not set to 0 during system core command processing.
Network activation	Network activation is the part of the production activation phase that activates NAM/VE and its associated tasks.

The following sections describe the deadstart prolog and epilog files and recommend commands you may want to include in them. All of these files reside in the catalog $SYSTEM.PROLOGS_AND_EPILOGS. These prologs and epilogs execute in a procedure file containing a condition handler that ensures that all commands execute regardless of previous errors. The system creates the status variables IGNORE_STATUS and LOCAL_STATUS for use in these prologs and epilogs.

SYSTEM_INITIATION_PROLOG File

This file executes at the beginning of the system initiation phase of deadstart. When originally installed, this file contains only comments. You can modify this file to perform any tasks appropriate for execution at the beginning of system initiation. These tasks might include the following:

- Defining attributes for the system job using the CHANGE_JOB_ATTRIBUTE command. For example, use this command to define PRINT_FILE defaults for the operator. The CHANGE_JOB_ATTRIBUTE command is described in the NOS/VE Commands and Functions manual.

- Changing job attribute defaults for all users using the CHANGE_JOB_ATTRIBUTE_DEFAULTS command. For example, use this command to change the default station name or the default login family. The CHANGE_JOB_ATTRIBUTE_DEFAULTS command is documented in the NOS/VE Operations manual.

- Changing the default date or time format using the CHANGE_DEFAULT_DATE_FORMAT and the CHANGE_DEFAULT_TIME_FORMAT commands. These commands are described in the NOS/VE Operations manual.

- Changing how SCL converts lowercase characters to uppercase. Use the CHANGE_SCL_OPTIONS command and specify the NAME_FOLDING parameter. The CHANGE_SCL_OPTIONS command is described in the NOS/VE Commands and Functions manual.

Normally, changes to the operator command list or the operator's job library list are made in the SYSTEM_INITIATION_EPILOG file. If you use the SYSTEM_INITIATION_PROLOG file to change the operator command list or the operator's job library list, you must take steps to preserve those changes. This is because installing deferred files for a batch corrective update (BCU) or installing the installation tape for an upgrade causes the operator command list and operator's job library list to be restored to their original states after the SYSTEM_INITIATION_PROLOG file executes. The variables RAV$SET_OPERATOR_COMMAND_LIST and RAV$SET_OPERATOR_LIBRARY_LIST preserve changes to the operator command list and operator's job library list you specify in the SYSTEM_INITIATION_PROLOG file. The following example shows the entries you would make in the SYSTEM_INITIATION_PROLOG file to preserve changes to the operator command list. Preserving changes to the operator's job library list can be done in a similar way.

```
var
  rav$set_operator_command_list: (xref) string
varend
rav$set_operator_command_list='create_command_list_entry..
  entry=$system.local_operator_library placement=after'
include_command command=rav$set_operator_command_list
```

JOB_ACTIVATION_PROLOG File

This file executes as the first step of the production activation phase of deadstart. When originally installed, this file contains only comments. You can modify this file to perform any tasks appropriate for execution at the beginning of the production activation phase. These tasks might include the following:

- Changing the level of information required to validate a job login using the CHANGE_VALIDATION_LEVEL command. The CHANGE_VALIDATION_LEVEL command is described in the NOS/VE Operations manual.
- Changing job scheduling attributes using the MANAGE_ACTIVE_SCHEDULING utility. The MANAGE_ACTIVE_SCHEDULING utility is described in the NOS/VE System Performance and Maintenance manual, Volume 1.

NETWORK_ACTIVATION_PROLOG File

This file executes as the first step in the network activation phase of deadstart. This file does not execute if you set the NETWORK_ACTIVATION system attribute to 0 during system core command processing. When originally installed, this file contains only comments. You can modify this file to perform any tasks appropriate for execution before activating NAM/VE.

You can include the CHANGE_NAM_ATTRIBUTES command in this file to change NAM/VE attributes such as the maximum number of connections. The CHANGE_NAM_ATTRIBUTES command is described in the NOS/VE Network Management manual.

NETWORK_ACTIVATION_EPILOG File

This file executes as the last step in the network activation phase of deadstart. This file does not execute if you set the NETWORK_ACTIVATION system attribute to 0 during system core command processing. You can modify this file to perform tasks appropriate for execution after activating NAM/VE, such as the following:

- Activating network-related applications such as SCF and PTF/QTF.
- Activating RHF if your system is running both RHFAM and NAM/VE.

When a mainframe is connected to a CDCNET network and the mainframe must load its own mainframe terminal interface (MTI), mainframe device interface (MDI) or integrated communications adaptor (ICA), the following commands must be included in the NETWORK_ACTIVATION_EPILOG file. When originally installed, the following commands appear in the NETWORK_ACTIVATION_EPILOG file. These commands are described in the NOS/VE Network Management manual.

```
ACTIVATE_NETWORK_FILE_ACCESS
ACTIVATE_NETWORK_INITIALIZER
ACTIVATE_NETWORK_LOG
ACTIVATE_NETWORK_CLOCK
```

Only one network clock command is allowed per network. You may need to make changes to the NETWORK_ACTIVATION_EPILOG file if there is more than one mainframe on the network.

JOB_ACTIVATION_EPILOG File

This file executes as the last step in the production activation phase of deadstart. When originally installed, this file contains only comments. You can modify this file to perform any tasks appropriate for execution just before the system becomes available to users. These tasks might include the following:

- Activating RHFAM, PTF, and QTF if your system is running RHFAM, but not NAM/VE. Refer to the LCN Configuration and Network Management manual for information about the commands that perform this task.
- Initiating the IM/DM kernel.

SYSTEM_INITIATION_EPILOG File

This file executes at the end of the system initiation phase of deadstart. You can modify this file to perform any tasks appropriate for execution just before the deadstart is complete. This includes the following kinds of tasks for tailoring the operator environment:

- Adding site-developed operator command libraries to the operator command list. Site-defined procedures that you want to make available to your operators should be placed in command libraries. These libraries can then be added to the operator command list using the CREATE_COMMAND_LIST_ENTRY command. The $SYSTEM.OSF$OPERATOR_COMMAND_LIBRARY file is the operator's command library and should not be used for site-defined procedures.
- Adding libraries to the operator's job library list. To make the entire installed product set available to the operator's job, the SYSTEM_PROLOG command should be added to the SYSTEM_INITIATION_EPILOG file. For example, if a site has the FORTRAN product and wants to execute FORTRAN programs at the system console, the SYSTEM_PROLOG command should be added to the SYSTEM_INITIATION_EPILOG file.
- Setting the initial displays for the system console using the VEDISPLAY command.

System Prolog and Epilog

The SYSTEM_PROLOG and SYSTEM_EPILOG files are used to tailor the environment in which user jobs execute. The SYSTEM_PROLOG executes at the beginning of all user jobs and SYSTEM_EPILOG executes as the last step before a user job terminates. NOS/VE does not create either of these files. If you want to execute a system prolog or epilog, you must create these files in the catalog $SYSTEM.PROLOGS_AND_EPILOGS. To make both of these files secure and available to all user jobs, specify EXECUTE as the access mode and (3,13,13) for the ring attributes.

The following sections describe the SYSTEM_PROLOG and SYSTEM_EPILOG files and suggested modifications for each.

SYSTEM_PROLOG File

This file executes as the first step in the execution of a user job. Before the system prolog executes, all user command lists are deleted and the $SYSTEM command list is reinstated, if necessary.

The SYSTEM_PROLOG file executes prior to the executon of any other prologs executed at user login. The following is the order of execution of any prologs that may exist:

1. System prolog
2. Job class prolog
3. Account prolog
4. Project prolog
5. User prolog

The purpose of the SYSTEM_PROLOG file is to perform the following kinds of tasks appropriate for execution before a user job begins:

- Adding a command library to users' command lists.
- Setting site default loader options.
- Activating locally developed accounting procedures.
- Changing the SCL interpreter options. For more information, refer to the CHANGE_SCL_OPTION command described in the NOS/VE Commands and Functions manual.
- Defining an application as the initial or governing application for a job.

NOTE

When adding commands to the SYSTEM_PROLOG file, be sure to specify values for the STATUS parameter. This prevents the prolog from aborting if a command fails.

If you are attaching files in the system prolog, we recommend that these files reside on a class K (service critical) volume. This allows the login to complete in the event that a non-service critical volume is missing or unavailable. For more information refer to chapter 3, Logical Configuration Utility, and chapter 11, Fault Tolerance.

Be careful to avoid creating a program loop in the SYSTEM_PROLOG file. NOS/VE prevents system prolog and epilog processing from being interrupted by a terminate break, a pause break, or a TERMINATE_JOB command; therefore, there is be no way to terminate such a loop. Similarly, there is no way a user can prevent the system from executing the system prolog or system epilog.

Site-defined procedures you want to make available to your users should be placed on a command library. These libraries can then be added to the users' command list by including the appropriate commands in the SYSTEM_PROLOG file. For example, suppose you want to add the two libraries $SYSTEM.SCU.COMMAND_LIBRARY and $SYSTEM.OSF$SITE_COMMAND_LIBRARY to all users' command lists. Your new system prolog would include the following:

```
var
ignore_status: status
varend

/create_command_list_entry entry=($system.scu.command_library ..
../$system.osf$site_command_library) placement=after status=ignore_status
```

Suppose your family name is NVE, your user name is MAIN, and you edit the file :NVE.MAIN.NEW_PROLOG to contain these commands. To make this file the new system prolog, enter the following from the NOS/VE console:

```
/copy_file :nve.main.new_prolog $system.prologs_and_epilogs.system_prolog.$next
```

If this is the first time you are creating a system prolog, you must also enter the following commands from the system console:

```
/change_file_attributes file=$system.prologs_and_epilogs.system_prolog ..
../  ring_attributes=(3,13,13)
/create_file_permit file=$system.prologs_and_epilogs.system_prolog ..
../group=public access_mode=execute
```

If you install any NOS/VE applications, you must add each application's commands to the user command lists. To do this, enter the following command in the SYSTEM_PROLOG file:

```
/$system.applications.app$setup
```

This command is described in the installation instructions for the application.

The $SYSTEM.OSF$SITE_COMMAND_LIBRARY file contains unsupported tools and procedures and should not be modified by users. This file is not automatically added to users' command lists. Documentation for some of the commands on this library is available as an online manual. To access the online manual, enter the following command:

```
/help m=osf$site_command_library
```

To print the documentation, enter the following command:

```
/print_file file=$system.manuals.line_printer_manuals.osf$site_command_library
```

During the login prolog sequence, you can specify which application is to be initiated once the prolog process has completed. For example, suppose you have an application that performs some additional system accounting (ADDITIONAL_ACCOUNTING) and you want to execute it as the initial application. You would include the following in your system prolog:

```
/define_initial_application application='ADDITIONAL_ACCOUNTING' logout_upon_termi-
nation=NO
 status
```

The first parameter, APPLICATION, specifies the call to the command that initiates the application. The second parameter, LOGOUT_UPON_TERMINATION, specifies whether the initial application is to log you out after the application has terminated. For more information on how to define an initial application, refer to the NOS/VE Object Code Management manual.

SYSTEM_EPILOG File

This file executes just before a user job terminates. The purpose of this file is perform tasks that are appropriate for execution just before a user job terminates. The following events occur before the SYSTEM_EPILOG file executes:

- All open files are closed.
- All SCL variables are deleted.
- All tape files are detached.
- All command libraries are removed from the command list.
- All file connections, except those to file $RESPONSE, are deleted.

When adding commands to the SYSTEM_EPILOG file, be sure to specify values for the STATUS parameter. This prevents the epilog from aborting if a command fails. This is particularly important for sites that initiate the Accounting Analysis System in the SYSTEM_EPILOG file.

The system epilog executes after the execution of any other epilogs executed at user logout. The following is the order of execution of any epilogs that may exist:

1. User epilog
2. Project epilog
3. Account epilog
4. Job class epilog
5. System epilog

If your site runs programs within either a job class epilog or the system epilog, you must rebuild the environment required by these programs. This includes using the CREATE_FILE_VARIABLE command to define the path to execution time libraries. For example, to run the DISPLAY_BINARY_LOG command within the system epilog you must create a file variable named CYF$RUN_TIME_LIBRARY. (This variable is also required for the ANALYZE_BINARY_LOG command.) To determine which variables you must create, examine the program's load map.

Additionally, when running compiler programs (for example, FORTRAN), you must enter the command SET_PROGRAM_ATTRIBUTES ADD_LIBRARY=OSF$TASK_SERVICES_LIBRARY prior to executing the program. Libraries containing message modules must also be added to the command list; use the CREATE_COMMAND_LIST_ENTRY command.

SYSTEM_TERMINATION_PROLOG File

You use the SYSTEM_TERMINATION_PROLOG file to tailor system termination. This file executes during the termination process that begins by entering the TERMINATE_SYSTEM command. The SYSTEM_TERMINATION_PROLOG file resides in the catalog $SYSTEM.PROLOGS_AND_EPILOGS. You can modify the SYSTEM_TERMINATION_PROLOG file to perform tasks that are appropriate for execution each time you terminate the system. When originally installed, this file contains only comments.

This prolog executes in a procedure file containing a condition handler that ensures that all commands execute regardless of errors. The system creates the status variables IGNORE_STATUS and LOCAL_STATUS for use in this epilog.

Modifying System Modules

The following system processes can be tailored to your site's needs.

- System resource unit (SRU) calculation
- Verifying Validation Names
- Password encryption
- Password validation
- Permanent file maintenance
- Magnetic tape validation

The software defining these processes is maintained in the deadstart file. This means that you must create a new deadstart file and deadstart the system to activate any changes you make. Making changes to these processes involves the following steps:

1. Copy the file $SYSTEM.NOSVE_MAINTENANCE.SOURCE_LIBRARY to the file $SYSTEM.SITE_OS_MAINTENANCE.SOURCE_LIBRARY.

   The NOSVE_MAINTENANCE.SOURCE_LIBRARY file contains the software modules that define the processes you want to modify. Because this file is rewritten each time you install a new release of NOS/VE, make changes to the file SITE_OS_MAINTENANCE.SOURCE_LIBRARY.

2. Add the file $SYSTEM.NOSVE_MAINTENANCE.COMMAND_LIBRARY to your command list. This file contains the commands you need to compile the modules that you modify.

3. Modify and compile the appropriate modules. All of the modules are written in CYBIL except the permanent file maintenance modules, which are written in SCL. Refer to the following sections for information on how to compile the modified modules.

4. Initiate a MAINTAIN_DEADSTART_SOFTWARE utility session at the system console and generate a new deadstart tape using the GENERATE_VE_DEADSTART_CATALOG and CREATE_VE_DEADSTART_TAPE subcommands.

5. Use the COLLECT_DUMP_MATERIALS subcommand to create a tape containing the link output files resulting from the creation of the new deadstart tape.

   These link output files provide a record of what changes you have made. These files are helpful to Control Data support analysts if problems develop with the system.

6. Perform a continuation deadstart using the new deadstart tape.

7. After testing the new system, place the new deadstart file on disk by initiating a MAINTAIN_DEADSTART_SOFTWARE utility session at the system console and using the ESTABLISH_DISK_BASED_SYSTEM and COMMIT_NEW_SYSTEM subcommands.

The following sections provide specific information for each of the modules that can be modified.

NOTE

The object library information is provided for informational purposes only; the compiled modules are automatically placed on the appropriate library by their respective compiler commands.

SRU Calculation

Module:	AVM$CALCULATE_SRUS
Procedure:	AVP$CALCULATE_SRUS
Compile command:	COMPILE_SRUS_ALGORITHM (CYBIL compiler is required)
Object library:	$SYSTEM.SITE_OS_MAINTENANCE.OSF$BOUND_JOB_TEMPLATE_223

Your site can specify how the system calculates system resource units (SRU) by defining a formula using any of the available NOS/VE statistics. The procedure that calculates SRUs, AVP$CALCULATE_SRUS, is called for the following reasons:

- To periodically calculate new SRU accumulator values for jobs.
- To respond to user requests for CPU time or SRU limit information.
- To respond to user requests for the current SRU accumulator value.

The maximum time interval between periodic SRU calculations is the shortest of the following time intervals:

- 100 CPU seconds.
- Half of the CPU time remaining for a job, but no less than 1 second.
- The interval calculated by the AVP$CALCULATE_SRUS procedure.

Because the AVP$CALCULATE_SRUS procedure is called frequently, SRU formulas should be simple enough so as not to require excessive computing time. Initially, the formula sets 1 second of CPU usage equal to 1 SRU.

Verifying Validation Names

Module:	AVM$VERIFY_VALIDATION_NAME
Procedure:	AVP$VERIFY_VALIDATION_NAME
Compile command:	COMPILE_VERIFY_VALIDATION_NAME
Object library:	$SYSTEM.SITE_OS_MAINTENANCE.OSF$BOUND_JOB_TEMPLATE_23D

This module allows the site to verify the format of the names used for users, accounts, and projects. Each time a new validation record is created, any user, account, or project specified is passed to the AVP$VERIFY_VALIDATION_NAME procedure. By using this procedure, a site can define criteria for validation names; for example, a site may define criteria that:

- Restrict a user, account, or project name to 7 characters.
- Prevent the use of $, _, and # in the user, account, or project name.
- Forces a particular syntax such as *aannnaa* on a user, account, or project name, where *a* is alphanumeric and *n* is numeric.

Password Encryption

Module:	AVM$ENCRYPT_PASSWORD
Procedures:	AVP$ENCRYPT_PASSWORD AVP$OLD_ENCRYPT_PASSWORD
Compile command:	COMPILE_PASSWORD_ENCRYPTION (CYBIL compiler is required)
Object library:	$SYSTEM.SITE_OS_MAINTENANCE.OSF$BOUND_JOB_TEMPLATE_23D

The password validation module has two password encryption procedures. This allows you to change the password encryption formula and also keep the prior password encryption formula. Once you create a new encryption formula, users must be notified to change their passwords by a certain date. After that date, you can replace the old password encryption formula with the new password encryption formula. Users who did not change their password can no longer log in.

To create a new encryption formula, perform the following steps:

- Change the AVP$ENCRYPT_PASSWORD procedure. Compile the AVM$ENCRYPT_PASSWORD module, create a new deadstart tape and deadstart the system as usual.

  When a user logs in, the system calls the AVP$ENCRYPT_PASSWORD procedure, which encrypts the password and compares the result to the value in the validation file. If the two values match, the job begins processing. If the two values do not match, the system calls the AVP$OLD_ENCRYPT_PASSWORD procedure to encrypt the password. The result is compared with the value in the validation file and access is granted or refused accordingly.

- Notify all users to change their passwords.

- Once the deadline for changing passwords has passed, replace the AVP$OLD_ENCRYPT_PASSWORD procedure with a copy of the AVP$ENCRYPT_PASSWORD procedure.
- Compile the AVM$ENCRYPT_PASSWORD module.
- Create a new deadstart tape and deadstart the system as before. Now the system uses only the new password encryption formula.

Password Validation

Module:	AVM$PROCESS_PASSWORD_ATTRIBUTES
Procedure:	AVP$PROCESS_PASSWORD_ATTRIBUTES
Compile command:	COMPILE_PASSWORD_CHANGE (CYBIL compiler is required)
Object library:	$SYSTEM.SITE_OS_MAINTENANCE.OSF$BOUND_JOB_TEMPLATE_23D

This module controls the way in which the system validates changed passwords. When users change their passwords, the following information is passed to the AVP$PROCESS_PASSWORD_ATTRIBUTES procedure:

- User name
- Old password
- Old encrypted password
- New password
- New encrypted password
- List of password attributes

Password attributes are site-created and are specified in the LOGIN_USER validation field. Refer to the NOS/VE User Validation manual for more information about the LOGIN_USER validation field. The AVP$PROCESS_PASSWORD_ATTRIBUTES procedure evaluates the list of password attributes and determines whether to accept the new password. In this way, a site can define criteria for password names. A site, for example, may define criteria that:

- Prevents passwords that match the user name.
- Prevents the old password from being reused.
- Prevents passwords with too few characters.
- Prevents passwords that are easily guessed from being used.

Initially, there are no restrictions on the ability of users to change their passwords.

Permanent File Maintenance

Modules:	PUP$CREATE_AGED_FILE_BACKUP PUP$CREATE_CATALOG_BACKUP PUP$CREATE_FULL_BACKUP PUP$CREATE_PARTIAL_BACKUP PUP$DELETE_EXPIRED_FILES PUP$DISPLAY_ALL_FILES PUP$LABEL_TAPE_VOLUMES PUP$RESTORE_CATALOGED_FILES PUP$RESTORE_UNRECONCILED_CATS PUP$RESTORE_UNRECONCILED_FILES
Procedure:	Not applicable
Compile command:	COMPILE_PF_MAINT_PROCS
Object library:	$SYSTEM.SITE_OS_MAINTENANCE.OSF$SOU_LIBRARY

Your site can modify the set of special SCL permanent file maintenance commands. Each module name contains the name of the command it defines. All of these commands, except LABEL_TAPE_VOLUMES, are built from either the BACKUP_PERMANENT_FILE utility or the RESTORE_PERMANENT_FILE utility. The BACKUP_PERMANENT_FILE utility and the RESTORE_PERMANENT_FILE utility are described in chapter 8, Permanent File Maintenance.

Of the commands that you can modify, the following commands are described in the NOS/VE Operations manual:

CREATE_AGED_FILE_BACKUP
CREATE_CATALOG_BACKUP
CREATE_FULL_BACKUP
CREATE_PARTIAL_BACKUP
DELETE_EXPIRED_FILES
DISPLAY_ALL_FILES
LABEL_TAPE_VOLUMES

If you modify the PUP$CREATE_FULL_BACKUP and PUP$CREATE_PARTIAL_BACKUP modules, you must follow these steps:

1. Exclude the $SYSTEM.AAM.SHARED_RECOVER_LOG catalog.
2. Back up all files.
3. Back up the $SYSTEM.AAM.SHARED_RECOVERY_LOG catalog.

The following commands are described in chapter 8, Permanent File Maintenance:

RESTORE_CATALOGED_FILES
RESTORE_UNRECONCILED_FILES
RESTORE_UNRECONCILED_CATALOGS

Magnetic Tape Validation

Module:	RMM$VALIDATE_TAPE_OPERATIONS
Procedures:	RMP$VALIDATE_TAPE_REQUEST RMP$VALIDATE_TAPE_ASSIGNMENT RMP$VALIDATE_TAPE_VOL_INIT
Compile command:	COMPILE_SITE_TAPE_HOOKS (CYBIL compiler is required)
Object library:	$SYSTEM.SITE_OS_MAINTENANCE.OSF$BOUND_JOB_TEMPLATE_23D

NOS/VE calls the preceding procedures to validate tape requests, tape assignments, and tape volume initialization. As released, these procedures do not themselves perform validation. Rather, they are meant to be replaced by sites having validation requirements not met by NOS/VE. Each validation procedure has a companion procedure that must be called to complete the validated operation. The validation procedures are called at the beginning of the tape request, tape assignment, and tape initialization process. The tape volume or volumes involved are not actually displayed to, mounted by, or initialized by the operator until after the corresponding procedure is called.

The validation and completion procedures are documented in file $SYSTEM.CYBIL.OSF$PROGRAM_INTERFACE in the following decks:

PMH$EXECUTE_WITH_LESS_PRIVILEGE
RMH$COMPLETE_TAPE_ASSIGNMENT
RMH$COMPLETE_TAPE_REQUEST
RMH$COMPLETE_TAPE_VOLUME_INIT
RMH$VALIDATE_TAPE_ASSIGNMENT
RMH$VALIDATE_TAPE_REQUEST
RMH$VALIDATE_TAPE_VOLUME_INIT

The purpose of these procedures is to give sites control over who may request, mount, or initialize a particular tape volume. This may involve accessing a database. The procedures in module RMM$VALIDATE_TAPE_OPERATIONS execute in ring 3. At ring 3, any errors in the module could have an adverse impact on the rest of the operating system. For this reason, we recommend that the validation procedures execute a task in a higher ring to perform the actual validation. The status returned by the task can then be used to determine whether the completion interface should be called, or whether abnormal status should be returned to the user. For more information, refer to the documentation of the PMP$EXECUTE_WITH_LESS_PRIVILEGE procedure in deck PMH$EXECUTE_WITH_LESS_PRIVILEGE.

Once modifications have been made and installed, the CHANGE_TAPE_VALIDATIONS command controls the enabling of these tape validation processes. The CHANGE_TAPE_VALIDATIONS command is described in the NOS/VE Operations manual. Refer to the Site Analyst Examples online manual, for an example of how to modify the tape validation process.

Tape Request Validation

When a job requests a tape, the system calls the RMP$VALIDATE_TAPE_REQUEST procedure to validate the request prior to displaying information contained in the request to the operator. The tape request passes the following information to the procedure:

- Validation state (CHANGE_TAPE_VALIDATIONS command)
- Tape file name
- Tape density
- Write ring specification
- External and recorded VSNs of all tape volumes associated with the file
- Removable media group
- Volume overflow allowed
- Validation ring
- File password
- Attachment logging specification

Using this information, you can create a CYBIL validation program and insert it into the RMP$VALIDATE_TAPE_REQUEST procedure just before the call to the RMP$COMPLETE_TAPE_REQUEST procedure.

Tape Assignment Validation

When a job opens a tape file, the system calls the RMP$VALIDATE_TAPE_ASSIGNMENT procedure to validate each tape assignment request prior to asking the operator to mount the volume. The tape assignment request passes the following information to the procedure:

- Validation state (CHANGE_TAPE_VALIDATIONS command)
- File identifier
- Tape file name
- Tape density
- Write ring specification
- Tape label type
- Tape file access mode
- Initial assignment
- Next volume number
- Next volume external and recorded VSN
- Removable media group
- Removable media location

Using this information, you can create a CYBIL validation program and insert it into the RMP$VALIDATE_TAPE_ASSIGNMENT procedure just before the call to the RMP$COMPLETE_TAPE_ASSIGNMENT procedure.

Tape Initialization Validation

When an operator initializes a tape, the RMP$VALIDATE_TAPE_VOLUME_INIT procedure is called to validate the tape initialization request. The tape initialization request passes the following information to the procedure:

- Validation state (specified by the CHANGE_TAPE_VALIDATIONS command)
- Current label information, including the density, tape element name, expiration date, file accessibility code, internal code, ANSI standard version, tape owner, recorded VSN, and the volume accessibility code.
- New label information to record on the tape including the density, tape element name, expiration date, file accessibility code, internal code, ANSI standard version, tape owner, recorded VSN, and the volume accessibility code.

Using this information, you can design a CYBIL validation program and insert it into the RMP$VALIDATE_TAPE_VOLUME_INIT procedure just before the call to the RMP$COMPLETE_TAPE_VOLUME_INIT procedure.

The procedure returns the tape initialization request status to the requesting job.

Natural Languages

You can change the natural language for product or application message modules. To change the natural language, you must create the appropriate message modules for the language you want. This means modifying a copy of the message modules that were delivered with the product. You are also responsible for translating the message text in the module. Refer to the NOS/VE Object Code Management manual for information about how to create new message modules.

Message modules reside in the library $SYSTEM.OSF$COMMAND_LIBRARY. You can identify the product that a message module supports by the first two characters in the message module name. These two characters define a product identifier. Product identifiers are listed in the NOS/VE Diagnostic Messages manual. For example, QKM$MESSAGE_TEMPLATE_MODULE is the help and diagnostic message module for IM/QUICK:

- QK is the product identifier for IM/QUICK.

- M identifies the deck as a module.

- MESSAGE_TEMPLATE_MODULE means the module contains diagnostic and help messages.

Each product has a message module for diagnostic and help messages, and may have a message module that defines function key labels. For example, QKM$US_ENGLISH is the function key label message module for IM/QUICK.

After creating the necessary modules and translating the message text into the new language, users must enter the CHANGE_NATURAL_LANGUAGE command to implement the language change.

Setting Time Zone Information

Use the following system core commands to set and subsequently display the time zone (hour, minute, and either daylight saving time or standard time). The time zone data is saved across deadstarts. These commands are described in chapter 4, System Core Commands.

SET_TIME_ZONE
DISPLAY_TIME_ZONE

In addition, you can enter the CHANGE_TIME_ZONE command to change time zone information. The CHANGE_TIME_ZONE command is documented in the NOS/VE Operations manual.

Online Manual Maintenance

The source code for the online manuals is loaded to disk as SOURCE_CODE_UTILITY (SCU) libraries when you install the ONLINE_MANUALS_SOURCE product. This product is not loaded on your system by default. These source libraries are files that contain an SCU feature that reflects the revision level documentation changes (for example, feature VER_1_5_1 reflects version 1.5.1). Not only do you benefit from a revision packet to identify new capabilities and changes, but it also aids you in the update process for any site versions of online manuals you want to create.

To tailor an online manual for your site, complete the following steps:

1. Load the ONLINE_MANUALS_SOURCE product by entering the following commands at the system console:

   ```
   install_software
   install_product p=online_manuals_source
   quit
   ```

2. Create a new source library or additional cycle of $SYSTEM.MANUALS.MAINTENANCE.MANUALS_SOURCE_LIBRARY_I or $SYSTEM.MANUALS.MAINTENANCE.MANUALS_SOURCE_LIBRARY_II reflecting the site modifications. These source libraries are structured so that decks are various sections of a manual, grouped by the manual name. The deck with the same name as the manual copies other decks belonging to that manual.

3. Execute the $SYSTEM.MANUALS.MAINTENANCE.BINDING_PROCEDURE command from the NOS/VE console.

By combining all the SCU libraries specified on the SOURCE_CATALOG parameter, this command binds manuals and saves the bound manuals. Binding puts the manuals into machine-readable format and makes them readable via the EXPLAIN utility. The specified manuals are expanded as input to the CREATE_MANUAL command. The CREATE_MANUAL command binds each manual to a file. This public file is saved under the specified catalog; the file name is the same as the manual name.

This command has the following format:

$SYSTEM.MANUALS.MAINTENANCE.BINDING_PROCEDURE
MANUALS=list of name
SOURCE_CATALOG=file
CATALOG=file
STATUS=status variable

MANUALS or *M*

Specifies the SCU deckname or decknames of manuals you want to create. The default is ALL.

SOURCE_CATALOG or *SC*

Specifies the name of the catalog where the online manuals SCU libraries reside. The default is $SYSTEM.MANUALS.MAINTENANCE.

CATALOG or *C*

Specifies the name of the catalog where the new bound manual is to reside. The default is $USER.MANUALS.

STATUS

Returns the completion status for this command.

Terminal Definitions

The terminal definitions released by Control Data are stored on the object library $SYSTEM.TDU.TERMINAL_DEFINITIONS. Enter the DISPLAY_OBJECT_LIBRARY command, specifying this file, to list the names of all released terminal definitions.

Since full screen applications load their terminal definitions from this object library, any terminal definitions that are developed by your site to be shared by all users should be added to this object library. The NOS/VE Terminal Definition manual describes how to write or modify terminal definitions, how to use the DEFINE_ TERMINAL command, and how to update an object library.

NOTE

Any site changes to the files $SYSTEM.TDU.TERMINAL_DEFINITIONS or $SYSTEM.OSF$COMMAND_LIBRARY are eliminated by the installation of corrections to the subproducts TDU and NOSVE_INITIAL_LOAD, respectively. To determine whether you need to reapply site modifications, refer to the documentation accompanying the correction. The ring attributes of files $SYSTEM.TDU.TERMINAL_ DEFINITIONS or $SYSTEM.OSF$COMMAND_LIBRARY should be (3, 13, 13).

You can add new terminal definitions or override existing terminal definitions by modifying the $SYSTEM.TDU.TERMINAL_DEFINITIONS object library. Normally, users are restricted to adding new terminal definitions to their TERMINAL_ DEFINITION files.

To permit users to override existing terminal definitions, perform the following steps to ensure that the user definition is used rather than the site definition:

1. Modify the SYSTEM_PROLOG file to include the following commands:

   ```
   if $job(job_mode)=INTERACTIVE then
     set_program_attributes al=$system.tdu.terminal_definitions ..
       $job(job_mode)=INTERACTIVE
   ifend
   ```

   These commands add $SYSTEM.TDU.TERMINAL_DEFINITIONS to the job library list for each interactive job. This ensures that users who do not choose to write their own terminal definitions have terminal definitions to use.

2. For each product that uses the library $SYSTEM.TDU.TERMINAL_DEFINITIONS, modify the product's program description to delete this library. The following are some of the products that use $SYSTEM.TDU.TERMINAL_DEFINITIONS:

   DESIGN_SCREEN
   EDIT_CATALOG
   ENTER_PPE
   ENTER_PROGRAMMING_ENVIRONMENT
   EXPLAIN
   MAIL
   QUICK
   SOURCE_CODE_UTILITY

   For each of these products, enter the CHANGE_PROGRAM_DESCRIPTION subcommand of the CREATE_OBJECT_LIBRARY utility. Use the CHANGE_PROGRAM_DESCRIPTION subcommand to include in the new program description all libraries from the old program description except $SYSTEM.TDU.TERMINAL_DEFINITIONS.

   The following commands update the program description for EDIT_CATALOG to remove the library $SYSTEM.TDU.TERMINAL_DEFINITIONS. The new cycle of $SYSTEM.OSF$COMMAND_LIBRARY has the updated program description.

   ```
   /create_object_library
   COL/add_modules l=$system.osf$command_library
   COL/display_new_library m=edit_catalog do=all "Note the library list
   COL/change_program_description n=edit_catalog l=(..
   COL../$system.edit_catalog.bound_product $system.tdu.bound_product ..
   COL../osf$task_services_library)
   COL/generate_library l=$system.osf$command_library.$next
   COL/quit
   ```

3. Inform users who want to use their own terminal definitions to add their terminal definition library to their job library list.

Determining Job Resource Limits and Default Job Classes

Values for user default job class assignments and limits on job resource usage can be established in various places. Values can be established as follows:

- For a particular user, default job classes and resource limits can be specified in the validation file. The use of validation file defaults is described in the NOS/VE User Validation manual.
- The job scheduler definition of a particular job class can define a set of limits associated with that job class. The definition of job classes is described in the NOS/VE System Performance and Maintenance manual, Volume 1.
- One or more job classes can be defined by the job scheduler as automatic job classes. If a set of automatic job classes has been defined at a site, users who are validated to use automatic job class selection have the option of allowing the system to select an appropriate job class for jobs submitted. For information about how to define automatic job classes, see the descriptions of the AUTOMATIC_CLASS_SELECTION and SELECTION_RANK scheduling attributes in the NOS/VE System Performance and Maintenance manual, Volume 1.
- Default job classes and job limits can be defined on a site-wide basis using the CHANGE_JOB_ATTRIBUTE_DEFAULTS command, described in the NOS/VE Operations manual.

The next section describes the search orders or other methods used by the system to select the appropriate job class and limits for a submitted job.

Job Class Defaults

If a site has defined a set of automatic job classes, validated users have the option of specifying the AUTOMATIC keyword on the JOB_CLASS parameter of the LOGIN, JOB, or SUBMIT_JOB command. This causes the system to select an appropriate job class based on site-selected job characteristics such as working set size or magnetic tape requirements.

When automatic job class selection is not used, a job's job class is determined by the following search order:

1. The job class explicitly defined on the LOGIN, JOB, or SUBMIT_JOB command.
2. The default job class (batch or interactive) defined for the user in the user validation file.

If a user's batch or interactive job class is defined as SYSTEM_DEFAULT in the validation file, and if the user does not explicitly define a job class on the LOGIN, JOB, or SUBMIT_JOB command, the user is assigned the site default job class defined by the CHANGE_JOB_ATTRIBUTE_DEFAULTS command.

This feature provides a way for a site to routinely change the default job class for all users at the site. If all users are assigned the SYSTEM_DEFAULT keyword in their user validation records, you only need enter a single CHANGE_JOB_ATTRIBUTE_DEFAULTS command to change all of their default job classes simultaneously.

Determining CPU Time Limits and SRU Limits

The following steps describe how the system determines either the CPU time limit or SRU limit value for a given job. These steps assume that project level validation is performed at login. If your site uses account level or user level validation, disregard references to project limits or account limits, respectively.

1. If the user has specified a limit value on the LOGIN, JOB, or SUBMIT_JOB command, this value is used as the requested limit. The system compares the requested limit with the user's validated limit, as described in step 4. If the requested limit is greater than the validated limit, the system rejects the login attempt and returns an error message to the output file or job log.

2. If no limit was explicitly requested on the LOGIN, JOB, or SUBMIT_JOB command, the default limit used is dependent on whether automatic job class selection is used.

   - If automatic job class selection is used, the system obtains the default value for CPU time limit or SRU limit from the corresponding parameter of the CHANGE_JOB_ATTRIBUTE_DEFAULTS operator command.

   - If automatic job class selection is not used, the default value for CPU time limit or SRU limit is the value defined for the job class. However, if either limit is defined for the job class as SYSTEM_DEFAULT, the value specified on the CHANGE_JOB_ATTRIBUTE_DEFAULTS command is used.

3. The system determines the user's validated limit by comparing the following limits defined in the validation file:

   - The limit defined in the user's user record.

   - The limit defined for account members in the account member record.

   - The limit defined for project members in the project member record.

   The user's validated limit is the lowest of these three limits.

4. If the user specified a limit value on the LOGIN, JOB, or SUBMIT_JOB command, the user's validated limit is compared with the requested limit, and the job is accepted or rejected as described in step 1.

5. If the user did not specify a limit value on the LOGIN, JOB, or SUBMIT_JOB command, the default limit obtained in step 2 is compared with the validated limit obtained in step 3 and the lower of these two limits is used.

Determining Magnetic Tape Limits

The magnetic tape limit for a job is determined as follows:

1. If the user has specified a value on the LOGIN, JOB, or SUBMIT_JOB command, that value is used as the limit.

2. If the user did not specify a value for magnetic tape limit, the value defined by the corresponding parameter of the CHANGE_JOB_ATTRIBUTE_DEFAULTS operator command is used. If the magnetic tape limit on the CHANGE_JOB_ATTRIBUTE_DEFAULTS command is defined as UNSPECIFIED, the limit defined for the job's job class is used.

Software Installation 7

Software Installation 7

This chapter describes the process and utilities used to install software products on your system. The INSTALL_SOFTWARE utility installs software products and maintains a database of information about each installation.

Before installing software on the system, you need to consider whether the present installation environment is suitable. The installation environment consists of access permissions, job scheduling profiles, the default family, and the IM/DM master catalog. The CREATE_INSTALLATION_ENVIRONMENT utility specifies file permissions, job scheduling profiles, and catalog requirements for the software products that need them. Specific instructions for installing a particular release or correction can be found in the accompanying installation handbook or software release bulletin. Instructions for installing a particular release or correction for an application can be found in the installation instructions for that application.

Software Orders

A software order is a collection of software products delivered on a set of tapes or transmitted on a file from another mainframe. There are two types of orders: release orders and correction orders. A release order contains complete software products that are either new to your system or replace obsolete versions of the same software already on your system. A correction order, also called a batch corrective update (BCU), contains corrections to software products currently on your system, thus maintaining the same release levels.

A release order that contains the operating system product NOS_VE is called a full order. A release order that does not contain NOS_VE is called a component order. Because a full order contains the operating system product, installing a full order is a two-step process: first, install the operating system and second, install the remaining subproducts. Refer to the NOS/VE Installation Handbook (IHB) for specific instructions on installing a full order.

Packing Lists

A packing list is a file containing information about the software on an order. Every order has a packing list which, for orders delivered on tape, resides on the first tape in the set. The first tape in the set is clearly marked on the outside of the tape. Software is listed by licensed product, subproduct, and group. A licensed product, such as NOS_VE or TCP_IP, is a unit of software that a site can purchase from Control Data. A licensed product is made up of one or more subproducts. Some licensed products, such as NOS_VE, have several subproducts. Other licensed products, such as BASIC, have only one subproduct. Subproducts are the building blocks for any software installation. Some licensed products and subproducts are combined together into what is called a group, to simplify installation. A group, such as CDCNET_SOFTWARE, is a collection of related licenced products and subproducts that can be referenced by a single name. A subproduct can belong to more than one licensed product. Similarly, a licensed product or subproduct can belong to more than one group.

The process of loading the packing list varies depending on the type of order:

- When you install a full order, the system loads the packing list for you during deadstart. You enter the packing list name, tape type, the external volume serial number, and the recorded volume serial number. During deadstart use the SET_INSTALLATION_TAPE system core command when prompted to enter system core commands, or through a set of menus that appear during the deadstart process. The SET_INSTALLATION_TAPE command is described in chapter 4, System Core Commands.

- When installing a component order or a correction order, you must load the packing list after deadstarting using the INSTALL_SOFTWARE utility subcommand LOAD_PACKING_LIST. You enter the packing list name and the specifications for the medium type (tape or file), on the LOAD_PACKING_LIST subcommand.

After installing all of the software on an order, you may want to back up the packing list to tape and then delete it from the system. If you back up and delete a packing list for a correction order, restore the packing list before attempting to install a new correction order. You can install a new correction order without the previous correction order packing list; however, this results in a slower installation.

Installation Identifiers

An installation identifier identifies a specific installation of software. When you install an order, the system creates a catalog that bears the name of the installation identifier. This installation identifier catalog contains files that contain processing information about the installation. Refer to INSTALLATION_LOGS Catalog later in this chapter.

The system assigns a new installation identifier each time you enter any of the following INSTALL_SOFTWARE utility subcommands: INSTALL_PRODUCTS, INSTALL_CORRECTIONS, or APPLY_ALL_CORRECTIONS. For example, an installation of all or part of a release order might have an installation identifier such as INSP_L716_A. INSP is the abbreviation for the INSTALL_PRODUCT subcommand that identifies the installation of a release order; L716 is the packing list name; and A indicates that this is the first installation from packing list L716.

Similarly, the identifier for the installation of a correction order might be INSC_L716AA_B. INSC is the abbreviation for the INSTALL_CORRECTION subcommand that identifies the installation of a correction; L716AA is the packing list name; and B indicates that this is the second installation from packing list L716AA.

Immediate and Deferred Installations

When you install a release order, the subproducts become available immediately upon completion of the installation job. Use the INSTALL_SOFTWARE utility subcommand INSTALL_PRODUCT to install a release order. You can install a correction order in one of the the following ways:

- Make subproduct corrections available for use immediately upon completion of the installation job.
- Defer the availability of subproduct corrections until after a future deadstart of your choosing.

NOTE

When installing corrections, be sure that users are denied access to the software being updated. Also, when installing a BCU, do not delete cycles 1 and 2 of the product files that are on your system. The BCU process uses these cycles to store the correction and the new corrected product file that is to be upgraded to your system. These cycles disappear when you activate the deferred files on your system.

An installed subproduct must be activated before it can be used. Activating an installed subproduct involves moving the contents of a subproduct file from a protected low cycle to an accessible high cycle. For a deferred installation of subproduct corrections, the file contents are left in the protected low cycle where they are unavailable for use.

Use the INSTALL_SOFTWARE utility subcommands APPLY_ALL_CORRECTIONS or INSTALL_CORRECTION to install corrections. Both of these subcommands have an INSTALLATION_OPTION parameter which enables you to specify whether to activate corrections immediately upon completion of the installation job or to defer activation. To activate deferred subproduct corrections, you must deadstart the system with a pause for operator intervention and enter the appropriate selection from a menu that appears toward the end of the deadstart process. For more information about deadstarting with a pause for operator intervention, see the NOS/VE Operations manual. The APPLY_ALL_CORRECTION and INSTALL_CORRECTIONS subcommands are described later in this chapter.

SOFTWARE_MAINTENANCE Catalog

The $SYSTEM.SOFTWARE_MAINTENANCE catalog contains the files and catalogs that support the installation process. The SOFTWARE_MAINTENANCE catalog contains the following subcatalogs:

- INSTALLATION_DATABASE
- INSTALLATION_LOGS
- CORRECTION_BASES

The following sections describe the role each catalog plays in the installation process.

INSTALLATION_DATABASE Catalog

The INSTALLATION_DATABASE catalog contains two types of files: packing list files and the installation database directory file. Figure 7-1 shows the file entries that might appear in an INSTALLATION_DATABASE catalog. In the figure, L716 is the packing list file for a release order at PSR level 716. L716AA is the packing list file for a correction order against the level 716 software. You can display the contents of a packing list file using the DISPLAY_PACKING_LIST subcommand.

The installation database directory file, RAF$IDB_DIRECTORY, contains a record of all software installed on your system. Use the INSTALL_SOFTWARE utility subcommand DISPLAY_INSTALLED_SOFTWARE to display the software currently on your system.

```
    FILE:  L716
    FILE:  L716AA
    FILE:  RAF$IDB_DIRECTORY
```

Figure 7-1. INSTALLATION_DATABASE Catalog Example

INSTALLATION_LOGS Catalog

The INSTALLATION_LOGS catalog contains a subcatalog for each installation. These subcatalogs, called installation identifier catalogs, bear the names of the installation identifiers for the installations they support. Figure 7-2 shows the installation identifier catalog and file entries for a sample INSTALLATION_LOGS catalog.

```
CATALOG:  ACTP_1988_12_07_A
   FILE:    COMMAND_LOG

CATALOG:  INSC_L716AA_A
   FILE:    COMMAND_LOG
   FILE:    JOB_LOG_VSN_ABD001
   FILE:    RAF$INSTALLATION_CONTROL_FILE
   FILE:    RAF$PROCESSING_SUMMARY_FILE

CATALOG:  INSP_L716_A
   FILE:    COMMAND_LOG
   FILE:    RAF$PROCESSING_SUMMARY_FILE

CATALOG:  INSP_L716_B
   FILE:    COMMAND_LOG
   FILE:    RAF$PROCESSING_SUMMARY_FILE

CATALOG:  INSP_L716_C
   FILE:    COMMAND_LOG
   FILE:    JOB_LOG_VSN_ABC001
   FILE:    JOB_LOG_VSN_ABC002
   FILE:    JOB_LOG_VSN_ABC003
   FILE:    RAF$INSTALLATION_CONTROL_FILE
   FILE:    RAF$PROCESSING_SUMMARY_FILE
```

Figure 7-2. INSTALLATION_LOGS Catalog Example

The catalogs shown in figure 7-2 are described as follows:

- Catalogs INSP_L716_A, INSP_L716_B, INSP_L716_C contain the files that support three separate installations from the packing list L716.
- Catalog INSC_L716AA_A contains the files that support the installation of a correction order from the packing list L716AA.
- Catalog ACTP_1988_12_07_A supports a deferred installation. A deferred installation is the activation of subproduct corrections that were installed previously, but not made available for use. Because a deferred installation can reflect more than one packing list, a different installation identifier naming scheme is used. An identifier for a deferred installation consists of the name ACTP, followed by the date on which you activated the deferred products, followed by an alphabetic increment character. In this example, the deferred installation was performed on December 7, 1988.

Each installation identifier catalog can contain four types of files: a command log, installation job logs, an installation control file, and a processing summary file.

The command log file contains validation messages, processing messages, and error messages that the system issues during the execution of the INSTALL_PRODUCT, INSTALL_CORRECTION, or APPLY_ALL_CORRECTIONS subcommands. A command log file that supports a deferred installation also contains the installation identifiers for all subproducts that were activated.

An installation job log file is a job log for an installation job. For orders installed from tape, there is one installation job for each tape volume in the set. Each job log file name bears the external volume serial number of the tape that was processed by that job. For example, in figure 7-2, the first job log file in catalog INSP_L716_C is JOB_LOG_VSN_ABC001. ABC001 is the external volume serial number of the first tape in that installation. For orders installed from a disk file, only one installation job is submitted.

An installation control file contains information that controls the execution of installation jobs. Installations that are performed during deadstart do not have an installation control file.

A processing summary file contains current status information for the installation jobs.

After you install an entire release order, you can back up the supporting installation identifier catalogs to tape and delete them to conserve disk space. However, an installation identifier catalog for a deferred installation must remain on the system; otherwise, any deferred subproduct corrections cannot be activated.

CORRECTION_BASES Catalog

The CORRECTION_BASES catalog contains copies of the original release subproduct files against which you have applied corrections. When you install a correction, the system copies the original release files affected by the correction into a subcatalog for that release of the subproduct. Each successive correction to a release is an accumulation of earlier corrections plus the new corrections that are then applied against the original release files.

Figure 7-3 shows an example of the catalog and file entries in a CORRECTION_BASES catalog. The presence of the NOSVE_MAINTENANCE subcatalog indicates that corrections have been applied against this subproduct. Within this subcatalog is another subcatalog, R141_L716, indicating the release level against which the corrections have been applied. The file entries shown in the R141_L716 catalog represent the original release subproduct files against which corrections were applied.

```
CATALOG:    NOSVE_MAINTENANCE

  CATALOG:    R141_L716
    FILE:       OSF$COMMAND_LIBRARY

    CATALOG:    LINK_INPUT_FILES
      FILE:       OS_VERSION
      FILE:       OSF$BOUND_MONITOR
```

Figure 7-3. CORRECTION_BASES Catalog Example

If you must back up the CORRECTION_BASES catalog to tape and delete it from the system to conserve disk space, you must restore it before you can install any future correction orders.

INSTALL_SOFTWARE Command and Subcommands

This section describes the INSTALL_SOFTWARE utility and its subcommands. The INSTALL_SOFTWARE command is presented first followed by its subcommands in alphabetical order.

- INSTALL_SOFTWARE command
  - APPLY_ALL_CORRECTIONS subcommand
  - CHANGE_INSTALLATION_DEFAULTS subcommand
  - CHANGE_INSTALLATION_PATH subcommand
  - DISPLAY_INSTALLED_SOFTWARE subcommand
  - DISPLAY_PACKING_LIST subcommand
  - INSTALL_CORRECTION subcommand
  - INSTALL_PRODUCT subcommand
  - LOAD_PACKING_LIST subcommand
  - QUIT subcommand

INSTALL_SOFTWARE Command

Purpose Initiates an INSTALL_SOFTWARE utility session. This utility installs software products and corrections.

Format **INSTALL_SOFTWARE** or
INSS
STATUS

Parameters *STATUS*

Returns the completion status for this utility.

Remarks You must enter this command from the system console to install NOS/VE and its related products.

APPLY_ALL_CORRECTIONS Subcommand

Purpose

Submits a batch job that installs all corrections from a specified packing list. This subcommand installs only corrections that have not already been installed. If the packing list includes corrections to the NOSVE_MAINTENANCE subproduct that have not already been installed, this subcommand also creates a new deadstart catalog or deadstart tape.

Format

APPLY_ALL_CORRECTIONS or
APPAC
  PACKING_LIST=name
  NEW_VE_DEADSTART_TAPE=record
  CONFIGURATION_FILES_CATALOG=file
  FORCE_REINSTALL=boolean
  INSTALLATION_OPTION=keyword
  SAVE_PREVIOUS_CYCLES=boolean
  STATUS=status variable

Parameters

PACKING_LIST or PL

Specifies the name of the file in the installation database catalog containing the packing list. This parameter is required.

NEW_VE_DEADSTART_TAPE or *NVDT*

Specifies a record defining the volume serial number and tape density information of the deadstart tape. This parameter is ignored if the packing list does not contain corrections to the NOSVE_MAINTENANCE subproduct. By default, if the packing list contains corrections to the NOSVE_MAINTENANCE subproduct, a deadstart catalog is created but no deadstart tape is written. The record has the following format:

(**external_vsn**, *recorded_vsn*, *type*)

external_vsn

Specifies a string defining the external volume serial number of the new deadstart tape. This field is required.

recorded_vsn

Specifies a string defining the recorded volume serial number of the new deadstart tape. The default is the value for the external_vsn field.

type

The type of tape unit required. The following keyword values can be specified; the default is MT9$6250.

MT9$1600

9-track, 1600-cpi density.

MT9$6250

9-track, 6250-cpi density.

MT18$38000

18-track, 38000-cpi density.

CONFIGURATION_FILES_CATALOG or *CFC*

Specifies the name of the catalog that contains the deadstart command files that you want to include in the deadstart catalog or tape. The default is $SYSTEM.SITE_OS_MAINTENANCE.DEADSTART_COMMANDS.

FORCE_REINSTALL or *FR*

Specifies a boolean value indicating whether subproduct corrections are to be reinstalled if already present on the system at the level indicated by the packing list. The default is FALSE.

INSTALLATION_OPTION or *IO*

Specifies a keyword indicating whether the subproduct corrections are to be activated immediately or are to be deferred. You can specify one of the following keywords; the default is DEFERRED:

IMMEDIATE or I

Activates corrected product files for production immediately.

DEFERRED or D

Defers activation of subproduct corrections until a subsequent deadstart.

NOTE

Corrections to the NOSVE_MAINTENANCE subproduct cannot be deferred using this subcommand. To defer corrections to the NOSVE_ MAINTENANCE subproduct, use the INSTALL_CORRECTION subcommand.

SAVE_PREVIOUS_CYCLES or *SPC*

Specifies a boolean value indicating whether the subproduct versions that are superceded by corrections are to be saved. The default is FALSE.

TRUE

Saves the previous subproduct version.

FALSE

Discards the previouse subproduct version.

STATUS

Returns the completion status for this subcommand.

Remarks

- Corrections can be installed only if the corresponding original release subproduct is installed on the system.
- To reinstall a previouly installed correction, you must specify the FORCE_REINSTALL parameter as TRUE.
- After you enter this subcommand, the system displays the installation identifier and the names of the jobs performing the installation. The installation identifier associated with an execution of the APPLY_ALL_ CORRECTIONS subcommand begins with the INSC abbreviation.

- If the packing list includes corrections to the NOSVE_MAINTENANCE subproduct that have not already been installed or are being installed because you specified FORCE_REINSTALL=TRUE, this subcommand creates a new deadstart catalog and a new link output files catalog. Both of these catalogs reside in the catalog SITE_OS_ MAINTENANCE.Lxxx, where *Lxxx* is the PSR summary level for the installed correction. If you also specify a value for the NEW_VE_ DEADSTART_TAPE parameter, this subcommand writes a new deadstart tape and deletes the deadstart catalog from the SITE_OS_ MAINTENANCE.Lxxx catalog.

- When creating a new deadstart tape, this subcommand includes any site tailoring changes you have made to the deadstart file software as described in chapter 6, Site Tailoring. After writing the new deadstart tape, copy the link output files to tape using the MAINTAIN_ DEADSTART_SOFTWARE utility subcommand COLLECT_DUMP_ MATERIALS. The COLLECT_DUMP_MATERIALS subcommand is described in chapter 5, Deadstart File Software.

- If you make changes to the deadstart command files DCFILE, PROLOG_LIBRARY, and PHYSICAL_CONFIG, be sure to specify the catalog that contains these files on the CONFIGURATION_FILES_ CATALOG parameter. By default, these files reside in the SITE_OS_ MAINTENANCE.DEADSTART_COMMANDS catalog. The deadstart command files are described in chapter 5, Deadstart File Software.

- You can copy a deadstart catalog to tape using the MAINTAIN_ DEADSTART_SOFTWARE utility subcommand CREATE_VE_ DEADSTART_TAPE, or you can copy the deadstart catalog to the system disk using the MAINTAIN_DEADSTART_SOFTWARE utility subcommand ESTABLISH_DISK_BASED_SYSTEM. The MAINTAIN_ DEADSTART_SOFTWARE utility is described in chapter 5, Deadstart File Software.

- Active jobs are not recovered the first time you deadstart the system using a deadstart tape or file containing corrections to the NOSVE_ MAINTENANCE subproduct.

- To activate deferred subproduct corrections, you must deadstart the system with a pause for operator intervention. During the deadstart process, a menu appears giving you the opportunity to activate deferred subproduct corrections. Refer to the NOS/VE Operations manual for more information about deadstarting with a pause for operator intervention.

- If an error occurs during the installation, a message appears in the operator action display window indicating that an error has occurred. Enter the ACKNOWLEDGE_OPERATOR_MESSAGE command to clear the message. Examine the command log or the installation job log files in the appropriate installation identifier catalog to determine the cause for the error. Use the DISPLAY_INSTALLED_SOFTWARE subcommand to determine which corrections were installed and which were not.

Examples The following example shows how to install all corrections from packing list L716AA. If corrections to the NOSVE_MAINTENANCE subproduct are included, the following example installs all corrections, defers activation, and creates a new deadstart tape.

```
/install_software
INS/apply_all_corrections packing_list=l716aa ..
INS../new_ve_deadstart_tape=('veds01')
INS/quit
```

CHANGE_INSTALLATION_DEFAULT Subcommand

Purpose

Changes the default installation catalogs (INSTALLATION_DATABASE, INSTALLATION_LOGS, or CORRECTION_BASES) for the current INSTALL_SOFTWARE utility session. The values specified on this subcommand remain in effect for the duration of the utility session.

Format

CHANGE_INSTALLATION_DEFAULT or
CHANGE_INSTALLATION_DEFAULTS or
CHAID
  CORRECTION_BASES=file
  INSTALLATION_DATABASE=file
  INSTALLATION_LOGS=file
  STATUS=status variable

Parameters

CORRECTION_BASES or *CB*

Specifies the name of the catalog that contains the original release subproduct files against which corrections have been applied. You must have the proper access permission to create catalogs and files in this catalog. The default is $SYSTEM.SOFTWARE_MAINTENANCE.CORRECTION_BASES.

INSTALLATION_DATABASE or *ID* or *IDB*

Specifies the name of the catalog that contains the packing list files and the installation database directory file RAF$IDB_DIRECTORY. The default is $SYSTEM.SOFTWARE_MAINTENANCE.INSTALLATION_DATABASE.

INSTALLATION_LOGS or *IL*

Specifies the name of the catalog that contains the installation job logs. You must have the proper access permission to create catalogs and files in this catalog. The default is $SYSTEM.SOFTWARE_MAINTENANCE.INSTALLATION_LOGS.

STATUS

Returns the completion status for this subcommand.

Remarks

- Use this subcommand with applications that the site maintains on a family and user name other than $SYSTEM.
- If you intend to install corrections and defer activation, you must use the default values for the INSTALLATION_DATABASE and INSTALLATION_LOGS parameters. Otherwise, you will be unable to activate those deferred corrections during a subsequent deadstart.

CHANGE_INSTALLATION_PATH Subcommand

Purpose Defines the family name and user name for products that permit or require a site-defined installation path. This subcommand also modifies the packing list to reflect the site-defined path. See the Remarks section for more information about which products require a site-defined installation path.

Format

CHANGE_INSTALLATION_PATH or
CHAIP
  PACKING_LIST=name
  PRODUCT=name
  FAMILY_NAME=name
  USER_NAME=name
  STATUS=status variable

Parameters

PACKING_LIST or **PL**

Specifies the name of the packing list file in the installation database catalog that contains the products for which an installation path is to be defined. This parameter is required.

PRODUCT or **P**

Specifies the name of the licensed product, subproduct, or group for which an installation path is to be defined. This parameter is required.

FAMILY_NAME or *FN*

Specifies the family name portion of the installation path for a product that permits a site-defined family name.

USER_NAME or *UN*

Specifies the user name portion of the installation path for a product that permits a site-defined user name.

STATUS

Returns the completion status for this subcommand.

Remarks

- Examine the packing list using the the DISPLAY_PACKING_LIST subcommand to determine which subproducts require a site-defined installation path.
- The system does not keep track of the original installation path after you've changed it. To determine the original installation path, refer to the product installation document or reload the packing list using the LOAD_PACKING_LIST subcommand.
- If you enter this subcommand more than once before installing products or corrections, the system uses the installation path specified by the most recent entry.
- This subcommand only affects subsequent installations.

DISPLAY_INSTALLED_SOFTWARE Subcommand

Purpose Displays information about the software that has been installed on your system.

Format **DISPLAY_INSTALLED_SOFTWARE** or
DISIS
PRODUCT =list of name or *keyword*
DISPLAY_OPTION=keyword
OUTPUT=file
STATUS=status variable

Parameters *PRODUCT* or *PRODUCTS* or *P*

Specifies the names of the licensed products, subproducts, and groups for which installation information is displayed. Not specifying this parameter, or entering the keyword ALL, displays installation information for all of the products installed on your system.

DISPLAY_OPTION or *DO*

Specifies the level of detail to be displayed about software installed on your system. You can specify one of the following keywords; the default is BRIEF:

BRIEF or B

Displays the following information:

Licensed products
Subproducts within each licensed product
Subproduct descriptions
Availability type: ACTIVE, BASE, or DEFERRED.
Release level
Installation date and time

FULL or F

Displays the following information:

Licensed products
Subproducts within each licensed product
Subproduct descriptions
Availability type: ACTIVE, BASE, or DEFERRED.
Release level
Installation date and time
Packing list
Installation identifier

OUTPUT or *O*

Specifies the name of the file to which the system writes the display information. The default is $OUTPUT.

STATUS

Returns the completion status for this subcommand.

Examples Figure 7-4 is an example of part of a full installed software display.

```
Licensed Product ACCOUNTING_ANALYSIS_SYSTEM:

   Subproduct ACCOUNTING_ANALYSIS_SYSTEM Information:

     Description:  Accounting Analysis System

     Type:                         ACTIVE
     Level:                        R141_L716
     Install Date/Time:            1988-07-21 06:36:12
     Packing List:                 L17612
     Installation Id               INSP_L17612_C

Licensed Product NOS_VE:

   Subproduct AAM Information:

     Description:  Advanced Access Method

     Type:                         ACTIVE
     Level:                        R141_L716
     Install Date/Time             1988-07-21 06:18:26
     Packing List:                 L17612
     Installation ID               INSP_L17612_A
```

Figure 7-4. Installed Software Display

DISPLAY_PACKING_LIST Subcommand

Purpose Displays the contents of a specified packing list.

Format **DISPLAY_PACKING_LIST** or
DISPL
  PACKING_LIST=name
  DISPLAY_OPTION=keyword
  OUTPUT=file
  STATUS=status variable

Parameters **PACKING_LIST or PL**

Specifies the name of the packing list file in the installation database catalog for which the contents are to be displayed. This parameter is required.

DISPLAY_OPTION or *DO*

Specifies the level of detail for the packing list display. You can specify one of the following keywords; the default is BRIEF:

BRIEF or B

Displays order information and the following licensed product information: licensed product name, release level, and storage space requirements. This display also includes the following information for subproducts that are not installed automatically or have a site-definable installation path:

Subproduct name
Automatic installation status
Release level
Storage space requirements
Installation path modification capability

FULL or F

Displays order information and the following licensed product information: licensed product name, release level, and storage space requirements. In addition, the following information is displayed for all subproducts:

Subproduct name
Subproduct description
Automatic installation status
Release level
Storage space requirements
Installation path
Installation path modification capability
Corrections and PSR numbers when applicable.

FULL also displays all group names and their subproduct membership.

OUTPUT or *O*

Specifies the name of the file to which the system writes the display information. The default is $OUTPUT.

STATUS

Returns the completion status for this subcommand.

Remarks Every packing list display includes the following information:

- Packing list name
- Installation database catalog that contains the packing list file
- Order identifier
- Order type: release or correction
- Date and time the order was created
- Medium by which the order was delivered: tape or disk file.
- External and recorded volume serial numbers of the tapes that constitute the order.

Examples Figure 7-5 is an example of part of a packing list display.

```
IDB Catalog:          :$SYSTEM.$SYSTEM.SOFTWARE_MAINTENANCE.INSTALLATION_DATABASE
Order Identifier:     TEMP_NAME
Order Type:           RELEASE
Order Creation:       1988-12-07 22:41:27
Order Medium          TAPE
External VSNs:        '9N001A', '9N001B', '9N001C'
Recorded VSNs:        '9N001A', '9N001B', '9N001C'

Licensed Product ACCOUNTING_ANALYSIS_SYSTEM             Size:     .18 Megabytes
  Level: R141_L716

  Subproduct ACCOUNTING_ANALYSIS_SYSTEM (RELEASE)   Auto Install: YES
    Description:  Accounting Analysis System
    Level: R141_L716                                    Size:     .18 Megabytes
    Path: :$SYSTEM.$SYSTEM.ACCOUNTING_ANALYSIS_SYSTEM

Licensed Product ACCOUNTING_UTILITIES                   Size:     .75 Megabytes
  Level: R141_L716

  Subproduct ACCOUNTING_UTILITIES (RELEASE)   Auto Install: YES
    Description:  Accounting Utilities
    Level: R141_L716                                    Size:     .75 Megabytes
    Path: :$SYSTEM.$SYSTEM.ACCOUNTING_AND_VALIDATION
```

Figure 7-5. Packing List Display

INSTALL_CORRECTION Subcommand

Purpose Submits a batch job that installs corrections for selected licensed products, subproducts, and groups from the specified packing list. This subcommand installs corrections that have not already been installed.

Format

INSTALL_CORRECTION or
INSTALL_CORRECTIONS or
INSC
  PACKING_LIST=name
  PRODUCT=list of name or **keyword**
  EXCLUDE_PRODUCT=list of name
  FORCE_REINSTALL=boolean
  INSTALLATION_OPTION=keyword
  SAVE_PREVIOUS_CYCLE=boolean
  STATUS=status variable

Parameters

PACKING_LIST or PL

Specifies the name of the packing list file in the installation database catalog that references the corrections to be installed. This parameter is required.

PRODUCT or PRODUCTS or P

Specifies the names of the licensed product, subproduct, and group corrections to be installed. The keyword ALL selects all subproduct corrections listed on the packing list that show an automatic installation status of YES. This parameter is required.

EXCLUDE_PRODUCT or *EXCLUDE_PRODUCTS* or *EP*

Specifies the names of licensed products, subproducts, and groups to be excluded from the list specified on the PRODUCT parameter.

FORCE_REINSTALL or *FR*

Specifies a boolean value indicating whether subproduct corrections are to be reinstalled if already present on the system at the level indicated by the packing list. The default is FALSE.

INSTALLATION_OPTION or *IO*

Specifies whether the subproduct corrections are to be activated immediately or are to be deferred until a subsequent deadstart. You can specify one of the following keywords; the default is DEFERRED:

DEFERRED or D

Defers activation of subproduct corrections until a subsequent deadstart.

IMMEDIATE or I

Activates subproduct corrections immediately upon completion of the installation job.

SAVE_PREVIOUS_CYCLES or *SPC*

Specifies a boolean value indicating whether subproduct versions that have been superceded by subproduct corrections are to be saved. The default is FALSE.

TRUE

Saves the previous subproduct versions.

FALSE

Discards the previous subproduct versions.

STATUS

Returns the completion status for this subcommand.

Remarks

- To install corrections most efficiently, install all licensed product, subproduct, and group corrections with one INSTALL_CORRECTIONS subcommand entry. This avoids having several installation jobs requesting the same tape simultaneously.
- To reinstall a previously installed correction, you must specify the FORCE_REINSTALL parameter as TRUE.
- If you intend to install corrections and activate them immediately (INSTALLATION_OPTION=IMMEDIATE), first be sure users are denied access to the subproducts being corrected.
- After you enter this subcommand, the system displays the installation identifier and the names of the jobs performing the installation.
- If you are installing a correction to the subproduct NOSVE_MAINTENANCE, be sure to specify INSTALLATION_OPTIONS=IMMEDIATE so that the corrected subproduct files are accessible when you create a new deadstart catalog using the MAINTAIN_DEADSTART_SOFTWARE utility subcommand GENERATE_VE_DEADSTART_CATALOG. The GENERATE_VE_DEADSTART_CATALOG subcommand is described in chapter 5, Deadstart File Software. If you plan to install all corrections, use the APPLY_ALL_CORRECTIONS subcommand; this automatically generates a new deadstart tape.
- Active jobs are not recovered the first time you deadstart the system using a deadstart tape or file containing corrections to the NOSVE_MAINTENANCE subproduct.
- To activate deferred subproduct corrections, you must deadstart the system with a pause for operator intervention. During the deadstart process, a menu appears giving you the opportunity to activate deferred subproduct corrections. Refer to the installation instructions in the NOS/VE Installation Handbook (IHB).

- If an error occurs during the installation, a message appears in the operator action display window indicating that an error has occurred. Enter the ACKNOWLEDGE_OPERATOR_MESSAGE command to clear the message. Examine the command log or the installation job log files in the appropriate installation identifier catalog to determine the cause for the error. Use the DISPLAY_INSTALLED_SOFTWARE subcommand to determine which corrections were installed and which were not.

INSTALL_PRODUCT Subcommand

Purpose Submits one or more batch jobs to install and activate licensed products, subproducts, and groups that have not already been installed.

Format **INSTALL_PRODUCT** or
INSTALL_PRODUCTS or
INSP
  PACKING_LIST=name
  PRODUCT=list of name or **keyword**
  EXCLUDE_PRODUCT=list of name
  FORCE_REINSTALL=boolean
  SAVE_PREVIOUS_CYCLE=boolean
  STATUS=status variable

Parameters **PACKING_LIST** or **PL**

Specifies the name of the packing list file in the installation database catalog that references the products to be installed. This parameter is required.

PRODUCT or **PRODUCTS** or **P**

Specifies the names of licensed products, subproducts, and groups to be installed. The keyword ALL selects all subproducts listed on the packing list that show an automatic installation status of YES. This parameter is required.

EXCLUDE_PRODUCT or *EXCLUDE_PRODUCTS* or *EP*

Specifies the names of licensed products, subproducts, and groups to be excluded from the list specified on the PRODUCT parameter.

FORCE_REINSTALL or *FR*

Specifies a boolean value indicating whether subproducts are to be reinstalled if already present on the system at the level indicated by the packing list. The default is FALSE.

SAVE_PREVIOUS_CYCLES or *SPC*

Specifies a boolean value indicating whether subproduct versions that have been superceded by the new subproduct release versions are to be saved. The default is FALSE.

TRUE

Saves the previous subproduct versions.

FALSE

Discards the previous subproduct versions.

STATUS

Returns the completion status for this subcommand.

Remarks

- To install software most efficiently, install all licensed products, subproducts, and groups with one INSTALL_PRODUCT subcommand entry. This avoids having several installation jobs requesting the same tape simultaneously.
- After you enter this subcommand, the system displays the installation identifier and the names of the jobs performing the installation.
- If an error occurs during the installation, a message appears in the operator action display window indicating that an error has occurred. Enter the ACKNOWLEDGE_OPERATOR_ACTION command to clear the message. Examine the command log or the installation job log files in the appropriate installation identifier catalog to determine the cause for the error. Use the DISPLAY_INSTALLED_SOFTWARE subcommand to determine which corrections were installed and which were not.
- To reinstall a previously installed product, you must specify the FORCE_REINSTALL parameter as TRUE.

LOAD_PACKING_LIST Subcommand

Purpose Copies the packing list from tape or from a disk file to a file in the installation database catalog.

Format **LOAD_PACKING_LIST or**
LOAPL
 PACKING_LIST=name
 EXTERNAL_VSN=string
 RECORDED_VSN=string
 TYPE=keyword
 DISK_FILE=file
 STATUS=status variable

Parameters **PACKING_LIST or PL**

Specifies a 1- to 16-character name for the file to which the packing list is to be copied. This packing list file resides in the installation database catalog. The default installation database catalog is $SYSTEM.SOFTWARE_MAINTENANCE.INSTALLATION_DATABASE. This parameter is required.

EXTERNAL_VSN or *EVSN* or *EV*

Specifies the external volume serial number of the tape volume containing the packing list. If the packing list resides on tape, either this parameter or the RECORDED_VSN parameter is required.

RECORDED_VSN or *RVSN* or *RV*

Specifies the recorded volume serial number of the tape volume containing the packing list. If the packing list resides on tape, either this parameter or the EXTERNAL_VSN parameter is required.

TYPE or *T*

Specifies the type of tape containing the packing list. You can specify the following keywords; the default is MT9$6250:

MT9$1600

9-track, 1600-cpi density.

MT9$6250

9-track, 6250-cpi density.

MT18$38000

18-track, 38000-cpi density.

DISK_FILE or *DF*

Specifies the name of the permanent file containing the packing list and software order. This parameter is required only when a software order is transmitted electronically directly to your mainframe, rather than using tapes.

STATUS

Returns the completion status for this subcommand.

Remarks

- Because a packing list name is used to identify and track individual installations, these names must be unique and should not be reused.
- A packing list accompanies every software order regardless of whether the order is delivered on tape or on a file transmitted from another mainframe. When the order is delivered on tape, the packing list always resides on the first tape in the set.
- Use the DISPLAY_PACKING_LIST subcommand to display the contents of an order.

QUIT Subcommand

Purpose Terminates the INSTALL_SOFTWARE utility session.

Format **QUIT** or
QUI
STATUS=status variable

Parameters *STATUS*

Returns the completion status for this subcommand.

CREATE_INSTALLATION_ENVIRONMENT Command and Subcommands

This section describes the CREATE_INSTALLATION_ENVIRONMENT command and its subcommands. The CREATE_INSTALLATION_ENVIRONMENT command is presented first, followed by the subcommands in alphabetical order:

- CREATE_INSTALLATION_ENVIRONMENT command
  - CHANGE_CONFIG_FILE_ACCESS subcommand
  - CREATE_DEFAULT_FAMILY subcommand
  - CREATE_DMROOT_USER_NAME subcommand
  - CREATE_PROFILE_FOR_UPGRADE subcommand
  - CREATE_SCHEDULING_CLASSES subcommand
  - QUIT subcommand

CREATE_INSTALLATION_ENVIRONMENT Command

Purpose Initiates a CREATE_INSTALLATION_ENVIRONMENT utility session. This utility prepares the system for the installation of certain products.

Format **CREATE_INSTALLATION_ENVIRONMENT** or
CREIE
STATUS=status variable

Parameters *STATUS*

Returns the completion status for this utility.

Remarks You must enter this command from the system console.

CHANGE_CONFIG_FILE_ACCESS Subcommand

Purpose Creates or deletes file access permission to the current physical configuration file and the deadstart command files. Refer to the Remarks section for information about the specific files and access permissions.

Format

CHANGE_CONFIG_FILE_ACCESS or
CHACFA
  CONFIGURATION_FILE_ACCESS=boolean
  FAMILY_NAME=name
  USER_NAME=name
  STATUS=status variable

Parameters

CONFIGURATION_FILE_ACCESS or **CFA**

Specifies a boolean value indicating whether the access permissions to the physical configuration file and the deadstart command files are to be created or deleted. This parameter is required.

ON

Creates file permits.

OFF

Deletes file permits.

FAMILY_NAME or **FN**

Specifies the family name of the user for whom you are creating or deleting file access permissions. This parameter is required.

USER_NAME or **UN**

Specifies the name of the user for whom you are creating or deleting file access permissions. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks The CONFIGURATION_FILE_ACCESS parameter creates or deletes the following file and catalog access permissions:

- READ access to the current physical configuration file, $SYSTEM.MAINFRAME.CONFIGURATION. Refer to chapter 2, Physical Configuration Utility, for more information about this file.
- READ, APPEND, MODIFY, SHORTEN, WRITE, EXECUTE, CYCLE, and CONTROL access to catalog $SYSTEM.SITE_OS_MAINTENANCE.DEADSTART_COMMANDS. This catalog contains the deadstart command files PHYSICAL_CONFIG, PROLOG_LIBRARY, and DCFILE. The deadstart command files are described in chapter 5, Deadstart File Software.

CREATE_DEFAULT_FAMILY Subcommand

Purpose

Creates the default login family for batch and interactive jobs and defines the family administrator user name for the default family. Enter this command when installing the first release order on a mainframe.

Format

CREATE_DEFAULT_FAMILY or
CREDF
  FAMILY_NAME=name
  USER_NAME=name
  PASSWORD=name
  ACCOUNT_NAME=name
  PROJECT_NAME=name
  PERMANENT_FILE_SET=name
  STATUS=status variable

Parameters

FAMILY_NAME or **FN**

Defines the name of the family. The family name must be unique among all family names in the system. In a network configuration, this name must be unique within the network. This parameter is required.

USER_NAME or **UN**

Defines the administrator user name for the family given by the FAMILY_NAME parameter. The system creates this user name. This parameter is required.

PASSWORD or **PW**

Defines the password for the family administrator user name. This parameter is required.

ACCOUNT_NAME or *AN*

Defines the account name established for the default family user administrator. The default is NONE.

PROJECT_NAME or *PN*

Defines the project name and overrides the default name established for the default family user administrator. The default is NONE.

PERMANENT_FILE_SET or *PFS*

Specifies the name of the mass storage set of which the family is to be a member. The default is the system set.

STATUS

Returns the completion status for this subcommand.

Remarks

- This subcommand copies the current physical configuration file $SYSTEM.MAINTENANCE.CONFIGURATION to the file $SYSTEM.SITE_OS_MAINTENANCE.DEADSTART_COMMANDS.PHYSICAL_CONFIG.
- This subcommand grants the following access permissions to the user you specify on the USER_NAME and FAMILY_NAME parameters:
  - READ access to the current physical configuration file, $SYSTEM.MAINFRAME.CONFIGURATION. Refer to chapter 2, Physical Configuration Utility, for more information about this file.
  - READ, APPEND, MODIFY, SHORTEN, WRITE, EXECUTE, CYCLE, and CONTROL access to catalog $SYSTEM.SITE_OS_MAINTENANCE.DEADSTART_COMMANDS. This catalog contains the deadstart command files PHYSICAL_CONFIG, PROLOG_LIBRARY, and DCFILE. The deadstart command files are described in chapter 5, Deadstart File Software.
- The family administrator user name receives permission to access the physical configuration file and the deadstart command files for the purpose of creating deadstart files. Refer to chapter 5, Deadstart File Software, for more information about creating deadstart files.
- If you plan to use the account or project level validation, we recommend that you specify values for the account name and project name. For more information, refer to the NOS/VE User Validation manual.

CREATE_DMROOT_USER_NAME Subcommand

Purpose Creates the user name DMROOT in preparation for installing IM/DM.

Format **CREATE_DMROOT_USER_NAME** or
CREDUN
  PASSWORD=name
  FAMILY=name
  ACCOUNT_NAME=name
  PROJECT_NAME=name
  SCHEDULING_CLASS=name
  STATUS=status variable

Parameters **PASSWORD** or **PW**

Specifies the password to be assigned to the user name DMROOT. This parameter is required.

FAMILY or *F*

Specifies the name of the family of which the DMROOT user name is a member. The default is NVE.

ACCOUNT_NAME or *AN*

Defines the account name established for the DMROOT user name. The default is NONE.

PROJECT_NAME or *PN*

Defines the project name established for the DMROOT user name. The default is NONE.

SCHEDULING_CLASS or *SC*

Specifies the scheduling class in which the IM/DM kernel is to execute. The default is DM_KERNEL.

STATUS

Returns the completion status for this subcommand.

Remarks

- Before installing IM/DM, use this subcommand to create the DMROOT user name.
- The minimum ring privilege for the DMROOT user name is 4.
- The job class name for the DMROOT user name is the same as the name given for the SCHEDULING_CLASS parameter.
- If you plan to use the account or project level validation, we recommend that you specify values for the account name and project name. For more information, refer to the NOS/VE User Validation manual.

CREATE_PROFILE_FOR_UPGRADE Subcommand

Purpose

Creates a temporary scheduling profile that restricts access to the system while you are installing software. This prevents users from using mixed levels of software while products are being installed. By default, this scheduling profile enables only two scheduling classes: SYSTEM and MAINTENANCE. You can enable other scheduling classes.

Format

CREATE_PROFILE_FOR_UPGRADE or
CREPFU
ENABLE_SITE_CLASSES=list of record
SAVED_PROFILE=file
STATUS=status variable

Parameters

ENABLE_SITE_CLASSES or *ESC*

Specifies one or more records that indicate which scheduling classes are to be enabled in addition to SYSTEM and MAINTENANCE. Each record has the following format:

(class_name, job_count)

class_name

Specifies the name of the scheduling class to be enabled. This field is required.

job_count

Specifies the number of jobs to be allowed for the scheduling class. This field is required.

SAVED_PROFILE or *SP*

Specifies the name of the file containing the profile to be saved for restoration after the completion of the upgrade. The default is $SYSTEM.SCHEDULING_PROFILE.

STATUS

Returns the completion status for this subcommand.

Remarks

- The SYSTEM and MAINTENANCE classes are each allowed to initiate 10 jobs.
- After installing software, use the MANAGE_ACTIVE_SCHEDULING utility to restore the scheduling profile your site normally uses. The MANAGE_ACTIVE_SCHEDULING utility is described in the NOS/VE System Performance and Maintenance manual, Volume 1. Refer to the Examples section for an example of how this is done.
- If you are installing IM/DM, use the ENABLE_SITE_CLASSES parameter to specify the scheduling class for the IM/DM kernel job.

Examples The following example creates a modified version of the scheduling profile and saves the current scheduling profile onto the file $SYSTEM.SCHEDULING_PROFILE. In addition to the scheduling classes SYSTEM and MAINTENANCE, this new profile enables the scheduling classes SITE_CLASS_1 and DM_KERNEL. SITE_CLASS_1 permits a maximum of five jobs; DM_KERNEL permits a maximum of four jobs.

```
/create_installation_environment
creie/create_profile_for_upgrade enable_site_classes=..
creie../((site_class_1,5) (dm_kernel,4))
creie/quit
```

After you have installed all software and are ready to begin production, restore the original scheduling profile by entering the following commands:

```
/manage_active_scheduling
manas/activate_profile profile=$system.scheduling_profile ..
manas../enable_job_reclassification=true
manas/quit save_change=true
```

CREATE_SCHEDULING_CLASSES Subcommand

Purpose Creates a scheduling profile to support the software products IM/DM, NFS, PTF/QTF, or TCP/IP. This subcommand either modifies an existing profile in the $SYSTEM.SCHEDULING_PROFILE file or creates a new one if the file does not exist.

Format **CREATE_SCHEDULING_CLASSES** or
CRESC
OPTION=list of keyword
IM_DM_SCHEDULING_CLASS=name
ACTIVATE_PROFILE=boolean
STATUS=status varable

Parameters *OPTION* or *O*

Specifies either the creation of a new scheduling profile or modifications to an existing profile in the $SYSTEM.SCHEDULING_PROFILE file. You can specify one or more of the following keywords; the default is DEFAULT:

DEFAULT

Creates the default scheduling profile. The default scheduling profile is described in the NOS/VE System Performance and Maintenance manual, Volume 1.

IM_DM

Adds instructions to the existing profile to create the scheduling class for IM/DM.

NFS

Adds instructions to the appropriate scheduling definitions for the Network File System (NFS).

PTF_QTF

Adds instructions to the existing profile to create the scheduling class for PTF/QTF.

TCP_IP

Adds instructions to the existing profile to create the scheduling class for TCP/IP.

ALL

Creates the default scheduling profile and adds the scheduling classes for IM/DM, NFS, PTF/QTF, and TCP/IP. Do not enter ALL in combination with any other values.

IM_DM_SCHEDULING_CLASS or *IDSC*

Specifies the IM/DM scheduling class. The default is DM_KERNEL. This parameter is ignored if IM_DM is not one of the values for the OPTION parameter.

ACTIVATE_PROFILE or *AP*

Specifies a boolean value indicating whether the scheduling profile is to be activated. The default is TRUE.

TRUE

Activates the scheduling profile.

FALSE

Does not activate the scheduling profile.

STATUS

Returns the completion status for this subcommand.

Remarks

- Enter this subcommand to set the scheduling profile for your site, and then enter the CREATE_PROFILE_FOR_UPGRADE subcommand to create a special scheduling profile for the installation.
- If you create scheduling classes for PTF/QTF or TCP/IP, this subcommand creates a job class, job category, and service class with the name FILE_TRANSFER. The job class and service class have the same characteristics as the BATCH class, except the INITIATION_LEVEL and MAXIMUM_ACTIVE_JOBS scheduling attributes are set to 40.
- If you create scheduling classes for IM/DM, this subcommand creates a job class and service class with the name given by the IM_DM_SCHEDULING_CLASS parameter. The job class and service class have the same characteristics as the MAINTENANCE class.
- This subcommand activates the scheduling profile only if ACTIVATE_PROFILE=TRUE and there were no errors in creating the scheduling classes. An attempt to create a job class that already exists does not prevent the system from activating the profile.
- If you create scheduling classes for NFS, this subcommand creates two job classes and service classes with the names NETWORK_FILE_SYSTEM and PORTMAP_PCNFS. Both have the same characteristics as the SYSTEM class except that the NETWORK_FILE_SYSTEM service class has the DISPATCHING_CONTROL service class attribute set to ((P9,UNLIMITED,1,1)).

QUIT Subcommand

Purpose Ends the CREATE_INSTALLATION_ENVIRONMENT utility session.

Format **QUIT** or
QUI
STATUS=status variable

Parameters *STATUS*

Returns the completion status for this subcommand.

Permanent File Maintenance 8

Permanent File Maintenance 8

This chapter describes the tasks and commands required to perform analyst-level permanent file maintenance functions on a NOS/VE system. Operator-level tasks, such as performing a full backup, partial backup, or catalog backup are described in the NOS/VE Operations manual.

For the purposes of this chapter, the term *permanent files* includes families, master catalogs, subcatalogs, files, and file cycles.

This chapter is divided into four major sections:

- A section on creating and maintaining NOS/VE families.
- A section that describes mass storage sets.
- A reference section that describes commands used to restore catalogs and files if a disk fails.
- A reference section that describes the BACKUP_PERMANENT_FILES and RESTORE_PERMANENT_FILES utilities.

Permanent file maintenance commands are subcommands of the System Operator Utility. To execute permanent file maintenance commands, you must enter the command in a System Operator Utility session and have the appropriate validation capability. A System Operator Utility session can be initiated either at a terminal or at the system console. If a System Operator Utility session is initiated at the system console, you are given all validation capabilities by default.

To initiate the System Operator Utility, enter:

```
system_operator_utility
```

The system responds with the prompt: sou/. The System Operator Utility is discussed in the NOS/VE Operations manual. The manual also has an appendix that lists both the System Operator Utility subcommands and the commands that have an extended scope in a System Operator Utility session. The subcommands and commands are grouped by validation capability.

The reference sections on the BACKUP_PERMANENT_FILES and RESTORE_PERMANENT_FILES utilities are provided for sites that want to write their own procedures to back up and restore permanent files.

The process of restoring permanent files after an installation deadstart is described in chapter 13, Repair Solutions.

Managing Families

This section describes how to change and delete family names.

Changing Family Names

Changing a family name may occasionally be necessary, for example, to make all family names unique within a network. To change a family name, perform the following steps at the system console or a terminal. At a terminal, you need to be validated for the CONFIGURATION_ADMINISTRATION, NETWORK_OPERATION, SCHEDULING_ADMINISTRATION, SYSTEM_ADMINISTRATION, and SYSTEM_OPERATION capabilities.

1. Enter the following commands to prevent the initiation of new jobs for all job classes except the SYSTEM job class. Note that the MANAGE_ACTIVE_SCHEDULING utility requires the SCHEDULING_ADMINISTRATION capability, but does not require you to be in a System Operator Utility session.

   ```
   /manage_active_scheduling
   MAS/change_job_class class_name=all enable_class_initiation=false
   MAS/change_job_class class_name=system enable_class_initiation=true
   MAS/quit
   ```

2. Send a message to all interactive users requesting that they log out. Refer to the NOS/VE Operations manual for information about sending a message to interactive users.

3. Initiate the System Operator Utility:

   ```
   /system_operator_utility
   sou/
   ```

4. Monitor the user job activity on the Initiated Jobs Display. Make note of jobs that remain after users have been requested to log out.

   ```
   sou/vedisplay display_options=initiated_jobs
   ```

5. Terminate interactive and batch jobs that have not logged out after sufficient warning. For example, to terminate the job $0855_0002_AAA_0001, enter the following command:

   ```
   sou/terminate_job name=$0855_0002_AAA_0001 job_state=all
   ```

6. If your site uses MAIL/VE Version 1, perform the following steps to prepare for the transfer of MAIL/VE Version 1 mailboxes to the new family:

   a. Log in at an interactive terminal under the MAIL/VE Version 1 administrator user name.

b. Create and execute the following procedure. This procedure compiles a list of all mailboxes on the current family and builds the file $LOCAL.REGU_FILE. $LOCAL.REGU_FILE contains the REGISTER_USER subcommands.

```
procedure  create_register_users_file, creruf (
  new_family_name, nfn : name = $required
  )

  push interaction_style
  change_interaction_style s=line
  mail p=$null
    display_mailbox mn=* sm=private do=mailbox_name df=keyword ..
          o=$local.regu_file
  quit
  edit_file f=$local.regu_file p=$null o=$null
    replace_text t='mn=' ..
          nt='register_user fn='//$strrep($value(nfn))//' un=' l=a
  quit
  put_line l=..
  ' File $LOCAL.REGU_FILE has the REGISTER_USER subcommands.'

procend create_register_users_file
```

If you created this procedure on file $LOCAL.CRERUF, enter the following command to create the REGISTER_USER subcommands for users on the new family:

```
sou/creruf new_family_name=nve_1
```

7. Enter the appropriate set of commands to change the family name, depending on the condition of the current permanent file base. If the current file base is acceptable, change the family name and the file permits. Otherwise, restore the permanent files in addition to changing the family name and file permits. In the following examples, the family name is being changed from NVE to NVE_1.

   - To change the family name and the file permits without restoring files, enter the following commands:

```
sou/change_family family_name=nve new_family_name=nve_1
sou/job job_name=chap job_class=system
job/system_operator_utility
job/restore_permanent_files
job/change_all_permits family_name=nve new_family_name=nve_1
job/quit
job/quit
job/jobend
```

- To change the family name and file permits, and restore permanent files, enter the following commands:

```
sou/job job_name=respr job_class=system
job/system_operator_utility
job/request_magnetic_tape file=$local.tape recorded_vsn=('back01','back02')
job/restore_permanent_files
job/restore_catalog catalog=:nve new_catalog_name=:nve_1 ..
job../backup_file=$local.tape
job/change_all_permits family_name=nve new_family_name=nve_1
job/quit
job/quit
job/jobend
```

  When the job is complete, enter the following command to create the new family and administrator. If you have more than one mass storage set on your system, you must specify the name of the the mass storage set you want on the PERMANENT_FILE_SET parameter.

```
sou/create_family family_name=nve_1 family_user_administrator=klm101
```

8. Enter the following commands to complete the transfer of MAIL/VE Version 1 mailboxes to the new family:

```
sou/task ring=4
sou/create_command_list_entry entry=$system.mailve.maintenance.command_library
sou/maintain_mail
mm/add_family family_name=nve_1 host_name=nve_1
mm/include_file file=$local.regu_file
mm/quit
sou/taskend
```

   Notify MAIL/VE users that they can gain access to their old mailboxes by entering the full path for the old mailbox. For example, the old path to mailbox TKELLY would be :NVE.TKELLY. In screen mode, users would enter the path name on the "Mailbox name:" line of the CHECK MAIL screen. In line mode, a user would enter the following command after the mail/ prompt:

```
/display_letter mailbox_name=:nve.tkelly
```

   The old family and its mailboxes remain on the mail system until you delete the old family using the MAIL/VE maintenance subcommand REMOVE_FAMILY.

9. Change the default login family if necessary. For example, if you are changing the family name NVE to NVE_1, and NVE is the default login family, enter the following command to make NVE_1 the default login family:

```
sou/change_job_attribute_defaults login_family=nve_1
```

10. In the following files, replace any references to the old family name with the new family name:

    SYSTEM_INITIATION_PROLOG
    SYSTEM_INITIATION_EPILOG
    JOB_ACTIVATION_PROLOG
    JOB_ACTIVATION_EPILOG
    NETWORK_ACTIVATION_PROLOG
    NETWORK_ACTIVATION_EPILOG
    SYSTEM_TERMINATION_PROLOG
    SYSTEM_PROLOG
    SYSTEM_EPILOG

    All of these files reside in catalog $SYSTEM.PROLOGS_AND_EPILOGS.

11. If your site is using NAM/VE, change network application definitions to refer to the new family name. This includes definitions for BTFS and PTF/QTF applications. Refer to the NOS/VE Network Management manual for more information on how to change network application definitions.

12. If your site is using IM/DM, update the IM/DM kernel jobs to reflect the new family name using the DM_CHANGE_KERNEL_JOB_ATTRIBUTES command. The DM_CHANGE_KERNEL_JOB_ATTRIBUTES command is described in the IM/DM Data Administration manual.

13. If your site is using IM/Control, enter the LOCATE_DM command to specify the new family so that IM/Control can locate IM/DM. The LOCATE_DM command is described in the NOS/VE Software Release Bulletin.

14. Instruct family administrators to change references to the old family name in the prolog and epilog validation entries in the validation files.

15. Instruct all users to change path names referencing the old family name. References are typically found in user or editor prolog files and in program descriptions on object library files.

16. For systems running VX/VE, edit the VX/VE password file, $SYSTEM.VX.ETC.PASSWD, changing the old family name in each entry to the new family name. Refer to the VX/VE Administrator Guide and Reference manual for more information about the VX/VE password file.

17. For NOS dual-state systems, if the NOS/VE family name does not match the NOS family name, instruct all users to enter the CHANGE_LINK_ATTRIBUTES command, specifying the NOS family name. This provides the correct NOS family name to NOS/VE and allows file transfers between the two operating systems. If the NOS/VE family does not exist on NOS, the default NOS family is used. The CHANGE_LINK_ATTRIBUTES command is described in the NOS/VE Commands and Functions manual. The ADMINISTER_VALIDATIONS subcommand CHANGE_LINK_ATTRIBUTES, which is described in the NOS/VE User Validation manual, can be used also.

    NOTE

    If you have only one NOS family, edit the partner job template files to reference that family. The partner job template files are described in the NOS 2 Installation Handbook.

18. Enable the initiation of jobs for all job classes. Enter the following utility command and subcommands:

```
sou/manage_active_scheduling
MAS/change_job_class class_name=all enable_class_initiation=true
MAS/change_job_class class_name=unassigned enable_class_initiation=false
MAS/quit
sou/quit
```

Deleting a Family Name

The following example describes how to delete a family name. Enter the commands either at the system console or a terminal. At a terminal, you need to have the following validation capabilities: SCHEDULING_ADMINISTRATION, SYSTEM_OPERATION, and either FAMILY_ADMINISTRATION or SYSTEM_ADMINISTRATION. This example deletes the family NVE.

1. Prevent the initiation of new jobs for all job classes except the SYSTEM job class. Enter the following utility command and subcommands. The MANAGE_ACTIVE_SCHEDULING utility requires the SCHEDULING_ADMINISTRATION capability, but does not require you to be in a System Operator Utility session.

```
/manage_active_scheduling
MAS/change_job_class class_name=all enable_class_initiation=false
MAS/change_job_class class_name=system enable_class_initiation=true
MAS/quit
```

2. Send a message to all interactive users requesting that they log out. Refer to the NOS/VE Operations manual for information about sending a message to interactive users.

3. Initiate the System Operator Utility:

```
/system_operator_utility
sou/
```

4. Monitor the user job activity on the Initiated Jobs Display. Make note of jobs that remain after users have been requested to log out.

```
sou/vedisplay display_options=initiated_jobs
```

5. Terminate jobs that have not been logged out after sufficient warning. For example, to terminate the job $0855_0001_AAA_0001, enter the following command:

```
sou/terminate_job name=$0855_0002_AAA_0001 job_state=all
```

6. Enter the following commands to delete the family name and the family's master catalogs, subcatalogs, and files:

```
sou/job job_name=delete_family_files job_class=system
job/system_operator_utility
job/task ring=3
job/backup_permanent_files backup_file=$null
job/include_empty_catalogs delete_catalogs=true
job/include_master_catalogs delete_master_catalogs=true
job/delete_catalog_contents catalog=:nve
job/quit
job/taskend
job/quit
job/jobend
sou/quit
```

7. Enter the following commands to enable the initiation of jobs for all job classes:

```
/manage_active_scheduling
MAS/change_job_class class_name=all enable_class_initiation=true
MAS/change_job_class class_name=unassigned enable_class_initiation=false
MAS/quit
```

Commands That Manage Family Names

This section describes the commands that change and create family names. The following commands are described in this section:

CHANGE_FAMILY
CREATE_FAMILY

To execute these commands, you need to be in the System Operator Utility and have the SYSTEM_ADMINISTRATION capability.

CHANGE_FAMILY Command

Purpose Changes the permanent file family name.

> **CHANGE_FAMILY**
> **FAMILY_NAME=name**
> **NEW_FAMILY_NAME=name**
> *STATUS=status variable*

Parameters

FAMILY_NAME or FN

Specifies the name of the family to be changed. The FAMILY_NAME value cannot be $SYSTEM. This parameter is required.

NEW_FAMILY_NAME or NFN

Specifies the new name to be used for the family. This family name must not already exist. This parameter is required.

STATUS

Returns the completion status for this command.

Remarks

- Use of this command requires the System Operator Utility to be active with the SYSTEM_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- Enter this command when no user jobs are executing or initiated and the system is not actively connected through CDCNET.

Examples This example changes the family name NVE to NVE_1:

```
sou/change_family_name family_name=nve new_family_name=nve_1
```

CREATE_FAMILY Command

Purpose Creates a family name and establishes the family administrator.

Format

CREATE_FAMILY
  FAMILY_NAME=name
  FAMILY_ADMINISTRATOR=name
  ACCOUNT=name
  PROJECT=name
  PASSWORD=name
  PERMANENT_FILE_SET=name
  STATUS=status variable

Parameters

FAMILY_NAME or **FN**

Specifies the name of the family. The family name must be unique among all family names in the system. This parameter is required.

FAMILY_ADMINISTRATOR or **FAMILY_USER_ADMINISTRATOR** or **FA** or **FUA**

Specifies the user name to serve as the administrator for the family given by the FAMILY_NAME parameter. The system creates this user name. The default is the name given for the FAMILY_NAME parameter. This parameter is required.

ACCOUNT or *ACCOUNT_NAME* or *A* or *AN*

Specifies the account name to be assigned to the administrator user name. The default is ACCOUNT.

PASSWORD or *PW*

Specifies the password for the administrator user name. The default is PLEASE_CHANGE_THIS_PASSWORD_NOW.

PERMANENT_FILE_SET

Specifies the name of the mass storage set on which the family's permanent files are to reside. The default is the name for the system set of devices.

STATUS

Returns the completion status for this command.

Remarks

- Use of this command requires the System Operator Utility to be active with the SYSTEM_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- For NOS dual-state with access through NAM, the family administrator must have identical user names on NOS and NOS/VE. The passwords should also be identical.
- For NOS/BE dual-state, the password specified for the NOS/VE administrator is unknown to the NOS/BE side of the system and is not required for interactive login through INTERCOM to NOS/VE. The NOS/VE password is required for NOS/VE batch jobs.
- The CREATE_FAMILY command has no abbreviation.

Examples

This example creates the family NVE and establishes the family administrator user name KLM101:

```
sou/create_family family_name=nve family_administrator=KLM101
```

Mass Storage Sets

You can back up and restore files on a mass storage set. A mass storage set is a logical grouping of physical disk volumes that are reserved for one or more families. Mass storage sets are described in chapter 3, Logical Configuration Utility.

Backing Up Permanent Files by Sets

Backing up permanent files by sets makes it possible to restore sets to the same mainframe or to another mainframe. Because a family can belong to only one set, you can also restore complete families in this way. The BACKUP_PERMANENT_FILES utility subcommand BACKUP_SET backs up all permanent files on a specified active set. The following is an example of the commands you enter to back up the set NVESET to tape. Enter the commands at the system console or a terminal. At a terminal, you need to be validated for the SYSTEM_ADMINISTRATION capability:

```
/system_operator_utility
sou/task ring=3
sou/request_magnetic_tape file=$local.tape evsn=('tape01 'tape02') ..
sou../ring=true type=mt9$6250
sou/backup_permanent_files backup_file=$local.tape list=$local.list
PUB/backup_set set_name=nveset
PUB/quit
sou/taskend
sou/quit
```

If the permanent files in the set include permanent tape files, the catalog entry information for the permanent tape files is backed up. The tapes are not accessed.

Restoring Permanent Files by Sets

Restoring a mass storage set effectively restores the families that are members of that set. Families that exist on the system are restored to the sets on which they are defined. For families that do not exist, the system attempts to create the families on a set as follows:

- If the family being restored is $SYSTEM, it is created on the system set.
- If there is only one set active on the system, the family is created on that set.
- If more than one set is active and one of the sets is named NVESET, the family is created on the set NVESET.

If none of the these conditions apply, you must explicitly create the family and assign it to a mass storage set using the CREATE_FAMILY command, then restore the family's permanent files.

You can move sets between mainframes only if the set names between those mainframes are unique. In addition, family names must also be unique between the two sets.

When a mainframe goes down, you can recover its sets from another mainframe. However, if you want to perform online maintenance on the down mainframe, you may have to install the CYBER Initialization Package (CIP) on a volume that is currently not in use.

Example: Restoring a Set to Another Mainframe Using Backup Tapes

You can restore a mass storage set to another mainframe from backup tapes containing the set. For example, suppose you have backed up the set VE_SET to tape and you want to restore this set to another mainframe for testing purposes. VE_SET set contains the families GREEN, PINK, and BLUE. Perform the following steps on the test mainframe. Enter the commands at the system console or a terminal. At a terminal, you need to be validated for the SYSTEM_ADMINISTRATION capability.

1. Create a set on the test mainframe, in this example, TEST_SET. Refer to chapter 3, Logical Configuration Utility, for information about how to create a set.

2. Enter the following commands to create the families as members of the set TEST_SET:

```
/system_operator_utility
sou/create_family family_name=green family_administrator=admin01 ..
sou../permanent_file_set=test_set
sou/create_family family_name=pink family_administrator=admin01 ..
sou../permanent_file_set=test_set
sou/create_family family_name=blue family_administrator=admin01 ..
sou../permanent_file_set=test_set
```

3. Enter the following commands to restore the families to the set TEST_SET from the backup tapes containing the set VE_SET:

```
sou/task ring=3
sou/request_magnetic_tape file=$local.tape evsn=('tape01' tape02') ..
sou../ring=false type=mt9$6250
sou/restore_permanent_files l=$local.restore_list
PUR/restore_all_files backup_file=$local.tape
PUR/quit
sou/taskend
sou/quit
```

Restoring a Family to a Different Set

To change a family's set membership, perform the following steps. Enter the commands at the system console or a terminal. At a terminal, you need to be validated for the SYSTEM_ADMINISTRATION capability. In this example, the family RED is moved to an existing set VE_SET.

1. Enter the following commands to back up the family RED:

```
/system_operator_utility
sou/task ring=3
sou/request_magnetic_tape file=$local.tape evsn=('tape01' tape02') ..
sou../ring=true t=mt9$6250
sou/backup_permanent_files backup_file=$local.tape list=$local.list
PUB/backup_catalog catalog=:red
PUB/quit
sou/taskend
```

2. Enter the following commands to delete the family:

```
sou/task ring=3
sou/backup_permanent_files backup_file=$null list=$local.list
PUB/include_empty_catalogs delete_catalogs=true
PUB/include_master_catalogs delete_catalogs=true
PUB/delete_catalog_contents catalog=:red
PUB/quit
sou/taskend
```

3. Enter the following command to create the family RED in the set VE_SET:

```
sou/create_family family_name=red family_administrator=tkw87 ..
sou../permanent_file_set=ve_set
```

4. Enter the following commands to restore the permanent files for family RED from the tapes created in step 1.

```
sou/task ring=3
sou/request_magnetic_tape file=$local.tape evsn=('tape01' tape02') ..
sou../ring=false t=mt9$6250
sou/restore_permanent_files list=$local.restore_list
PUR/restore_all_files backup_file=$local.tape
PUR/quit
sou/taskend
sou/quit
```

Recovering a Set on Another Mainframe

If you have two mainframes cabled to a set of common disk units, it is possible to recover a mass storage set directly from one mainframe to the other mainframe in the event that one mainframe fails. This process is called alternate set recovery. It provides immediate access to users' permanent files after a mainframe failure.

In order to perform alternate set recovery, the physical configurations for both mainframes must include element definitions for all disk units in the set. The element states must be defined as ON for one mainframe and OFF for the other mainframe.

If a mainframe fails and you perform an alternate recovery, there are some disadvantages that can be avoided if you wait for the failed mainframe to become available:

- The files are recovered without image. This can cause the following possible problems:
  - The modified pages of data that were not written to disk before the failure may be lost.
  - Up to the last 30 seconds of file allocation information may be lost.
  - The end-of-information may be rounded up to a multiple of the file's allocation size.
- Jobs executing on the mainframe that failed can never be recovered after an alternate set recovery.
- For families in the set, the files in the input queues are not recovered until the next deadstart.

- Files in the output queue are not available on the alternate mainframe because they reside in the $SYSTEM catalog and the $SYSTEM family of the failed mainframe is not accessible.

- If your site uses the NOS/VE File Server, access to server files is terminated for jobs on all client mainframes using the families being recovered. These users must logout and log back in to the server mainframe to regain access to these files.

Sites considering configuring their mainframes for alternate set recovery must decide whether these limitations will cause more inconvenience than waiting for the failed mainframe to return to service. Refer to Analyze Critical Mainframe Failure in chapter 12, Failure Analysis, for additional information.

Example: Moving a Set to Another Mainframe Through Common Disks

If mainframes A and B are cabled to common disk units, you can move a set directly from mainframe A to mainframe B as long as the physical configurations for both mainframes include the element definitions of the common disk units. The element states for the disk units on one mainframe must be ON; on the other mainframe the element states must be OFF.

Assuming the set is active on mainframe A and mainframe A fails, perform the following steps on mainframe B to recover the set:

1. Change the element states of the common disk units to ON.
2. Activate the set on mainframe B using the ACTIVATE_SET command. All families that exist on the restored set that do not already exist on mainframe B are now available on mainframe B.

To return the set to mainframe A, perform the following steps:

1. Terminate mainframe B.
2. Deadstart mainframe B with a pause for operator intervention. Choose to intervene before activating existing active set members from the NOS/VE Reconfiguration Menu. Modify the logical configuration by changing the element states of the common disk units to OFF. Refer to the NOS/VE Operations manual for more information about pausing for operator intervention during deadstart.
3. Deadstart mainframe A.

Commands That Restore Files After a Failure

This section describes the commands that restore permanent files if a disk fails. These commands are listed as follows:

RESTORE_CATALOGED_FILES
RESTORE_UNRECONCILED_CATALOGS
RESTORE_UNRECONCILED_FILES

To execute these commands, you need to be in the System Operator Utility and have the SYSTEM_ADMINISTRATION capability.

RESTORE_CATALOGED_FILES Command

Purpose

Submits a batch job or task that restores permanent files from backup tapes. This command does not restore file cycles that already exist on the system. This command uses backup tapes created by the CREATE_FULL_BACKUP, CREATE_PARTIAL_BACKUP, or CREATE_CATALOG_BACKUP commands.

Format

RESTORE_CATALOGED_FILES or
RESCF or
RES
  RESTORE_CATALOGS=boolean
  VSN_PREFIX=name, string, or **integer**
  VSN_COUNT=integer
  VSN_SUFFIX=name, string, or *integer*
  INCREMENT_SCHEME=keyword
  FILE_LABEL_TYPE=keyword
  TYPE=keyword
  EXECUTION_MODE=keyword
  OUTPUT=file
  STATUS=status variable

Parameters

RESTORE_CATALOGS or **RC**

Specifies whether to restore catalog information from the set of backup tapes. This parameter is required. When restoring files from two or more sets of backup tapes, specify TRUE for the most recent backup tape set, and FALSE for all others.

TRUE

Restores catalog information and file cycle data.

FALSE

Restores just file cycle data.

VSN_PREFIX or **VSNP**

Specifies the 1- to 5-character prefix that all backup tape volume serial numbers have in common. The system completes the volume serial numbers by generating the characters that follow the prefix. This parameter is required.

VSN_COUNT or *VSNC*

Specifies the number of tapes in the backup tape set. The default is 9.

VSN_SUFFIX or *VSNS*

Specifies the appropriate characters that, when appended to the value given by the VSN_PREFIX parameter, identify the tape with which to restart a failed restore attempt. Use this parameter only when you need to restart the restore process with a tape other than the first tape in the set. This value must be an integer when INCREMENT_SCHEME=DECIMAL, and a name or string when INCREMENT_SCHEME=ALPHABETIC.

INCREMENT_SCHEME or *IS*

Specifies the format of the system-generated characters following the volume serial number prefix. You can specify the following keywords; the default is DECIMAL:

DECIMAL or D

Generates leading zeros (as needed) and integers to complete the volume serial numbers.

ALPHABETIC or A

Generates letters to complete the volume serial numbers. The advantage of specifying ALPHABETIC is that more labels can be generated for a given prefix.

FILE_LABEL_TYPE or *FLT*

Specifies whether the backup tapes are labelled or unlabelled. You can specify the following keywords; the default is the system default tape label type:

LABELLED or L

Specifies labelled tapes.

UNLABELLED or U

Specifies unlabelled tapes.

TYPE or *T*

Specifies the type of tape unit required. You can specify the following keywords; the default is MT9$6250:

MT9$800

9-track, 800-cpi density.

MT9$1600

9-track, 1600-cpi density.

MT9$6250

9-track, 6250-cpi density.

MT18$38000

18-track, 38000-cpi density.

EXECUTION_MODE or *EM*

Specifies the mode of execution for the task restoring the data. You can specify the following keywords; the default is BATCH_JOB:

ASYNCHRONOUS_TASK or AT

The task that restores the data is submitted as an asynchronous task.

BATCH_JOB or BJ

The task that restores the data is submitted as a batch job. A batch job cannot restore the data if a file or catalog that is required for job initiation is not accessible.

SYNCHRONOUS_TASK or ST

The task that restores the data is submitted as a synchronous task.

OUTPUT or O

Specifies the file that records the output from the restore session. This parameter is required if the EXECUTION_MODE parameter specifies ASYNCHRONOUS_TASK or SYNCHRONOUS_TASK. Otherwise, the default is $LIST.

STATUS

Returns the completion status for this command.

Remarks

- Use of this command requires the System Operator Utility to be active with the SYSTEM_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- When restoring permanent files after performing an installation deadstart, some system files are not restored. This is because the deadstart creates these system files. Consequently, when you attempt to restore a file that already exists, a message to that effect appears on the console screen and the file is not restored. For more information about performing an installation deadstart and restoring permanent files, see chapter 13, Repair Solutions.
- To display the default tape label type, enter the DISPLAY_BACKUP_LABEL_TYPE command. To change the default tape label type, enter the CHANGE_BACKUP_LABEL_TYPE command. Both commands are described in the NOS/VE Commands and Functions manual.
- Use this command to restore all permanent files after an installation deadstart. However, because the installation deadstart creates certain critical files, subsequent restoring of all permanent files may produce error messages indicating that these existing files were not restored.

Examples

- This example illustrates how to restore permanent files. The most recent backup is a partial backup and the backup before that was a full backup. Begin by restoring from the partial backup tapes.

```
sou/restore_cataloged_files restore_catalogs=true ..
sou../vsn_prefix=part1 vsn_count=6 increment_scheme=decimal ..
sou../file_label_type=labelled type=mt9$1600
```

After the partial backup tapes have been read, enter the following command to restore files from the full backup tape set:

```
sou/restore_cataloged_files restore_catalogs=false ..
sou../vsn_prefix=full vsn_count=12 increment_scheme=decimal ..
sou../file_label_type=labelled type=mt9$1600
```

- Suppose the job to restore files in the previous example aborted while tape FULL09 was being read. You can restart the restore process, beginning with FULL09, by entering the following command:

```
sou/restore_cataloged_files restore_catalogs=false ..
sou../vsn_prefix=full vsn_count=4 vsn_suffix=09 ..
sou../increment_scheme=decimal file_label_type=labelled ..
sou../type=mt9$1600
```

RESTORE_UNRECONCILED_CATALOGS Command

Purpose

Submits a batch job or task to restore catalog information and, as an option, file cycle data. This command restores permanent files for one or more devices that are missing or unavailable. A missing device is one that was DOWN or OFF during the previous deadstart. An unavailable device is one that has gone down since the deadstart. Permanent files are restored to the remaining devices in the system.

This command uses backup tapes created by the CREATE_FULL_BACKUP, CREATE_PARTIAL_BACKUP, and CREATE_CATALOG_BACKUP commands.

Format

RESTORE_UNRECONCILED_CATALOGS or
RESUC or
RESMC
  VSN_PREFIX=name, string, or **integer**
  VSN_COUNT=integer
  VSN_SUFFIX=name, string, or *integer*
  INCREMENT_SCHEME=keyword
  FILE_LABEL_TYPE=keyword
  TYPE=keyword
  RESTORE_EXCLUDED_FILE_CYCLES=list of keyword
  EXECUTION_MODE=keyword
  OUTPUT=file
  STATUS=status variable

Parameters

VSN_PREFIX or VSNP

Specifies the 1- to 5-character prefix that all backup tape volume serial numbers have in common. The system completes the volume serial numbers by generating the characters that follow the prefix. This parameter is required.

VSN_COUNT or *VSNC*

Specifies the number of tapes in the backup tape set. This default is 9.

VSN_SUFFIX or *VSNS*

Specifies the appropriate characters that, when appended to the value given by the VSN_PREFIX parameter, identify the tape with which to restart a failed restore attempt. Use this parameter only when you need to restart the restore process with a tape other than the first tape in the set. This value must be an integer when INCREMENT_SCHEME=DECIMAL, and a name or string when INCREMENT_SCHEME=ALPHABETIC.

INCREMENT_SCHEME or *IS*

Specifies the format of the system-generated characters following the volume serial number prefix. You can specify the following keywords; the default is DECIMAL:

DECIMAL or D

Generates leading zeros (as needed) and integers to complete the volume serial numbers.

ALPHABETIC or A

Generates letters to complete the volume serial numbers. The advantage of specifying ALPHABETIC is that more labels can be generated for a given prefix.

FILE_LABEL_TYPE or *FLT*

Specifies whether the backup tapes are labelled or unlabelled. You can specify the following keywords; the default is the system default tape label type:

LABELLED or L

Specifies labelled tapes.

UNLABELLED or U

Specifies unlabelled tapes.

TYPE or *T*

Specifies the type of tape unit required. You can specify the following keywords; the default is MT9$6250:

MT9$800

9-track, 800-cpi density.

MT9$1600

9-track, 1600-cpi density.

MT9$6250

9-track, 6250-cpi density.

MT18$38000

18-track, 38000-cpi density.

RESTORE_EXCLUDED_FILE_CYCLES or *REFC*

Specifies the conditions under which to restore file cycle data. This parameter is used only when the missing or unavailable devices contain both catalog information and file cycle data. You can specify the following keywords; the default is the list (MEDIA_MISSING, NO_DATA_DEFINED):

NONE or N

Restores just catalog information from the backup tapes.

MEDIA_MISSING or MM

Restores file cycle data for a missing device.

NO_DATA_DEFINED or NDD

Restores data for file cycles that are defined in a catalog but have no data.

VOLUME_UNAVAILABLE or VU

Restores file cycle data for an unavailable device.

EXECUTION_MODE or *EM*

Specifies the mode of execution for the task restoring the data. You can specify the following keywords; the default is BATCH_JOB:

ASYNCHRONOUS_TASK or AT

The task that restores the data is submitted as an asynchronous task.

BATCH_JOB or BJ

The task that restores the data is submitted as a batch job. A batch job cannot restore the data if a file or catalog that is required for job initiation is not accessible.

SYNCHRONOUS_TASK or ST

The task that restores the data is submitted as a synchronous task.

OUTPUT or O

Specifies the file that records the output from the restore session. This parameter is required if the EXECUTION_MODE parameter specifies ASYNCHRONOUS_TASK or SYNCHRONOUS_TASK. Otherwise, the default is $LIST.

STATUS

Returns the completion status for this command.

Remarks

- Use of this command requires the System Operator Utility to be active with the SYSTEM_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- To restore permanent files, execute this command for all partial backup tape sets that have been created since the previous full backup. Begin with the most recent partial backup tape set and work backward using each successive partial backup tape set.

When all partial backup tape sets have been read, execute the RESTORE_UNRECONCILED_CATALOGS command and use the most recent set of full backup tapes. When all full backup tapes have been read, enter the following commands:

```
sou/restore_permanent_files
PUR/set_restore_missing_catalogs operation=end
PUR/quit
```

- File cycles that reside on catalogs that have been restored are marked with the damage conditions PARENT_CATALOG_RESTORED and RESPF_MODIFICATION_MISMATCH. Users can remove these conditions using the CHANGE_CATALOG_ENTRY command or the CHANGE_CATALOG_CONTENTS command. You can remove these damage conditions from user files using the CHANGE_CATALOG_CONTENTS command at the system console. The CHANGE_CATALOG_ENTRY command is described in the NOS/VE System Usage manual. The CHANGE_CATALOG_CONTENTS command is described in the NOS/VE Commands and Functions manual.
- This command cannot be used to restore permanent files on the system device.
- If the missing or unavailable device contains only catalogs, you can restore the catalogs by executing this command using the most recent set of backup tapes that contain catalog information.
- To display the default tape label type, enter the DISPLAY_BACKUP_LABEL_TYPE command. To change the default tape label type, enter the CHANGE_BACKUP_LABEL_TYPE command. Both commands are described in the NOS/VE System Usage manual.

Examples

This example restores catalog information for an unavailable catalog device. The tapes from the most recent catalog backup are used.

```
sou/restore_unreconciled_catalogs vsn_prefix=ctlg vsn_count=3 ..
sou../restore_excluded_file_cycles=none
```

When all tapes have been read, enter the following commands to complete the catalog restoration process:

```
sou/restore_permanent_files
PUR/set_restore_missing_catalogs operation=end
PUR/quit
```

RESTORE_UNRECONCILED_FILES Command

Purpose

Submits a batch job or task to restore file cycle data for one or more disk volumes that are missing or unavailable. A missing device is one that was DOWN or OFF during the previous deadstart. An unavailable device is one that has gone down since the deadstart. File cycle data is restored to the remaining devices in the system.

This command uses backup tapes created by the CREATE_FULL_BACKUP and the CREATE_PARTIAL_BACKUP commands.

Format

RESTORE_UNRECONCILED_FILES or
RESUF or
RESLC
  VSN_PREFIX=name, string, or **integer**
  VSN_COUNT=integer
  VSN_SUFFIX=name, string, or *integer*
  INCREMENT_SCHEME=keyword
  FILE_LABEL_TYPE=keyword
  TYPE=keyword
  RECORDED_VSNS=list of name
  RESTORE_OPTIONS=list of keyword
  EXECUTION_MODE=keyword
  OUTPUT=file
  STATUS=status variable

Parameters

VSN_PREFIX or **VSNP**

Specifies the 1- to 5-character prefix that all backup tape volume serial numbers have in common. The system completes the volume serial numbers by generating the characters that follow the prefix. This parameter is required.

VSN_COUNT or *VSNC*

Specifies the number of tapes in the backup tape set. The default is 9.

VSN_SUFFIX or *VSNS*

Specifies the appropriate characters that, when appended to the value given by the VSN_PREFIX parameter, identify the tape with which to restart a failed restore attempt. Use this parameter only when you need to restart the restore process with a tape other than the first tape in the set. This value must be an integer when INCREMENT_SCHEME=DECIMAL, and a name or string when INCREMENT_SCHEME=ALPHABETIC.

INCREMENT_SCHEME or *IS*

Specifies the format of the system-generated characters following the volume serial number prefix. You can specify the following keywords; the default is DECIMAL:

DECIMAL or D

Generates leading zeros (as needed) and integers to complete the volume serial numbers.

ALPHABETIC or A

Generates letters to complete the volume serial numbers. The advantage of specifying ALPHABETIC is that more labels can be generated for a given prefix.

FILE_LABEL_TYPE or *FLT*

Specifies whether the backup tapes are labelled or unlabelled. You can specify the following keywords; the default is the system default tape label type:

LABELLED or L

Specifies labelled tapes.

UNLABELLED or U

Specifies unlabelled tapes.

TYPE or *T*

Specifies the type of tape unit required. You can specify the following keywords; the default is MT9$6250:

MT9$800

9-track, 800-cpi density.

MT9$1600

9-track, 1600-cpi density.

MT9$6250

9-track, 6250-cpi density.

MT18$38000

18-track, 38000-cpi density.

RECORDED_VSNS or *RECORDED_VSN* or *RVSN*

Specifies the volume serial numbers of the volumes for which file cycle data is to be restored. The default is all volumes that are missing or unavailable.

RESTORE_OPTIONS or *RO*

Specifies the conditions under which file cycle data is to be restored. You can specify the following keywords; the default is the list (MEDIA_MISSING, NO_DATA_DEFINED):

MEDIA_MISSING or MM

Restores file cycle data for a missing device.

NO_DATA_DEFINED or NDD

Restores data for file cycles that are defined in a catalog but have no data.

VOLUME_UNAVAILABLE or VU

Restores file cycle data for an unavailable device.

EXECUTION_MODE or *EM*

Specifies the mode of execution for the task restoring the data. You can specify the following keywords; the default is BATCH_JOB:

ASYNCHRONOUS_TASK or AT

The task that restores the data is submitted as an asynchronous task.

BATCH_JOB or BJ

The task that restores the data is submitted as a batch job. A batch job cannot restore the data if a file or catalog that is required for job initiation is not accessible.

SYNCHRONOUS_TASK or ST

The task that restores the data is submitted as a synchronous task.

OUTPUT or O

Specifies the file that records the output from the restore session. This parameter is required if the EXECUTION_MODE parameter specifies ASYNCHRONOUS_TASK or SYNCHRONOUS_TASK. Otherwise, the default is $LIST.

STATUS

Returns the completion status for this command.

Remarks

- Use of this command requires the System Operator Utility to be active with the SYSTEM_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

- To display the default tape label type, enter the DISPLAY_BACKUP_LABEL_TYPE command. To change the default label type, enter the CHANGE_BACKUP_LABEL_TYPE command. Both commands are described in the NOS/VE Commands and Functions manual.

- Restored file cycles that have been modified since the most recent backup are marked with the RESPF_MODIFICATION_MISMATCH damage condition. Users can remove this damaged condition from their files using the CHANGE_CATALOG_ENTRY command or the CHANGE_CATALOG_CONTENTS command. You can remove these damage conditions from user files using the CHANGE_CATALOG_CONTENTS command at the system console. The CHANGE_CATALOG_ENTRY command is described in the NOS/VE Commands and Functions manual. The CHANGE_CATALOG_CONTENTS command is described in the NOS/VE Operations manual.

Examples

This example restores the unreconciled files from the missing disk volume VOL123:

```
sou/restore_unreconciled_files vsn_prefix=ABC ..
sou../vsn_count=5 recorded_vsn=vol123 ..
sou../restore_options=(media_missing no_data_defined)
```

Backing Up and Restoring Permanent Files

How you choose to back up and subsequently restore permanent files depends on the configuration at your site. Factors that affect your backup and restore requirements include whether you have multiple mainframes configured together and how your disk volumes are divided into mass storage sets. The BACKUP_PERMANENT_FILES (BACPF) and RESTORE_PERMANENT_FILES (RESPF) utilities are the tools to create the procedures to back up and restore permanent files according to your site's needs.

For sites using Archive/VE, see the NOS/VE File Archiving manual for information about archiving file cycles and releasing file cycle data from mass storage.

BACKUP_PERMANENT_FILES Utility

This section describes the BACKUP_PERMANENT_FILES utility and its subcommands. When permanent tape files are selected for backup, only the catalog entry information is backed up; the tape is not accessed.

To backup files for users besides yourself, you need to execute the BACKUP_ PERMANENT_FILES utility within a System Operator Utility session and have either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION capability. To initiate the System Operator Utility session, enter:

```
/system_operator_utility
```

The system responds with the prompt: sou/. For more information on the System Operator Utility, see the NOS/VE Operations manual.

The BACKUP_PERMANENT_FILES utility command and subcommands are as follows:

```
BACKUP_PERMANENT_FILES command
   BACKUP_ALL_FILES subcommand
   BACKUP_CATALOG subcommand
   BACKUP_FILE subcommand
   BACKUP_SET subcommand
   DELETE_ALL_FILES subcommand
   DELETE_CATALOG_CONTENTS subcommand
   DELETE_FILE_CONTENTS subcommand
   EXCLUDE_CATALOG subcommand
   EXCLUDE_FILE subcommand
   EXCLUDE_HIGHEST_CYCLES subcommand
   INCLUDE_CYCLES subcommand
   INCLUDE_EMPTY_CATALOGS subcommand
   INCLUDE_LARGE_CYCLES subcommand
   INCLUDE_MASTER_CATALOGS subcommand
   INCLUDE_SMALL_CYCLES subcommand
   INCLUDE_USERS subcommand
   INCLUDE_VOLUMES subcommand
   QUIT subcommand
   SET_BACKUP_OPTIONS subcommand
   SET_LIST_OPTIONS subcommand
   SORT_USERS subcommand
```

NOTE

A new software version often includes new parameters that change the position-dependent format for a subcommand. To ensure that your backup procedures function as expected from release to release, specify parameter names in all subcommand entries.

Performance Considerations:

If you back up files to labelled, open-reel magnetic tapes, you can decrease the time necessary for the system to back up the files as well as decrease the number of tape volumes needed to record the data by changing the block size file attribute of the backup file. A block size of 32,640 bytes gives optimal performance. With that block size, a backup may use up to 94.6 percent of the tape and 94.4 percent of the tape unit's speed. These improvements are very noticeable for full or partial file backups. Improvements may be negligible for catalog backups.

For example, a full backup that requires 100 tapes when the block size is 4,128 bytes (the default value) only needs 62 tapes when the block size is 32,640 bytes, a decrease of 38 percent. In addition, when using a tape unit with a recording density of 6250 cpi and a recording rate of 200 inches per second, the job time and elapsed time for backing up a file greater than 1 megabyte is reduced 30 percent when the block size is changed from 4,128 bytes to 32,640 bytes.

To change the block size of the backup file, use the SET_FILE_ATTRIBUTES command before you enter the BACKUP_PERMANENT_FILES utility. For example, to increase the block size, enter:

```
/set_file_attributes file=backup_file_name block_type=us ..
../file_label_type=labelled maximum_block_length=32640 record_type=variable
```

Although you can specify any block size, we recommend that you select one that is a multiple of 960 bytes.

To use a large block size, you need the proper hardware configuration. A 639, 679, or 698 tape unit needs two peripheral processors. A 5698 or 9639 tape unit needs one peripheral processor. For information about defining tape units and peripheral processors, see the DEFINE_ELEMENT subcommand in chapter 2, Physical Configuration Utility.

NOTE

If you changed the hardware configuration to back up files, it is necessary to maintain that hardware configuration in order to restore those files. For example, files backed up on a 679 tape unit that has two peripheral processors can be restored only on a tape unit that has two peripheral processors.

If you back up files to cartridge tapes, you do not have to change the block size. For cartridge tapes, the default block size is 32,640 bytes, which is optimal.

If you back up files to unlabelled, open-reel tapes, you are constrained to a block size of 4,128 bytes.

If you want to modify the Control Data backup procedures, refer to the Site Tailoring section in chapter 6, Site Tailoring.

BACKUP_PERMANENT_FILES Command

Purpose Initiates execution of the utility that backs up permanent files and catalogs. Further processing is directed by BACKUP_PERMANENT_FILES 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 disk file or a labelled tape. If no file position is specified or the file is an unlabelled tape, then the file is initially positioned to beginning-of-information. This parameter is required.

LIST or *L*

Specifies the name of the file to receive summary information about the backup. This is file is called the backup listing. The default is $LIST.

STATUS

Returns the completion status for the utility.

Remarks

- Use of this command to back up files other than your own requires the System Operator Utility to be active with either the SYSTEM_ ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

- After entering the BACKUP_PERMANENT_FILES command, the following prompt appears:

```
PUB/
```

Examples This example initiates a BACKUP_PERMANENT_FILE utility session. The command specifies that the backed up files are to be written to file BACKED_UP_FILES with the backup listing written to file BACKUP_ LISTING.

```
sou/backup_permanent_files backup_file=$local.backed_up_files ..
sou../list=backup_listing
```

BACKUP_ALL_FILES Subcommand

Purpose Backs up all catalogs and permanent files in the system.

Format **BACKUP_ALL_FILES** or
BACAF
STATUS=status variable

Parameters *STATUS*

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to back up files other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- The BACKUP_ALL_FILES subcommand skips a file cycle if the file cycle cannot be accessed with an access mode of read and a share mode of read and execute.
- Enter the INCLUDE_USERS subcommand before the BACKUP_ALL_FILES subcommand to back up a specific set of users.
- Any catalog, file, or cycle specified on a previous EXCLUDE_CATALOG, EXCLUDE_FILE, or EXCLUDE_HIGHEST_CYCLE subcommand is not backed up.
- Enter the INCLUDE_CYCLES subcommand before the BACKUP_ALL_FILES subcommand to backup a specific set of file cycles.
- The INCLUDE_CYCLES subcommand enables you to create a partial backup. A partial backup includes all catalog information and all permanent file cycles modified after the date specified on the INCLUDE_CYCLES subcommand. This catalog information is used when restoring files to select file cycles from the previous partial or full backup.
- You can use the INCLUDE_CYCLES, INCLUDE_SMALL_CYCLES, INCLUDE_VOLUME, INCLUDE_LARGE_CYCLES, EXCLUDE_HIGHEST_CYCLE, or INCLUDE_EMPTY_CATALOGS subcommands in conjunction with a subsequent DELETE_ALL_FILES subcommand to reduce the number of files maintained online.

Examples This example backs up all catalogs and permanent files to the backup file COMPLETE_BACKUP:

```
/system_operator_utility
sou/task ring=3
sou/backup_permanent_files backup_file=$local.complete_backup
PUB/backup_all_files
PUB/quit
sou/taskend
sou/quit
```

BACKUP_CATALOG Subcommand

Purpose Backs up all subcatalogs, files, and file cycles in the 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.

STATUS

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to back up files other than your own requires the System Operator Utility to be active with either the SYSTEM_ ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- The BACKUP_CATALOG subcommand skips a file cycle if the file cycle cannot be accessed 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_SMALL_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 sites using Archive/VE:

  If the catalog you want to back up contains file cycles for which data has been released from mass storage, enter the SET_BACKUP_ OPTIONS subcommand and specify INCLUDE_DATA=ALL to include the file cycle catalog entries in the backup.

Examples

This example backs up all files in the master catalog DLM313 in family NVE:

```
sou/backup_permanent_files backup_file=$local.backup ..
sou../list=$local.backup_listing
PUB/backup_catalog catalog=:nve.dlm313
PUB/quit
```

BACKUP_FILE Subcommand

Purpose Backs up a permanent file.

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

Parameters **FILE or F**

Specifies the permanent file or file cycle to be backed up. This parameter is required.

PASSWORD or *PW*

Specifies the password for the file to be backed up. The keyword NONE specifies no password. NONE is the default. When you enter this subcommand as the $SYSTEM user, this parameter is not needed.

STATUS

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to back up files other than your own requires the System Operator Utility to be active with either the SYSTEM_ ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- If you specify a single file cycle, the backup does not include the file access log, password, permits, or accounting information. If you specify a file without a cycle reference, all file cycles are backed up and the backup includes the file access log, password, permits, and accounting information.
- The BACKUP_FILE subcommand skips a file cycle if the file cycle cannot be accessed with an access mode of read and a share mode of read and execute.
- The EXCLUDE_FILE subcommand specifies cycles to exclude from the backup operation.
- Previous INCLUDE_CYCLES, INCLUDE_SMALL_CYCLES, INCLUDE_VOLUME, INCLUDE_LARGE_CYCLES, and EXCLUDE_ HIGHEST_CYCLE subcommands limit the number of file cycles backed up by the BACKUP_FILE subcommand.

Examples This example backs up cycle number 87 of a file for user TKWS87:

```
sou/backup_permanent_file backup_file=$local.backup
PUB/backup_file file=.tkws87.catalog_1.data_file_0.87
PUB/quit
```

BACKUP_SET

Purpose Backs up all permanent files in a specified mass storage set.

Format

BACKUP_SET or
BACS
  SET_NAME=name
  STATUS=status variable

Parameters

SET_NAME or **SN**

The name of the set for which permanent files are to be backed up.

STATUS

Returns the completion status for this subcommand.

Remarks Use of this subcommand to back up files other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

DELETE_ALL_FILES Subcommand

Purpose Deletes all permanent files. You can use this subcommand following a BACKUP_ALL_FILES subcommand to reduce the size of the permanent file base. Before executing the DELETE_ALL_FILES subcommand, review the recent backup listing to ensure that the selected files have been backed up.

Format **DELETE_ALL_FILES**
STATUS=status variable

Parameters *STATUS*

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to delete files other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- Before entering the DELETE_ALL_FILES command, enter one or more of the following subcommands to specify which files to delete:

  INCLUDE_CYCLES
  INCLUDE_EMPTY_CATALOGS
  INCLUDE_LARGE_CYCLES
  INCLUDE_MASTER_CATALOGS
  INCLUDE_SMALL_CYCLES
  INCLUDE_USER
  INCLUDE_VOLUME
  EXCLUDE_CATALOG
  EXCLUDE_FILE
  EXCLUDE_HIGHEST_CYCLES

  CAUTION

  If you do not precede the DELETE_ALL_FILES subcommand with one of these subcommands, all files are deleted.

- The DELETE_ALL_FILES subcommand skips file cycles that are attached to a job.
- If the backup and delete operations are performed in a busy system, changes in file access can result in files being deleted that have not previously been backed up.

Examples This example deletes all files not accessed since January 1, 1988:

```
sou/backup_permanent_files backup_file=$null
PUB/include_cycles selection_criteria=accessed ..
PUB../before=1988-1-1
PUB/delete_all_files
PUB/quit
```

DELETE_CATALOG_CONTENTS Subcommand

Purpose Deletes all files in the specified catalog and its subcatalogs. It also deletes subcatalogs if they are specified in a previous INCLUDE_EMPTY_CATALOGS or INCLUDE_MASTER_CATALOGS subcommand.

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

Parameters CATALOG or C

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

STATUS

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to delete files other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- The DELETE_CATALOG_CONTENTS subcommand skips file cycles that are attached to jobs. When all jobs detach these file cycles, the file cycles are deleted.
- Previous EXCLUDE_CATALOG, EXCLUDE_FILE, EXCLUDE_HIGHEST_CYCLES, INCLUDE_CYCLES, INCLUDE_LARGE_CYCLES, INCLUDE_VOLUME, INCLUDE_EMPTY_CATALOGS, and INCLUDE_MASTER_CATALOGS subcommands specify a subset of permanent files to be deleted.

Examples This example deletes the contents of catalog CATALOG_1 for user TKWS87:

```
sou/backup_permanent_files backup_files=$null
PUB/delete_catalog_contents catalog=.tkws87.catalog_1
PUB/quit
```

DELETE_FILE_CONTENTS Subcommand

Purpose Deletes all cycles of a file.

Format

DELETE_FILE_CONTENTS or
DELETE_FILE_CONTENT or
DELFC
  FILE=file
  PASSWORD=name or *keyword*
  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*

Specifies the password for the file to be backed up. The keyword NONE specifies no password. NONE is the default. This parameter is required for all users except those validated for the system administration capability.

STATUS

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to delete files other than your own requires the System Operator Utility to be active with either the SYSTEM_ ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- The DELETE_FILE_CONTENTS subcommand skips file cycles that are attached to jobs. When all jobs detach these file cycles, the file cycles are deleted.
- Previous EXCLUDE_FILE, EXCLUDE_HIGHEST_CYCLES, INCLUDE_VOLUME, INCLUDE_LARGE_CYCLES, and INCLUDE_ CYCLES subcommands specify a subset of permanent file cycles to be deleted.

Examples This example deletes all cycles of permanent file DATA_FILE_1 in the master catalog TKWS87:

```
sou/backup_permanent_files backup_file=$null
PUB/delete_file_contents file=.tkws87.data_file_1
PUB/quit
```

EXCLUDE_CATALOG Subcommand

Purpose Excludes a catalog from subsequent backup or delete operations. The catalog is excluded only if it is a subcatalog of a catalog that is being backed up.

Format **EXCLUDE_CATALOG** or
EXCC
  CATALOG=file
  STATUS=status variable

Parameters CATALOG or C

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

STATUS

Returns the completion status for this subcommand.

Remarks This subcommand takes precedence over all INCLUDE subcommand entries.

Examples This example excludes subcatalog CATALOG_1 from the backup of master catalog TKWS87:

```
sou/backup_permanent_files backup_file=$local.backup
PUB/exclude_catalog catalog=.tkws87.catalog_1
PUB/backup_catalog catalog=.tkws87
PUB/quit
```

EXCLUDE_FILE Subcommand

Purpose Excludes a file or cycle from subsequent backup or 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 or delete operations. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks This subcommand takes precedence over all INCLUDE subcommands.

Examples This example excludes the file CATALOG_1.REGISTER from the backup of master catalog TKWS87:

```
sou/backup_permanent_files backup_file=backup_1023
PUB/exclude_file file=.tkws87.catalog_1.register
PUB/backup_catalog catalog=.tkws87
PUB/quit
```

EXCLUDE_HIGHEST_CYCLES Subcommand

Purpose Excludes a specified number of high file cycles from subsequent backup or delete operations.

Format **EXCLUDE_HIGHEST_CYCLES** or
EXCLUDE_HIGHEST_CYCLE or
EXCHC
NUMBER_OF_CYCLES=integer or *keyword*
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 to 999. The keyword ALL specifies all file cycles. The default is 3.

STATUS

Returns the completion status for this subcommand.

Remarks This subcommand takes precedence over all INCLUDE subcommands.

Examples

- This example excludes the highest cycle of each file in a user's catalog from a subsequent DELETE_CATALOG_CONTENTS command:

```
sou/backup_permanent_files backup_file=$local.backup
PUB/exclude_highest_cycles number_of_cycles=1
PUB/delete_catalog_contents catalog=.tkws87
PUB/quit
```

- This example backs up all catalog information in the system, excluding all file cycle data:

```
sou/backup_permanent_files backup_file=$local.catalog_backup
PUB/exclude_highest_cycles number_of_cycles=all
PUB/backup_all_files
PUB/quit
```

INCLUDE_CYCLES Subcommand

Purpose Selects file cycles for subsequent backup or delete operations based on a file cycle's last access date and time, creation date and time, expiration date and time, or last modification date and time. This subcommand enables a site to perform a partial backup or to delete file cycles that have not been accessed since a specified date and time.

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 criteria by which file cycles for subsequent backup or delete operations are to be selected. This parameter is required. You must specify one of the following keywords:

ACCESSED

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

CREATED

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

EXPIRED

Selects cycles based on the date and time they are to expire.

MODIFIED

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

IGNORE_DATE_TIME or IDT

Cancels any INCLUDE_CYCLES subcommand selection criteria that were specified earlier in the same BACKUP_PERMANENT_FILES utility session.

AFTER or *A*

Specifies the date and time limit after which file cycles are to be selected based on the value of the SELECTION_CRITERIA parameter. The default is 1980-01-01.00:00:00.000. The date_time data type consists of several integer fields in the following format:

year–month–day.hour:minute:second.millisecond

For more information about the date_time data type, refer to the NOS/VE System Usage manual.

BEFORE or *B*

Specifies the date and time limit before which file cycles are to be selected based on the value of the SELECTION_CRITERIA parameter. The default is the current date and time. The date_time data type consists of several integer fields in the following format:

year–month–day.hour:minute:second.millisecond

For more information about the date_time data type, refer to the NOS/VE System Usage manual.

STATUS

Returns the completion status for this subcommand.

Remarks

- You can specify values for both the BEFORE and AFTER parameters to define a time window from which to select file cycles that meet the selection criteria.
- The following subcommands override the effects of this subcommand: EXCLUDE_CATALOG, EXCLUDE_FILE, EXCLUDE_HIGHEST_CYCLES.
- Within a BACKUP_PERMANENT_FILES utility session, only the most recent INCLUDE_CYCLES subcommand entry is applied to subsequent backup operations.

Examples

- This example produces a partial backup composed of all files that were modified after 4:00 p.m. on January 13, 1989:

```
sou/backup_permanent_files backup_file=$local.backup
PUB/include_cycles selection_criteria=modified ..
PUB../after=1989-01-13.16:00
PUB/backup_all_files
PUB/quit
```

- The next example backs up all files that were accessed after 5:00 a.m. January 14, 1989 and before 5:00 a.m. January 30, 1989:

```
sou/backup_permanent_files backup_file=$local.backup
PUB/include_cycles selection_criteria=accessed ..
PUB../after=1989-01-14.5:00 before=1989-01-30.5:00
PUB/backup_all_files
PUB/quit
```

INCLUDE_EMPTY_CATALOGS Subcommand

Purpose Specifies whether subsequent DELETE_ALL_FILES or DELETE_CATALOG_CONTENTS subcommand entries will delete empty catalogs. An empty catalog is one that contains no files or subcatalogs.

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 empty catalogs are to be deleted. The default is TRUE:

TRUE

Delete empty catalogs.

FALSE

Do not delete empty catalogs.

STATUS

Returns the completion status for this subcommand.

Examples

- This example deletes all empty catalogs in subcatalog CATALOG_1 of master catalog TKWS87:

```
sou/backup_permanent_files backup_file=$null
PUB/include_empty_catalogs
PUB/delete_catalog_contents catalog=.tkws87.catalog_1
PUB/quit
```

- This example saves empty catalogs from being deleted for user DLH in family NVE:

```
sou/backup_permanent_files backup_file=$local.backup
PUB/include_empty_catalogs delete_catalog=false
PUB/delete_catalog_contents :nve.dlh
PUB/quit
```

INCLUDE_LARGE_CYCLES Subcommand

Purpose Selects file cycles that exceed a specified number of bytes for subsequent backup or delete operations.

Format

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

Parameters

MINIMUM_SIZE or **MS**

Specifies a minimum number of bytes. File cycles containing an equal or greater number of bytes are included in the backup or delete operation. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks

- If you enter the BACKUP_FILE subcommand and specify a cycle reference on the FILE parameter, any subsequent INCLUDE_LARGE_CYCLES subcommand entry is ignored.
- Enter the INCLUDE_LARGE_CYCLES and INCLUDE_SMALL_CYCLES subcommands during the same BACKUP_PERMANENT_FILES utility session to specify a range of file sizes by which to select file cycles. However, be sure that the value you specify for the MINIMUM_SIZE parameter on the INCLUDE_LARGE_CYCLES subcommand is less than or equal to the value you specify for the MAXIMUM_SIZE parameter on the INCLUDE_SMALL_CYCLES subcommand.
- The following subcommands override the effects of this subcommand: EXCLUDE_CATALOG, EXCLUDE_FILE, EXCLUDE_HIGHEST_CYCLES.
- Within a BACKUP_PERMANENT_FILES utility session, only the most recent INCLUDE_LARGE_CYCLES subcommand entry is applied to subsequent backup operations.

Examples This example backs up and deletes all cycles greater than or equal to 1,000,000 bytes:

```
sou/backup_permanent_files backup_file=$local.backup
PUB/include_large_cycles minimum_size=1000000
PUB/backup_all_files
PUB/delete_all_files
PUB/quit
```

INCLUDE_MASTER_CATALOGS Subcommand

Purpose Specifies whether subsequent DELETE_ALL_FILES or DELETE_CATALOG_CONTENTS subcommand entries will delete empty master catalogs. An empty master catalog is one that contains no files or subcatalogs.

Format INCLUDE_MASTER_CATALOGS or
INCMC
DELETE_MASTER_CATALOG=boolean
STATUS=status variable

Parameters *DELETE_MASTER_CATALOG* or *DELETE_MASTER_CATALOGS* or *DMC*

Specifies whether empty master catalogs should be deleted. The default is TRUE.

TRUE

A subsequent DELETE_CATALOG_CONTENTS or DELETE_ALL_FILES subcommand also deletes the master catalogs for included users.

FALSE

Master catalogs for any included users are not deleted.

STATUS

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to include master catalogs other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- The following subcommands override the effects of this subcommand: EXCLUDE_CATALOG, EXCLUDE_FILE, EXCLUDE_HIGHEST_CYCLES.
- Within a BACKUP_PERMANENT_FILES utility session, only the most recent INCLUDE_MASTER_CATALOGS subcommand entry is applied to subsequent backup operations.

Examples This example deletes from family NVE the empty master catalogs and the master catalogs that contain only empty subcatalogs:

```
sou/backup_permanent_files backup_file=$null
PUB/include_master_catalogs
PUB/include_empty_catalogs
PUB/exclude_highest_cycles number_of_cycles=all
PUB/delete_catalog_contents catalog=:nve
PUB/quit
```

INCLUDE_SMALL_CYCLES Subcommand

Purpose Selects file cycles that contain fewer than a specified number of bytes for subsequent backup or delete operations.

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

Parameters **MAXIMUM_SIZE** or **MS**

Specifies a maximum number of bytes. File cycles containing an equal or smaller number of bytes are included in the backup or delete operation. The keyword MAXIMUM specifies all file cycles regardless of size. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks

- If you enter the BACKUP_FILE subcommand and specify a cycle reference on the FILE parameter, any subsequent INCLUDE_SMALL_CYCLES subcommand entry is ignored.
- Enter the INCLUDE_LARGE_CYCLES and INCLUDE_SMALL_CYCLES subcommands during the same BACKUP_PERMANENT_FILES utility session to specify a range of file sizes by which to select file cycles. However, be sure that the value you specify for the MINIMUM_SIZE parameter on the INCLUDE_LARGE_CYCLES subcommand is less than or equal to the value you specify for the MAXIMUM_SIZE parameter on the INCLUDE_SMALL_CYCLES subcommand.
- The following subcommands override the effects of this subcommand: EXCLUDE_CATALOG, EXCLUDE_FILE, EXCLUDE_HIGHEST_CYCLES.
- Within a BACKUP_PERMANENT_FILES utility session, only the most recent INCLUDE_SMALL_CYCLES subcommand entry is applied to subsequent backup operations.

Examples This example backs up and deletes all file cycles containing 1,000 bytes or less:

```
sou/backup_permanent_files backup_file=$local.backup
PUB/include_small_cycles maximum_size=1000
PUB/backup_all_files
PUB/delete_all_files
PUB/quit
```

INCLUDE_USERS Subcommand

Purpose Selects a set of user master catalogs to be backed up during a subsequent BACKUP_ALL_FILES subcommand.

Format **INCLUDE_USERS** or
INCLUDE_USER or
INCU
  USER=list of catalog or **keyword**
  STATUS=status variable

Parameters **USER** or **U**

Specifies the names of the user master catalogs to be backed up. The keyword ALL specifies all users and overrides any previous INCLUDE_USER subcommand entry. You can specify the list of users as a range. You can specify up to 20 users and ranges of users for this parameter. If you specify a family name, all users in the family are selected. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to back up master catalogs other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

- This subcommand enables you to back up groups of user catalogs to separate tape sets. This subcommand also enables you to perform multiple backups simultaneously when the appropriate tape units and controllers are available.

- User names are processed in the order in which they reside in the catalog, unless the SORT_USERS subcommand has been entered to sort them alphabetically.

- The following subcommands override the effects of this subcommand: EXCLUDE_CATALOG, EXCLUDE_FILE, EXCLUDE_HIGHEST_CYCLES.

- Within a BACKUP_PERMANENT_FILES utility session, only the most recent INCLUDE_USERS subcommand entry is applied to subsequent backup operations.

Examples

- This example directs the backup process to back up files for users AAA to FZZ in family NVE. The files are written to tape in the order those users are cataloged, which may not be alphabetical. That is, user FAB may be backed up before user BAB.

```
sou/backup_permanent_files backup_file=$local.backup
PUB/include_users :nve.aaa..:nve.fzz
PUB/backup_all_files
PUB/quit
```

- This example backs up user catalogs in alphabetical order:

```
sou/backup_permanent_files backup_file=$local.backup
PUB/sort_users
PUB/include_users :nve.aaa..:nve.fzz
PUB/backup_all_files
PUB/quit
```

- This example backs up all users in the family NVE:

```
sou/backup_permanent_files backup_file=$local.backup
PUB/include_users :nve
PUB/backup_all_files
PUB/quit
```

- This example the backs up files beginning with the first user name in family NVE through user name DLH:

```
sou/backup_permanent_files backup_file=$local.backup
PUB/include_users :nve..:nve.dlh
PUB/backup_all_files
PUB/quit
```

INCLUDE_VOLUMES Subcommand

Purpose Selects disk volumes for subsequent backup or delete operations.

Format

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

Parameters

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

Specifies the volumes to be included. This value can be a 1- to 6-character volume serial number or the keyword ALL. This parameter is required.

CYCLE_SELECTION or *CS*

Specifies which cycles on a volume should be backed up. You can specify the following keywords; the default is MULTIPLE_VOLUMES:

INITIAL_VOLUME or 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 or MV

Selects all cycles residing either partially or completely on the volume.

STATUS

Returns the completion status for this subcommand.

Remarks

- The system ignores values for the CYCLE_SELECTION parameter when RECORDED_VSN=ALL.
- When using the INCLUDE_VOLUMES subcommand to backup specific disk volumes to separate sets of full backup tapes, specify CYCLE_SELECTION=INITIAL_VOLUME. This prevents a file cycle that spans two volumes from being backed up twice.
- When using the INCLUDE_VOLUMES subcommand to backup specific disk volumes to safeguard against those disk volumes failing, specify CYCLE_SELECTION=MULTIPLE_VOLUMES. This ensures that the backup includes file cycles that span two volumes.
- If you enter the BACKUP_FILE subcommand and specify a cycle reference on the FILE parameter, any subsequent INCLUDE_VOLUMES subcommand entry is ignored.
- The following subcommands override the effects of this subcommand: EXCLUDE_CATALOG, EXCLUDE_FILE, EXCLUDE_HIGHEST_CYCLES.
- Within a BACKUP_PERMANENT_FILES utility session, only the most recent INCLUDE_VOLUMES subcommand entry is applied to subsequent backup operations.

Examples This example backs up all files for user TKWS87 that reside on the disk volume VOL033, then deletes those files from the volume:

```
sou/backup_permanent_files backup_file=$local.backup
PUB/include_volume recorded_vsn=VOL033 ..
PUB../cycle_selection=multiple_volumes
PUB/backup_catalog catalog=.tkws87
PUB/delete_catalog_contents catalog=.tkws87
PUB/quit
```

QUIT Subcommand

Purpose	Ends a BACKUP_PERMANENT_FILES utility session.
Format	**QUIT** or **QUI**
Parameters	None.

SET_BACKUP_OPTIONS Subcommand

Purpose Specifies actions to be taken by the BACKUP_PERMANENT_FILES utility.

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

Parameters *EXCLUDE_CATALOG_INFORMATION* or *ECI*

Specifies whether to exclude the catalog information from the backup. This allows you to backup file cycles and catalog information separately. The default is FALSE.

TRUE

Excludes catalog information from the backup.

FALSE

Includes catalog information in the backup.

NULL_BACKUP_FILE_OPTION or *NBFO*

Specifies whether to read file data during backups to $NULL or any file assigned to the NULL device class. You must also specify BACKUP_FILE=$NULL on the BACKUP_PERMANENT_FILES command. This parameter has the following keyword value; the default is not to read data when backing up permanent files to $NULL:

READ_DATA or RD

Reads all file data when backing up to $NULL. The backup listing identifies those files containing unreadable data.

INCLUDE_ARCHIVE_INFORMATION or *IAI*

Specifies whether archive information is to be included on the backup. This parameter applies only to sites using Archive/VE. Archive information describes the storage location for file cycle data that has been archived. You cannot retrieve archived file cycle data without this archive information. You can specify one of the following keywords; the default is TRUE for users validated for the system administration capability and FALSE for all others:

TRUE

Includes archive information in the backup for file cycles that have been archived and their data released from mass storage. You must be validated for the system administration capability to enter this value.

FALSE

Excludes archive information from the backup for file cycles that have been archived and their data released from mass storage.

INCLUDE_DATA or *ID*

Specifies the types of file cycle data to be included in the backup according to their archive status and storage location (disk or archive medium). This parameter applies especially to sites using Archive/VE. You can specify one or more of the following keyword values; the default is the list (UNRELEASABLE_DATA, RELEASABLE_DATA):

UNRELEASABLE_DATA or UD

Includes data for file cycles that have never been archived and for file cycles that have been modified since they were last archived.

RELEASABLE_DATA or RD

Includes data for file cycles that have not been modified since they were last archived.

OFFLINE_DATA or OD

Includes file cycle data that has been released from mass storage and currently resides on some archive medium.

ALL

Includes data for all file cycles regardless of whether they have been archived or their data has been released from mass storage.

STATUS

Returns the completion status for this subcommand.

Remarks

- When performing separate backups for different groups of volumes on a system, you must specify EXCLUDE_CATALOG_INFORMATION=FALSE for at least one of the backups. Otherwise, no catalog information will reside on the backup tapes and restoring files will be impossible.
- If you specify INCLUDE_DATA=OFFLINE_DATA, the system retrieves the archived file cycle data. After the data has been retrieved, the backup continues. The mass storage space is released after each file is backed up.
- The EXCLUDE_CATALOG_INFORMATION parameter overrides the effects of the INCLUDE_ARCHIVE_INFORMATION and INCLUDE_DATA parameters.
- Within a BACKUP_PERMANENT_FILES utility session, only the most recent SET_BACKUP_OPTIONS subcommand entry is applied to subsequent backup operations. Options remain in effect until you change them or until you end the utility session.

Examples

This example backs up file cycles only for master catalog TKWS87. Catalog information is excluded from the backup.

```
sou/backup_permanent_files backup_file=$local.backup
PUB/set_backup_options exclude_catalog_information=true
PUB/backup_catalog catalog=.tkws87
PUB/quit
```

SET_LIST_OPTIONS Subcommand

Purpose Specifies the types of information to be written to the backup listing file by subsequent backup or delete operations.

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 be recorded in the backup listing. You can specify the following keywords; the default is NONE:

ACCOUNT or A

Records the account name.

PROJECT or 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 information to record in the backup listing for each cycle. The cycle number and whether the cycle was excluded is also recorded. You can specify the following keywords; the default is (MODIFICATION_DATE_TIME, SIZE):

CREATION_DATE_TIME or CDT

Records the date and time the cycle was created.

ACCESS_DATE_TIME or ADT

Records the date and time the cycle was last accessed.

MODIFICATION_DATE_TIME or MDT

Records the date and time the cycle was last modified.

EXPIRATION_DATE or ED

Records the expiration date of the cycle.

ACCESS_COUNT or AC

Records the number of accesses to the cycle.

SIZE or S

Records the size of the cycle in bytes.

RECORDED_VSN or RVSN

Records all disk volumes on which the cycle resides.

GLOBAL_FILE_NAME or GFN

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

ALTERNATE_FILE_MEDIA_DESCRIPTOR or AFMD

Records the following information about each archived file cycle for sites using Archive/VE:

- Type of archive storage medium.
- Date and time the file cycle was last modified.
- Size of the file 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 backup listing file. The default is TRUE.

TRUE

Writes all excluded catalogs, files, and cycles in the backup listing.

FALSE

Does not write excluded catalogs, files, and cycles in the backup listing.

STATUS

Returns the completion status for this subcommand.

Remarks Within a BACKUP_PERMANENT_FILES utility session, only the most recent SET_LIST_OPTIONS subcommand entry is applied to subsequent backup operations.

SORT_USERS Subcommand

Purpose Specifies whether master catalogs are to be backed up in alphabetical order within families. The resulting backup listing makes it easier to locate a file or catalog for a particular user name.

Format **SORT_USERS** or
SORU
ALPHABETICAL_ORDER=boolean
STATUS=status variable

Parameters *ALPHABETICAL_ORDER* or *AO*

Specifies whether permanent files are to be backed up in alphabetical order by user name. The default is TRUE.

TRUE

Causes master catalogs to be backed up in alphabetical order.

FALSE

Causes the master catalogs to be backed up in the order they were created.

STATUS

Returns the completion status for this subcommand.

Remarks Use of this subcommand requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

Examples

- This example backs up a subset of master catalogs in family NVE. The master catalogs JOHNSON to TLH are backed up in alphabetical order.

```
sou/backup_permanent_files backup_file=$local.backup
PUB/sort_users alphabetical_order=true
PUB/include_users user=:nve.johnson..:nve.tlh
PUB/backup_all_files
PUB/quit
```

- This example backs up all master catalogs in family NVE that begin with the letter A. The master catalogs are backed up in alphabetical order.

```
sou/backup_permanent_files backup_file=$system_backup
PUB/sort_users alphabetical_order=true
PUB/include_users user=:nve.a..:nve.azzzzzz
PUB/backup_all_files
PUB/quit
```

- This example shows how the sorting process handles numbers. Suppose family NVE consists of master catalogs U1, U11, U12, U3, and U4. The following commands would only back up U1, U11, and U12:

```
sou/backup_permanent_files backup_file=$local.backup
PUB/sort_users alphabetical_order=true
PUB/include_users user=:nve.u1..:nve.u20
PUB/backup_all_files
PUB/quit
```

RESTORE_PERMANENT_FILES Utility

This section provides reference information for the RESTORE_PERMANENT_FILES utility and its subcommands and functions. To restore files for users besides yourself, you need to execute the RESTORE_PERMANENT_FILES utility within a System Operator Utility session and have either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION capability. To initiate the System Operator Utility session, enter:

```
/system_operator_utility
```

The system responds with the prompt: sou/. For more information on the System Operator Utility, see the NOS/VE Operations manual.

The RESTORE_PERMANENT_FILES utility command and subcommands are as follows:

- RESTORE_PERMANENT_FILES command
  - $BACKUP_FILE function
  - CHANGE_ALL_PERMITS subcommand
  - DISPLAY_BACKUP_FILE subcommand
  - INCLUDE_CYCLES subcommand
  - INCLUDE_VOLUMES subcommand
  - QUIT subcommand
  - RESTORE_ALL_FILES subcommand
  - RESTORE_CATALOG subcommand
  - RESTORE_EXCLUDED_FILE_CYCLES subcommand
  - RESTORE_EXISTING_CATALOG subcommand
  - RESTORE_EXISTING_FILE subcommand
  - RESTORE_FILE subcommand
  - RESTORE_MISSING_CATALOGS subcommand
  - SET_LIST_OPTIONS subcommand
  - SET_RESTORE_MISSING_CATALOGS subcommand
  - SET_RESTORE_OPTIONS subcommand

RESTORE_PERMANENT_FILES Command

Purpose

Initiates the utility that restores permanent files and catalogs from backup copies created by the BACKUP_PERMANENT_FILES utility. The restore operations are directed by RESTORE_PERMANENT_FILES utility subcommands.

Format

RESTORE_PERMANENT_FILES or
RESTORE_PERMANENT_FILE or
RESPF
   LIST=file
   STATUS=status variable

Parameters

LIST or *L*

Specifies the name of the file to receive summary information about the restore. This file is called the restore listing. You can also specify how the file is to be positioned prior to use. The default is $LIST.

STATUS

Returns the completion status for the utility.

Remarks

- Use of this command to restore files other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- After entering the RESTORE_PERMANENT_FILES command, the following prompt appears:

```
PUR/
```

- You can specify the content of the restore listing file using the SET_LIST_OPTION subcommand.

Examples

This example initiates a RESTORE_PERMANENT_FILE utility session. The restore listing is written to file $SYSTEM.RESTORE_LISTING.

```
sou/restore_permanent_files list=$system.restore_listing
```

$BACKUP_FILE Function

Purpose

Returns a string containing information on a backup file produced by the BACKUP_PERMANENT_FILES 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_FILES utility.

Format

$BACKUP_FILE or
$BF
  (file
  keyword)

Parameters

file

Specifies the name of the backup file to be queried. This parameter is required.

keyword

Specifies the particular attribute that is being queried. You can specify one of the following keywords; the default is IDENTIFIER:

IDENTIFIER or I

Returns a string containing the path name of the first name on the backup file.

IDENTIFIER_TYPE or 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.

Remarks

- This function is especially useful when you attempt to restore files from a backup file with a known path name, but an unknown file or catalog name.
- 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 subcommand.

Examples

This example restores the catalog TKWS87.MY_CATALOG. Assume that you are restoring files using a backup tape file produced by the BACKUP_CATALOG subcommand.

```
sou/restore_permanent_files list=$local.restore_listing
PUR/restore_catalog $fname($backup_file($local.backup identifier)) ..
PUR../backup_file=$local.backup new_catalog_name=.tkws87.my_catalog
PUR/quit
```

CHANGE_ALL_PERMITS

Purpose Changes the access permission for all files and catalogs in a family to reflect the new family name. Use this command when changing family names using the CHANGE_FAMILY command or when restoring an entire family to a new family catalog.

Format

CHANGE_ALL_PERMITS or
CHAAP
  FAMILY_NAME=name
  NEW_FAMILY_NAME=name
  STATUS=status variable

Parameters

FAMILY_NAME or FN

Specifies the name of the family for which file permits are to be changed. This parameter is required.

NEW_FAMILY_NAME or NFN

Specifies the new family name for all file permits. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand for files other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- The values for FAMILY_NAME and NEW_FAMILY_NAME must be different.
- Refer to Changing Family Names earlier in this chapter for an example of the use of this subcommand.

DISPLAY_BACKUP_FILE Subcommand

Purpose Displays information about the files recorded on the backup file.

Format **DISPLAY_BACKUP_FILE** or
DISBF
  BACKUP_FILE=file
  DISPLAY_OPTION=keyword
  NUMBER=integer or *keyword*
  STATUS=status variable

Parameters **BACKUP_FILE** or **BF**

Specifies the file that contains the backup of the files and catalogs previously backed up by a BACKUP_PERMANENT_FILES utility session.

DISPLAY_OPTION or *DO*

Specifies the level of information to be displayed. You can specify one of the following keywords; the default is IDENTIFIER:

IDENTIFIER or I

Displays the name and type (file or catalog) of each entry on the backup file.

DESCRIPTOR or 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 or 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 the data is read without error. No attempt is made to verify the data against the original files and catalogs.

NUMBER or *N*

Selects the number of catalogs, files, or cycles, beginning with the first item in the backup file, for which information is to be displayed. The keyword ALL specifies all catalogs, files, and cycles in the backup file. The default is ALL.

STATUS

Returns the completion status for this subcommand.

INCLUDE_CYCLES Subcommand

Purpose Selects file cycles for subsequent restore operations based on a file cycle's last access date and time, creation date and time, expiration date and time, or last modification date and time as defined on the backup file.

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 criteria by which file cycles are to be selected for restoration. This parameter is required. You must specify one of the following keywords:

ACCESSED

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

CREATED

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

EXPIRED

Selects cycles based on the date and time they are to expire.

MODIFIED

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

IGNORE_DATE_TIME or IDT

Cancels any INCLUDE_CYCLES subcommand selection criteria that were specified earlier in the same RESTORE_PERMANENT_FILES utility session.

AFTER or *A*

Specifies the date and time limit after which file cycles are to be selected based on the value of the SELECTION_CRITERIA parameter. The default is 1980-01-01.00:00:00.000. The date_time data type consists of several integer fields in the following format:

year–month–day.hour:minute:second.millisecond

For more information about the date_time data type, refer to the NOS/VE System Usage manual.

BEFORE or *B*

Specifies the date and time limit before which to select file cycles are to be selected based on the value of the SELECTION_CRITERIA parameter. The default is the current date and time. The date_time data type consists of several integer fields in the following format:

year–month–day.hour:minute:second.millisecond

For more information about the date_time data type, refer to the NOS/VE System Usage manual.

STATUS

Returns the completion status for this subcommand.

Remarks

- You can specify values for both the BEFORE and AFTER parameters to define a time window from which file cycles are to be selected that meet the selection criteria. In addition, these two parameters enable you to select from among several versions of the same file cycle on the backup file.
- Within a RESTORE_PERMANENT_FILES utility session, only the most recent INCLUDE_CYCLES subcommand entry is used with the subsequent restoring of files.

INCLUDE_VOLUMES Subcommand

Purpose Selects the disk volumes on which files are to be restored by a subsequent RESTORE_EXCLUDED_FILE_CYCLES subcommand entry.

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

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

Specifies the recorded volume serial numbers of the disk volumes for which files are to be restored. The keyword ALL selects all volumes represented on the backup file. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks Within a RESTORE_PERMANENT_FILES utility session, only the most recent INCLUDE_VOLUMES subcommand entry is used with the subsequent restoring of files.

QUIT Subcommand

Purpose	Ends a RESTORE_PERMANENT_FILES utility session.
Format	**QUIT** or **QUI**
Parameters	None.

RESTORE_ALL_FILES Subcommand

Purpose Restores all catalogs and all permanent files from the backup file. This subcommand only restores files and catalogs that are not currently registered in an existing catalog.

Format **RESTORE_ALL_FILES** or
RESAF
  BACKUP_FILE=file
  STATUS=status variable

Parameters **BACKUP_FILE** or **BF**

Specifies the file containing the backup copies of the files and catalogs to be restored. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to restore files other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- Catalogs and files that do not already exist in the permanent file system are restored. Catalogs and files that already exist are not altered.
- When restoring all files after a catalog volume has been lost, you must first restore catalogs from the most recent catalog backup. Refer to Restoring Catalogs in chapter 13, Repair Solutions.
- This subcommand does not restore file cycles that have been archived and released from mass storage using Archive/VE.
- To restore permanent files from partial backups, use the RESTORE_ALL_FILES subcommand 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 in the last partial backup is restored using the RESTORE_EXCLUDED_FILE_CYCLES subcommand.

Examples This example restores all files in the system that were previously backed up with a BACKUP_ALL_FILES subcommand:

```
sou/restore_permanent_files
PUR/restore_all_files backup_file=$local.backup
PUR/quit
```

RESTORE_CATALOG Subcommand

Purpose Restores a catalog that does not currently exist on the system. This subcommand only restores files and catalogs that are not currently registered in an existing 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 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. The default is the name as it exists on the backup file.

STATUS

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to restore a catalog other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- This command cannot be used to restore a master catalog.
- The catalog being restored must not currently exist.
- This subcommand does not restore file cycles that have been archived and released from mass storage using Archive/VE.

Examples This example restores the master catalog TKWS87 to a new subcatalog in the master catalog:

```
sou/restore_permanent_files list=$local.restore_listing
PUR/restore_catalog catalog=.tkws87 ..
PUR../new_catalog_name=.tkws87.catalog_2 ..
PUR../backup_file=$local.backup
PUR/quit
```

RESTORE_EXCLUDED_FILE_CYCLES Subcommand

Purpose

Restores file cycles that exist in the permanent file system but do not currently 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*

Specifies file for which file cycles are to be restored. If you do not specify a cycle number, all of the file's cycles are restored. A cycle reference must be an integer, not $HIGH or $LOW.

CATALOG or *C*

Specifies the catalog for which file cycles are to be restored. All file cycles in the catalog are restored.

BACKUP_FILE or **BF**

Specifies the file containing the backup information. This parameter is required.

NEW_NAME or *NN* or *NEW_CATALOG_NAME* or *NCN* or *NEW_FILE_NAME* or *NFN*

Specifies a new name for the catalog, file, or cycle into 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. The default is 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*

Specifies the circumstances under which excluded file cycles are restored. The default is (MEDIA_MISSING,NO_DATA_DEFINED). You can specify one of the the following keywords:

MEDIA_MISSING or MM

Restores file cycle data for a disk unit that was off, down, or not configured during the previous deadstart.

NO_DATA_DEFINED or NDD

Restores data for file cycles that are defined in a catalog but have no data.

VOLUME_UNAVAILABLE or VU

Restores file cycle data for a disk unit that has gone down since the deadstart.

STATUS

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to restore files other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- This subcommand restores file cycles that do not already exist on the system.
- 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 file cycle has data defined for it, the file cycle is not altered.
- You may specify values for FILE or CATALOG, but not both. If you omit the FILE and CATALOG parameters, all file cycles are restored.
- If the system fails due to a permanent file device failure, you may reload the lost cycles with the RESTORE_EXCLUDED_FILE_CYCLES subcommand using just the backup tapes containing the cycles of the failed device.

Examples

This example restores files from a previous partial backup and a previous full backup:

```
sou/restore_permanent_files
PUR/restore_all_files backup_file=$local.partial
PUR/restore_excluded_file_cycles backup_file=$local.full
PUR/quit
```

RESTORE_EXISTING_CATALOG Subcommand

Purpose Restores permanent files for a catalog that exists in the system.

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 to be restored from the backup file. This parameter is required.

BACKUP_FILE or **BF**

Specifies the file containing the backup copy of the catalog and its 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 in the backup file are restored. The default is the name as it exists on the backup file.

STATUS

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to restore files other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.

- This subcommand restores only subcatalogs and file cycles that are present in the backup file, but do not exist in the specified catalog.

Examples This example restores the master catalog TKWS87 that was backed up with the BACKUP_CATALOG subcommand:

```
sou/restore_permanent_files list=$local.restore_listing
PUR/restore_existing_catalog catalog=.tkws87 ..
PUR../backup_file=$local.backup
PUR/quit
```

RESTORE_EXISTING_FILE Subcommand

Purpose Restores file cycles for a file that exists on the system.

Format **RESTORE_EXISTING_FILE** or
RESEF
  FILE=file
  BACKUP_FILE=file
  PASSWORD=name or *keyword*
  NEW_FILE_NAME=file
  STATUS=status variable

Parameters FILE or F

Specifies the file to be restored from the backup file. File cycle references are 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*

Specifies the password for the file to be restored. The keyword NONE specifies no password. NONE is the default. When entering this subcommand as the $SYSTEM user, this parameter is not needed.

NEW_FILE_NAME or *NFN*

Specifies the existing file to be restored. The default is the name as it exists in the backup file

STATUS

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to restore files other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- This subcommand restores those file cycles that exist in the backup file but do not exist in the file specified by the FILE parameter. File information, such as file permits, accounting information, the file password, and the file access log, is not restored.
- Cycles that currently exist as permanent files are not altered.

Examples This example restores cycle number 4 of file DATA_FILE_0 in master catalog TKWS87:

```
sou/restore_permanent_files
PUR/restore_existing_file file=:tkws87.data_file_0.4 ..
PUR../backup_file=$local.backup
PUR/quit
```

RESTORE_FILE Subcommand

Purpose Restores a file that does not exist on the system.

Format

RESTORE_FILE or
RESF
  FILE=file
  BACKUP_FILE=file
  PASSWORD=name or *keyword*
  NEW_FILE_NAME=file
  STATUS=status variable

Parameters

FILE or **F**

Specifies the file to be restored from the backup file. If you specify a cycle reference, only that file cycle is restored. If a cycle reference is omitted, all file cycles are restored. 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*

Specifies the password for the file to be restored. The keyword NONE specifies no password. NONE is the default. This parameter is required for all users except those validated for the system administration capability.

NEW_FILE_NAME or *NFN*

Specifies a new name for the file being restored. The default is the name as it exists in the backup file.

STATUS

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to restore files other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- If you specify a single file cycle, this subcommand creates the file and restores the file cycle. The file information, such as file permits, the file access log, accounting information, and the file password, is not restored.
- If you include a cycle reference on the FILE parameter, it must be a specific cycle number; you cannot use the $HIGH and $LOW keywords.
- 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.

Examples This example restores cycle number 87 of file DATA_FILE_0 in master catalog TKWS87. The file is restored as cycle number 1 of file DATA_FILE_2.

```
sou/restore_permanent_files
PUR/restore_file file=.tksw87.data_file_0.87 ..
PUR../backup_file=$local.backup new_file_name=.tkws87.data_file_2
PUR/quit
```

RESTORE_MISSING_CATALOGS

Purpose Restores a catalog that resides on disk volumes that are missing or unavailable. A missing volume is one that was down or off during the previous deadstart. An unavailable disk unit is one that has gone down since the deadstart.

Format

RESTORE_MISSING_CATALOGS or
RESTORE_MISSING_CATALOG or
RESMC
  BACKUP_FILE=file
  CATALOG=name
  RESTORE_EXCLUDED_FILE_CYCLES=list of keyword
  STATUS=status variable

Parameters

BACKUP_FILE or **BF**

Specifies the name of the backup file. This parameter is required.

CATALOG or *C*

Specifies the name of the catalog you want to restore. The default is all catalogs in the backup file.

RESTORE_EXCLUDED_FILE_CYCLES or *REFC*

Specifies the conditions under which file cycle data is to be restored to the available disk volumes in the system. You can specify one or more of the following keywords; the default is the list (MEDIA_MISSING, NO_DATA_DEFINED):

NONE or N

Restores only catalog information from the backup file.

MEDIA_MISSING or MM

Restores file cycle data for missing disk units.

NO_DATA_DEFINED or NDD

Restores data for file cycles that are defined in a catalog but have no data.

VOLUME_UNAVAILABLE or VU

Restores file cycle data for unavailable disk units.

STATUS

Returns the completion status for this subcommand.

Remarks

- Use of this subcommand to restore catalogs other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- You must enter the SET_RESTORE_MISSING_CATALOGS subcommand immediately before and immediately after the RESTORE_MISSING_CATALOGS subcommand.

- If you restore file cycles using this subcommand, each file cycle catalog entry is marked with a damage condition: PARENT_CATALOG_RESTORED. Use the CHANGE_CATALOG_CONTENTS command to remove this damage condition from catalog entries. The CHANGE_CATALOG_CONTENTS is described in the NOS/VE Operations manual.

- To restore catalogs that reside on disk volumes that were activated after the last deadstart, you must perform a continuation deadstart. For more information about restoring catalogs, refer to chapter 13, Repair Solutions.

SET_LIST_OPTIONS Subcommand

Purpose Specifies the information that is written to the restore listing file by subsequent subcommands. The restore listing file contains information about the catalogs, files, and file cycles that have been restored.

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 be recorded in the restore listing file. You can specify the following keywords; the default is NONE:

ACCOUNT or A

Records the account name.

PROJECT or 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 be recorded in the restore listing for each cycle. The cycle number and whether the cycle was excluded is also recorded. You can specify the following keywords; the default is (MODIFICATION_DATE_TIME, SIZE):

CREATION_DATE_TIME or CDT

Records the date and time the cycle was created.

ACCESS_DATE_TIME or ADT

Records the date and time the cycle was last accessed.

MODIFICATION_DATE_TIME or MDT

Records the date and time the cycle was last modified.

EXPIRATION_DATE or ED

Records the expiration date of the cycle.

ACCESS_COUNT or AC

Records the number of accesses to the cycle.

SIZE or S

Records the size of the cycle in bytes.

RECORDED_VSN or RVSN

Records the disk volumes on which the cycle resides.

GLOBAL_FILE_NAME or GFN

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

ALTERNATE_FILE_MEDIA_DESCRIPTOR or AFMD

Records the following information about each file cycle that has been archived using Archive/VE:

- Type of archive storage medium.
- Date and time the file cycle was last modified.
- Size of the file 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 restore listing file. The default is TRUE.

TRUE

Writes all excluded catalogs, files, and cycles in the restore listing file.

FALSE

Does not write excluded catalogs, files, and cycles in the restore listing file.

STATUS

Returns the completion status for this subcommand.

SET_RESTORE_MISSING_CATALOGS

Purpose Establishes the beginning and the end of the restoration of missing catalogs performed by the RESTORE_MISSING_CATALOGS subcommand.

Format **SET_RESTORE_MISSING_CATALOGS** or
SETRMC
  OPERATION=keyword
  STATUS=status variable

Parameters **OPERATION or O**

Specifies the start or the end of the restoration of missing catalogs. This parameter is required. You can enter one of the following keywords:

START

Establishes the beginning of the missing catalog restoration process.

END

Establishes the end of the missing catalog restoration process.

STATUS

Returns the completion status of this subcommand.

Remarks

- Use of this subcommand to restore catalogs other than your own requires the System Operator Utility to be active with either the SYSTEM_ADMINISTRATION or FAMILY_ADMINISTRATION validation capability in effect. For more information about the System Operator Utility, refer to the NOS/VE Operations manual.
- Enter the following subcommand before entering the RESTORE_MISSING_CATALOGS subcommand:

```
PUR/set_restore_missing_catalogs operation=start
```

- Enter the following subcommand after entering the RESTORE_MISSING_CATALOGS subcommand and after the last backup tape has been unloaded:

```
PUR/set_restore_missing_catalogs operation=end
```

SET_RESTORE_OPTIONS Subcommand

Purpose Specifies a set of options for the disk volumes to be used in restore operations.

Format **SET_RESTORE_OPTIONS** or
SET_RESTORE_OPTION or
SETRO
REQUIRE_MATCHING_MODIFICATION=boolean or *keyword*
ALLOCATION_SIZE=integer or *keyword*
FILE_CLASS=name or *keyword*
INITIAL_VOLUME=name or *keyword*
UPDATE_CYCLE_STATISTICS=boolean or *keyword*
VOLUME_OVERFLOW_ALLOWED=boolean or *keyword*
RESTORE_ARCHIVE_INFORMATION=boolean or *keyword*
STATUS=status variable

Parameters *REQUIRE_MATCHING_MODIFICATION* or *RMM*

Specifies whether 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. You can specify the following keywords; the default is TRUE:

TRUE

Restores a file cycle only if the last modification date and time in the catalog matches the one in the backup file.

FALSE

Restores a file cycle regardless of the date and time of the last modification.

UNSPECIFIED

Retains the value when this parameter was last specified.

ALLOCATION_SIZE or *AS*

Specifies the amount of contiguous disk space, in bytes, to be allocated to the restored files each time additional space is needed. The system uses the value of this parameter as a guide in selecting the amount of space to allocate. Mass storage space actually allocated may vary slightly from the value specified. Values for this parameter can range from 16,384 to 16,777,215. By default, the system chooses the allocation size. The keyword UNSPECIFIED specifies the value when this parameter was last entered.

If the value for this parameter exceeds the value of the MAXIMUM_ALLOCATION_SIZE system attribute, the latter value is used. For more information about this system attribute, see the NOS/VE System Performance and Maintenance manual, Volume 1. Refer to the Remarks section for a description of the restrictions on the use of this parameter.

FILE_CLASS or *FC*

Specifies the mass storage class of the disk volume to which to restore files. The value can be any alphabetic character associated with a mass storage class or one of the following keywords; the default is USER_PERMANENT_FILE:

PRODUCT or M

The file class that provides users with access to products. This keyword is associated with class M mass storage devices.

SERVICE_CRITICAL_PRODUCT or K

The file class that provides all users with access to the system. Examples of this class are files that are attached by the system prolog, the terminal definition libraries, and command libraries. This keyword is associated with class K mass storage devices.

SYSTEM_CRITICAL_FILE or Q

The file class that provides continuous system operation without an interrupt. Only users validated for the system administration capability can specify this keyword. This keyword is associated with mass storage class Q.

SYSTEM_PERMANENT_FILE or K

The file class for files in the $SYSTEM family. Refers to a $SYSTEM permanent file. This type of file is associated with mass storage class K.

USER_PERMANENT_FILE or M

The file class for user permanent files. This type of file is associated with mass storage class M.

UNSPECIFIED

Retains the value when this parameter was last specified.

Only users validated for the system administration capability can specify classes C, J, K, L and Q either directly or indirectly using the keyword values for the parameter. A task executing in rings 6 to 4 can specify the a volume belonging to classes U, V, W, X, Y, and Z for a permanent file. A task executing in rings 13 to 7 can only specify class M for a permanent file.

If you specify neither the FILE_CLASS nor the INITIAL_VOLUME parameter, NOS/VE places the file on a volume that belongs to the class that is appropriate for the file and the job in which the file is created. For a description of the default NOS/VE file assignments, refer to chapter 3, Logical Configuration Utility.

INITIAL_VOLUME or *IV*

Specifies the 6-character volume serial number (VSN) of the disk volume to which to restore files. You can specify a volume serial number or one of the following keywords; the default is UNSPECIFIED:

SYSTEM_DEVICE

Specifies the system device volume. This ensures that the restored files are available whenever the system is available. If the restored files are accessed frequently, consider putting them elsewhere for performance reasons.

UNSPECIFIED

Retains the value when this parameter was last specified.

If the disk volume you specify is not a member of the file class specified in the FILE_CLASS parameter, the volume is full, or the volume does not exist in the active configuration, this subcommand aborts. If you omit the FILE_CLASS parameter and you specify the INITIAL_VOLUME parameter, the following conditions apply:

- A user validated for the system administration capability can restore files to any volume in the configuration, regardless of the execution ring.
- A task executing in rings 6 to 4 can specify a volume belonging to the classes U, V, W, X, Y, and Z for a permanent file.
- A task executing in rings 13 to 7 may only place permanent files on a volume that belongs to class M.

The default is a volume belonging to the mass storage class given by the FILE_CLASS parameter.

UPDATE_CYCLE_STATISTICS or *UCS*

Specifies a boolean value indicating whether the modification date for each restored file is to be the date on which the file was restored, or the modification date recorded on the backup file. You can specify one of the following keywords; the default is FALSE:

TRUE

Specifies the modification date to be the date on which the files were restored. This value ensures that the files are included in any subsequent partial backups.

FALSE

Specifies the modification date to be the modification date recorded on the backup file.

UNSPECIFIED

Retains the value when this parameter was last specified.

VOLUME_OVERFLOW_ALLOWED or *VOA*

Specifies a boolean value indicating whether a restored file can be assigned to more than one volume. The default is TRUE.

TRUE

Permits a restored file to span any volume subject to validation and file class restraints.

FALSE

Confines a restored file to the initial volume to which it is assigned. You can specify this value only if you are validated for the system administration capability.

UNSPECIFIED

Retains the value when this parameter was last specified.

RESTORE_ARCHIVE_INFORMATION or *RAI*

Specifies a boolean value indicating whether archive information is to be restored from the backup file. Archive information describes the storage location of file cycle data that has been archived using Archive/VE. You cannot retrieve archived file cycle data without this archive information. You can specify one of the following keywords; the default is UNSPECIFIED:

TRUE

Restores archive information from the backup file for file cycles that have been archived.

NOTE

You must also specify UPDATE_CYCLE_STATISTICS=FALSE to restore archive information.

FALSE

Does not restore archive information from the backup file for file cycles that have been archived.

UNSPECIFIED

Specifies TRUE for users validated for the system administration capability and FALSE for all others.

STATUS

Returns the completion status for this subcommand.

Remarks

- The SET_RESTORE_OPTIONS subcommand provides the following ways to select the initial volume to which files are restored; these ways are based on the presence or absence of the INITIAL_VOLUME and FILE_CLASS parameters:
  - If you specify only the INITIAL_VOLUME parameter, the volume it identifies is selected.
  - If you specify only the FILE_CLASS parameter, files are restored to a member of the designated class.
  - If you specify both the INITIAL_VOLUME and FILE_CLASS parameters, the specific volume is selected only if the disk volume is a member of the class specified.
  - If you specify neither the INITIAL_VOLUME nor the FILE_CLASS parameter, the system selects a candidate volume with the most available space, regardless of its mass storage class.
- Using allocation sizes larger than the default value (16,384 bytes) has the following disadvantages:
  - It may take longer for the system to allocate space when a file is restored. If a device contains files with different allocation sizes, it becomes harder to locate enough contiguous space to satisfy the file's requirement as the devices fill up.
  - It wastes capacity because the last allocation unit is only partially used (unless the the restored file size is an integral number of allocation units).
  - For security reasons, NOS/VE initializes each allocation unit of a file when at least one page is first written to that allocation unit. This prevents users from reading information that they did not write. The larger the allocation size you choose, the more disk revolutions are required to write the last page of a sequential file.
- To use the allocation size capability effectively, follow these guidelines:
  - Reserve some disks exclusively for large files. For example, remove the devices from membership in classes C, J, K, L, M, N, and Q. Then, assign large files to these volumes using the INITIAL_VOLUME and ALLOCATION_SIZE parameters. If you back up permanent files by volume, remember to include these volumes in your backup.
  - Use this capability for long files that are read sequentially and frequently. NOS/VE reads the entire allocation unit of a sequential file when the first page fault occurs within the allocation unit.

- Specifying no volume overflow is typically used in conjunction with requesting a specific disk volume. This is generally done for fault tolerance reasons. For example, specify no volume overflow if you want all files required for a particular application to reside exclusively on the same volume (so that if one volume fails, you know immediately whether the application is usable). However, assigning files to the same volume may have a negative impact on the application's performance.

- The parameters for this subcommand are functionally identical to corresponding parameters for the REQUEST_MASS_STORAGE command described in appendix D, Assigning Files to a Specific Device.

Part III: Debug and Dump Utilities

The System Core Debugger 9

The System Core Debugger 9

The system core debugger is primarily used by Control Data systems development analysts in debugging the NOS/VE operating system. Services provided by the debugger are task-oriented; selection of the tasks to be debugged must be made via debugger subcommands. No task is under control of the debugger unless it is selected. The selection capability allows any number of tasks to be debugged simultaneously, from one task to all tasks in the system. The system core debugger uses the debug hardware to provide these capabilities.

WARNING

The system core debugger assists Control Data NOS/VE development analysts. Although documented in this chapter, it is not intended for casual use by external analysts. Because of its power, the system core debugger can damage system and user programs and files. Do not experiment with subcommands that you do not fully understand.

Running the system core debugger in the job monitor task can slow or stop the entire job. This can affect the entire system if the job monitor task belongs to the system job, or if you bring the debugger up on a critical system task. Activated in a child task, the system core debugger does not affect the execution of the job monitor task.

Standard NOS/VE command alias conventions do not apply to all system core debugger commands but are used wherever possible. All command parameters are specified positionally.

Online help is available for the system core debugger. Refer to the HELP command description later in this chapter.

Use of the System Core Debugger

The system core debugger was created primarily for use by NOS/VE development analysts, but there are some features of the debugger that are useful to operators and site analysts. The commands listed below are those that an operator should be familiar with. The remainder of this chapter provides more detailed command descriptions.

DUMPJOB

Dumps important information about a hung job when it is not currently active in the system core debugger.

AUTO

Dumps important information about a hung job when it is already active in the system core debugger.

RUN

Causes the system core debugger to be terminated and control to be returned to the job.

The commands used by an operator, together with the ones listed below, are those that a site analyst should be familiar with.

SYSDEBUG

Causes execution of the system core debugger within the specified job.

DISPLAY_TASK_ENVIRONMENT

Displays task names, XCB addresses, and the global task IDs of the tasks present within the job selected by SYSDEBUG. The information provided by this command is used as input for TDEBUG.

TDEBUG

Causes execution of the system core debugger within the specified task.

DISPLAY_TRACE_BACK or DISPLAY_CALLS

Provides information relevant to stack frames associated with an interrupted procedure and its predecessor procedures.

Execution of the System Core Debugger

The system core debugger is activated by conditions that appear in the user condition register (UCR). Two of the conditions in the UCR that cause this activation and communication between hardware and software are the free flag bit and the debug bit.

You can initiate execution of the system core debugger by issuing one of the following commands from the critical display window input line:

SYSDEBUG job all_tasks

This command initiates the system core debugger in the job monitor task of the specified job. For the job parameter, you may enter either the active job list ordinal or the system-supplied name of the job. To determine the active job list ordinal of the job to be debugged, examine the VEDISPLAY ACTIVE_JOBS (VED AJ) display. This command opens a window for the system core debugger. For the all_ tasks parameter, you may specify either ALL, which brings up the debugger on all tasks in the job, or $JOBMNTR, which executes the debugger within the job monitor task of the specified job. The default value is $JOBMNTR.

NOTE

For the SYSDEBUG and DUMPJOB commands, specifying the system-supplied name of the job you want to debug is generally more reliable than specifying the active job list ordinal. This is because a job's active job list ordinal may change as it swaps in and out of memory.

Also, you can invoke the system core debugger on a job that is currently swapped out of memory by specifying its system-supplied name. In order for the debugger to operate on a swapped job, the job cannot currently be forced out of memory by an operator SWAPOUT command, and the MAXIMUM_ACTIVE_JOBS limit for the job's service class must be greater than zero.

TDEBUG task

This command initiates the system core debugger in the task specified. The task parameter is the global task ID of the task in which the system core debugger is to execute. You can get the global task ID from the DISPLAY_TASK_ENVIRONMENT subcommand described later in this chapter. The global task ID consists of two or three hexadecimal bytes. This command opens a window for the system core debugger.

DUMPJOB job output_disposition

This command initiates the system core debugger in the job specified, but it does not give interactive control to the debugger. It produces a separate dump of information for each task in the job. The information that is dumped is determined by the content of the auto command list. Refer to the AUTO subcommand description for details. For the job parameter, you can enter either the active job list ordinal (from the VED AJ display) or the system-supplied name of the job. For the OUTPUT_DISPOSITION parameter, you can specify one of the following values:

RETAIN (or R), which retains the output in the subcatalog $SYSTEM.DUMPS. A separate file is placed in this catalog for each task in the job.

PRINTER (or P), which sends output for automatic printing at a central site line printer. A separate listing is generated for each task in the job.

RETAIN_AND_PRINTER (or RAP), which performs both of the above actions.

The default value is RETAIN_AND_PRINTER. Files saved in the $SYSTEM.DUMPS subcatalog have a retention period of 14 days. Output printed automatically is printed under the $SYSTEM banner with the file name BROKEN_JOB_PFN_LFN.

Default Output Disposition

Disposition of output generated by system core debugger commands can be controlled by the DEFAULT_DEBUG_OUTPUT_DISPOSITION system attribute, which can be changed at any time during system operations. The output destination default can be changed by entering the following command at the system console:

```
SET_SYSTEM_ATTRIBUTE DEFAULT_DEBUG_OUTPUT_DISPOSITION <integer>
```

where integer is one of the following values:

0	Output is sent to the system console during execution of a command.
1	Output is sent to the printer upon completion of the session or command.
2	Output is retained in the $SYSTEM.DUMPS catalog upon completion of the session or command.
3	Output is sent to the printer and is also retained in the $SYSTEM.DUMPS catalog on completion of the session or command.

The default value is 0.

Disposition of output from a DUMPJOB session or the AUTO command is handled differently from that produced by other commands. If the value of the DEFAULT_DEBUG_OUTPUT_DISPOSITION system attribute is 0, output is handled according to the disposition selected for the DUMPJOB session or the AUTO command. For all other values of the DEFAULT_DEBUG_OUTPUT_DISPOSITION system attribute, the attribute takes precedence.

Examples of System Core Debugger Use

WARNING

These examples are provided to create familiarity with the system core debugger. Site analysts are expected to try them on a test system.

Example 1

The following example shows initiation of the system core debugger using both SYSDEBUG and TDEBUG:

1. Enter at the user terminal:

   ```
   TASK A
   LOOP
   LOOPEND
   TASKEND
   ```

   A job appears on the VEDISPLAY ACTIVE_JOBS (VED AJ) display and remains active. Notice the ordinal number the VED AJ display associates with this job.

2. In the critical window on the console, enter:

   ```
   SYSDEBUG ajlo
   ```

   where ajlo is the active job list ordinal number. This activates the system core debugger in the job and opens the system core debugger window.

3. While the system core debugger is active in the job, the terminal where the job was initiated does not respond to NOS/VE. Type a carriage return at the terminal to verify this.

   In the system core debugger window, obtain the task ID of task A by entering:

   ```
   DISPLAY_TASK_ENVIRONMENT
   ```

4. After finding the task ID, enter in the system core debugger window:

   ```
   RUN
   ```

   The system core debugger returns control to the job. The terminal where the job was initiated now responds to NOS/VE.

5. Using the task ID of task A, enter in the critical display window:

   ```
   TDEBUG taskid
   ```

   This activates the system core debugger in the specified task and opens the system core debugger window.

6. While the system core debugger is active in the task, the terminal where the task was initiated responds to NOS/VE. This is because the system core debugger is not currently active in the job monitor task.

7. To terminate the debugger session, enter in the system core debugger window:

   ```
   RUN
   ```

Example 2

The following example intentionally causes an address specification error that invokes the system core debugger. The system procedure ADRSPEC creates the error.

1. Enter in the operator window at the console:

   ```
   SET_SYSTEM_ATTRIBUTE SYSTEM_DEBUG_RING 15
   ```

   Every task running at ring 15 or below that encounters an error is brought up in the system core debugger.

2. At the user terminal, enter:

   ```
   EXECUTE_TASK SP=UUTL P='ADRSPEC'
   ```

3. The task encounters an error and comes up in the system core debugger. Return control to the task by entering in the system core debugger window:

   ```
   RUN
   ```

4. The user terminal receives an error message with the location of the error.

   ```
   --FATAL-- address specification error <address>
   ```

5. To reset the system debug ring for normal running mode, enter at the console:

   ```
   SET_SYSTEM_ATTRIBUTE SYSTEM_DEBUG_RING 0
   ```

Example 3

The following example illustrates the use of breakpoints:

1. Enter at a terminal:

   ```
   LOOP
     WAIT 5000
   LOOPEND
   ```

2. Enter in the console critical window:

   ```
   SYSDEBUG
   ```

3. Enter in the system core debugger window:

```
SELECT JOB <ajlo>
SELECT JOBMONITOR
SELECT HIGHRING 15
```

   where ajlo is the active job list ordinal (from the VED AJ display) for the terminal job in step 1.

4. Set a breakpoint called WAIT at PMP$LONG_TERM_WAIT by entering in the system core debugger window:

```
SET_BREAKPOINT WAIT RNI PMP$LONG_TERM_WAIT
RUN
```

5. Observe the job as it appears in the debugger.

6. To continue execution, enter:

```
RUN
```

7. When complete, be sure to clear breakpoints by entering in the system core debugger window:

```
REMOVE_BREAKPOINT WAIT
SELECT NOJOBS
SELECT NOJOBMONITOR
RUN
```

System Core Debugger Subcommands

All numeric values (not addresses) used as system core debugger command parameters default to decimal. You can specify a radix if desired.

The delimiter between the system core debugger subcommand and the first parameter, and between any two parameters, can either be a comma (,) or a space ( ). You must enter system core debugger subcommands and parameters on one line; continuation lines are not allowed.

System core debugger subcommands can be categorized by function as follows:

Table 9-1. Subcommand Functions

Function	Description
Help	The HELP subcommand accesses the system core debugger HELP file.
Control	Control subcommands control the execution of tasks in which the system core debugger is executing. The control subcommands allow you to hang tasks, abort tasks, and restart program execution after an interrupt occurs. The control subcommands are the following: KILL_TASK HANG RUN RUN ALL
Breakpoint	Breakpoint subcommands select the task in which breakpoints are to be active. The breakpoints and their conditions can be set, changed, and cleared within the selected task. When breakpoints are set, the system core debugger uses debug hardware that assists in the debugging. When breakpoints are activated, the system may run slower than normal because the hardware forces execution through additional paths that detect and act on the breakpoints. When the hardware encounters a breakpoint, it causes the task to be interrupted and calls the system core debugger into the task. Although the system core debugger allows you to set breakpoints in monitor code, the breakpoints are not executed (no interruption occurs), and the system core debugger is not called in. **WARNING** Using breakpoints (especially those applied globally to all tasks) can seriously degrade system performance.

(Continued)

Table 9-1. Subcommand Functions *(Continued)*

Function	Description
	To set breakpoints, you must first select the tasks in which the breakpoints are to be set by using the SELECT subcommand. Once a task is selected, you can set up to 32 breakpoints. The breakpoint subcommands are the following: CHANGE_BREAKPOINT LIST_BREAKPOINT REMOVE_BREAKPOINT SELECT SET_BREAKPOINT
Command list	The AUTO subcommand executes the auto command list. The auto dump command list is built into the system on the release tapes. The commands in this list provide information needed to aid in resolving the problem. The command list subcommand is: AUTO
Format control	Format control subcommands control output disposition of the system core debugger session. Use format control subcommands to print and/or save a permanent file containing output from the system core debugger. Format mode causes all output from subsequent system core debugger subcommands to be written to a local file in the system job. When you turn off format mode, the system, by default, automatically prints the file (by issuing a PRINT_FILE command) and copies the file to a permanent file in the system job catalog. The file is retained for 14 days. The format control subcommand is: SET_OUTPUT_DESTINATION
Log	Log subcommands display the segment number, length, and, optionally, the contents of the system and job logs. The log subcommands are the following: DISPLAY_JOB_LOG DISPLAY_SYSTEM_LOG
Memory	Memory subcommands display and change the contents of virtual memory. The following are the memory subcommands: CHANGE_MEMORY DISPLAY_MEMORY SUPER_CHANGE_MEMORY + (Increment Last DISPLAY_MEMORY Subcommand) - (Decrement Last DISPLAY_MEMORY Subcommand)

(Continued)

Table 9-1. Subcommand Functions *(Continued)*

Function	Description
Stack frame	Stack frame subcommands display information about and contents of specific stack frames. They also perform tracebacks through chains of predecessor stack frames. The stack frame subcommands are the following: DISPLAY_STACK_ENVIRONMENT DISPLAY_STACK_FRAME DISPLAY_TRACE_BACK DISPLAY_CALLS
Task status	Task status subcommands display and monitor the status of tasks in which the system core debugger is executing. The task status subcommands are the following: DISPLAY_ENTIRE_SEGMENT_TABLE DISPLAY_EXECUTION_CONTROL_BLOCK DISPLAY_INITIATED_JOB_LIST_ENT DISPLAY_JOB_TABLES DISPLAY_MONITOR_FAULT DISPLAY_OS_ADDRESS DISPLAY_OS_SYMBOL DISPLAY_REGISTER DISPLAY_SEGMENT_TABLE DISPLAY_SEGMENT_TABLE_EXTENDED DISPLAY_TASK_ENVIRONMENT
Output control	The output control subcommand controls the amount of output that goes to the system console screen. The output control subcommand is: SET_PAGE_WAIT
Subcommand control	The subcommand control subcommand allows repetition of a previous utility subcommand. The subcommand control subcommand is: REPEAT

NOTE

Subcommands that display a large amount of data to the system console when page wait is turned off can be terminated by pressing the STOP key on a Viking 721 console or the END key on a Zenith console. The System Core Debugger utility is then ready for the next subcommand. Also, if page wait is turned on, the display can be terminated when the following line is displayed after entry of a carriage return:

```
<OVER>
```

AUTO Subcommand

Purpose Forces a dump of the job and task environment by executing the entire auto dump command list.

Format AUTO
disposal

Parameters *disposal*

Specifies the disposition of the output generated by this subcommand. You can enter one of the following keywords; the default is RETAIN_AND_PRINTER:

RETAIN or R

Output is retained in the catalog $SYSTEM.DUMPS.

PRINTER or P

Output is automatically listed at a central site line printer.

RETAIN_AND_PRINTER or RAP

Output is automatically listed at a central site line printer and retained in the catalog $SYSTEM.DUMPS.

Remarks

- Output generated by this subcommand has a retention period of 14 days for retained files.
- The current version of the auto command list is as follows:

```
DISPLAY_EXECUTION_CONTROL_BLOCK
DISPLAY_MONITOR_FAULT
DISPLAY_ACTIVE_JOB_LIST
DISPLAY_INITIATED_JOB_LIST_ENT
DISPLAY_TASK_ENVIRONMENT
DISPLAY_STACK_ENVIRONMENT
DISPLAY_JOB_LOG ALL
DISPLAY_MEMORY 00300000000 1000000(16)
DISPLAY_MEMORY 00400000000 1000000(16)
DISPLAY_MEMORY 00500000000 1000000(16)
DISPLAY_MEMORY 00600000000 1000000(16)
DISPLAY_TRACE_BACK 1 1000
```

CHANGE_BREAKPOINT Subcommand

Purpose

Changes the virtual memory address range of a previously established breakpoint.

Format

CHANGE_BREAKPOINT or
CHAB
 name
 address
 offset
 length

Parameters

name

Specifies the user-supplied name for the breakpoint. Breakpoint names are a maximum of eight characters in length. You can specify a maximum of 32 individual breakpoints per task, but only one per subcommand. This parameter is required.

address

Specifies the virtual memory address (PVA) of the breakpoint. This is an 11-digit hexadecimal number addressing a specific byte of memory. This parameter is required.

offset

Specifies the number of bytes from the address at which the breakpoint becomes effective. The specified address plus the offset yields the first byte address of the virtual memory address range for the breakpoint. The default is 0.

length

Specifies the number of bytes for which the breakpoint is valid. The specified address plus the offset plus the length – 1 yields the last byte address of the virtual memory address range. The default is 1.

Examples

The following changes the virtual memory address range of an established breakpoint:

```
CHANGE_BREAKPOINT ALPHA1 00700000048 100(16) 1F(16)

CHAB,ALPHA2,00400000000
```

CHANGE_MEMORY Subcommand

Purpose Changes the value of a specified location in virtual memory.

Format **CHANGE_MEMORY** or
CHAM
  address
  value
  count

Parameters **address**

Specifies the virtual memory address (as a PVA or symbolic address) where the new value is entered (this value is an 11-digit hexadecimal number addressing a specific byte of memory). This parameter is required.

value

Specifies a number that replaces the bytes at the specified address. This parameter is required.

count

Specifies the number of consecutive bytes for which the new value is entered. The count can be a maximum of 8. The default is 1.

Remarks This command is valid even when the PVA specified is not in real memory.

CAUTION

This command may change the contents of any writable virtual memory in the task's address space. Use this command with extreme caution.

Examples The following changes the value of a specified memory location:

```
CHANGE_MEMORY,00700004af1,7B(16)

CHAM 10000000028 00 8
```

DISPLAY_CALLS Subcommand

See the description of the DISPLAY_TRACE_BACK subcommand.

DISPLAY_ENTIRE_SEGMENT_TABLE Subcommand

Purpose Displays the segment descriptor table (SDT) entry and segment descriptor table extended (SDTX) entry for all segments in the current task or in a specified task.

Format **DISPLAY_ENTIRE_SEGMENT_TABLE** or
DISEST
 gtid

Parameters *gtid*

Specifies the global task ID of the task whose segment number is to be displayed. You must specify a task in the job that is currently in the system core debugger or a task in the same job as the task that is currently in the system core debugger. The default task is the current task.

Remarks Output from this command is extensive. You may want to send the output to a printer by using the SET_OUTPUT_DESTINATION subcommand. The output for a segment (for SDT together with SDT extended) is printed on a separate page.

Examples The following entry generates a 100-page listing. Figure 9-1 shows a single page of output:

```
SET_OUTPUT_DESTINATION printer
DISPLAY_ENTIRE_SEGMENT_TABLE
SET_OUTPUT_DESTINATION console
```

```
DISPLAY SEGMENT TABLE ENTRY   5
ste.v1: regular segment
ste.xp: non executable
ste.rp: read uncontrolled
ste.wp: write uncontrolled
ste.r1: 03
ste.r2: 0D
ste.asid: 18E1
ste.key_lock: 00
fill1: 00
asti: 10E3

DISPLAY SEGMENT TABLE EXTENDED ENTRY  5
sdtx.open_validating_ring_number: 01
sdtx.segment_reservation_state: NOT RESERVED
sdtx.soft_attr_set:
sdtx.access_rights: sar_write_extend
sdtx.seg_origin: so_task_template
sdtx.locked_page_count: 00
sdtx.segment_lock: lss_none
sdtx.shadow_info.shadowed_segment_kind: ssk_none
 shadow sfid: 00000000
sdtx.shadow_start_page_number: 000000
sdtx.shadow_length_page_count: 000000
sdtx.stream.last_page_fault: 000000000000A094
sdtx.stream.sequential_accesses: 0000000000000000
sdtx.stream.transfer_size: 0000000000000002
sdtx.stream.streaming: FALSE
sdtx.assign active: FALSE
sdtx.offset_requiring_allocation: 00000000
sdtx.segment_kind: sk_transient_file (sk_first_file)
   (generic kind: sk_file)
 sfid: 0001000
```

Figure 9-1. Output of DISPLAY_ENTIRE_SEGMENT_TABLE

DISPLAY_EXECUTION_CONTROL_BLOCK Subcommand

Purpose Displays the contents of selected fields of the execution control block (XCB) for the current task or a specified task.

Format **DISPLAY_EXECUTION_CONTROL_BLOCK** or
DISECB
gtid

Parameters *gtid*

Specifies the global task ID of the task whose execution control block is to be displayed. You must specify a task in the job that is currently in the system core debugger or a task in the same job as the task that is currently in the system core debugger. The default task is the current task.

Examples The following entry:

```
DISPLAY_EXECUTION_CONTROL_BLOCK
```

displays the contents of selected XCB fields:

```
task name & addr: $JOBMNTR 100300000100
global task id: 007C01
xp.p: 0000101400012C1E
xp.x0: 0201100F00001A00
monitor flags: 8000
system table lock count: 0000000000000200
system flags: 00000000
wait inhib, task terminating, task rethreaded:
  TRUE  FALSE FALSE .
system give up cpu, subsystem give up cpu:
  TRUE  FALSE
parent global task id: 000101
cpu priority: 06
system error count: 00
link: FFFF80000000
task control block: 200400008A00
task id: 00000001
sdtp: 100300000500000002F00000000000000008
sdtx p: 1003000007F000000F0E0000000000000029
cp time: 0000A3B842000009FC05
page wait info: 101400008342
quantum: 0000C350
proc malf count: 00
signal(invalid): 000000 000000000000000000000000000000000000000000
signal(invalid): 000000 000000000000000000000000000000000000000000
signal(invalid): 000000 000000000000000000000000000000000000000000
signal(invalid): 000000 000000000000000000000000000000000000000000
paging statistics:
 pages from disk: 000000C3
 pages reclaimed: 0000062D
 new pages assigned: 00000057
 pages from server: 00000000
 page fault count: 0000000005A4
```

DISPLAY_INITIATED_JOB_LIST_ENT Subcommand

Purpose Displays the initiated job list (IJL) entry for the specified job.

Format **DISPLAY_INITIATED_JOB_LIST_ENT** or
DISIJLE
name

Parameters *name*

Specifies the 19-character, system-supplied job name of the job to be displayed. The default display is the IJL entry of the job that is currently in the system core debugger.

Examples The following is an initiated job list (IJL) for job $0830_0604_AAA_0000:

```
DISPLAY_INITIATED_JOB_LIST_ENT, $0830_0604_AAA_0000


system supplied name $0830_0604_AAA_0000
job name: $SYSTEM
ijl entry status:  ies job in memory non swap
ajl ordinal: 00
kjl ordinal: 0000
ijl swap status:   iss executing
ijl next swap status:  iss null
ijl last swap status:  iss null
active io page count: 000000
active io requests: 0000
swap queue link:
 queue id:  isqi_null
 backward link: 0000
 forward link: 0000
job fixed asid: 0000
long wait aging complete:  FALSE
notify swapper when io complete:  FALSE
scheduling dispatching priority: 0B
dispatching control:
 dispatching control index 01
 dispatching priority: 0B
 user requested dispatching priority: 00
 operator set dispatching priority: 00
 service remaining: FFFFF899E184
 CP service at class swtich: 0000000000000000
job monitor task_id: 000101
job mode:  BATCH
executing task count: 01
multiprocessing allowed:  FALSE
operator force out:  FALSE
swapin candidate:  FALSE
swapin candidate queue: 0000
estimated ready time: 000000000000
last think time: 000000000000
age purge timestamp: 000000000000
sfd purge timestamp: 000000000000
```

```
   . . .

job scheduler data:
 ready task in job: 00
 ready_task_link: 0000
 service accumulator: 000000000000
 service accumulator since swap: 000000000000
 guaranteed service remaining: 000000000000
 last cptime: 0000000000
 last page fault count: 0000000000
 job swap count: 000000
 swap entry status:  ses not swapped
 reason:  null
 priority: 000000
 unaged swap queue priority: 000000
 swapin q priority timestamp: 000000000000
 job class: 01
 service class: 01
 job terminated: 00
job page queue list:
 [37]
   link.bkw: 0274
   link.fwd: 01D9
   count: 009C
 [38]
   link.bkw: 0000
   link.fwd: 0000
   count: 0000
 [39]
   link.bkw: 0F97
   link.fwd: 0F02
   count: 00B7
swap data:
 swap direction:   in
 swap file sfid: 00000000
 swapping io error:  FALSE
 swapped job page count: 0000
 swap file length in pages: 0000
 asid reassigned timestamp: 000000000000
 timestamp: 000000000000
 reassigned job fixed asti: 0000
 swapped job entry:
   available modified page count: 0000
   job page queue count- job fixed: 0000
   job page queue count- job io error: 0000
   job page queue count- job working set: 0000
   swap file descriptor page count: 0000
swap io control:
 spd index: 0000
 next queue id: 00
 next pfti: 0000
 stop pfti: 0000
 swap file descriptor pfti: 0000
sfd_p: FFFF80000000000000000000000000000
system breakpoint selected:  FALSE
delayed swapin work:
statistics:
 cp time:
   time spent in job mode: 0004FDB95E
   time spent in mtr mode: 00026949DD
 paging statistics:
  page in count: 00000D0A
  pages reclaimed from queue: 0000157C
  new pages assigned: 000015A9
  pages from server: 00000000
  page fault count: 000000002CC3
  working set max used: 0270
 perm file space: 0000000000000000
 temp file space: 0000000000000000
 ready task count: 0001
 tasks not in long wait: 0004
job-fixed contiguous pages: 00
hung task in job:  FALSE
job damaged during recovery:  FALSE
maxws aio slowdown display: 00
unable to swap idle flag:  FALSE
queue file info:
 job abort disposition:  terminate on abort
 job recovery disposition:  terminate on recovery
 input file location: 00
relative_priority_enabled:  FALSE
task created after last swap:  FALSE
```

DISPLAY_JOB_LOG Subcommand

Purpose Displays the contents of the job log.

Format **DISPLAY_JOB_LOG** or
DISJL
entries

Parameters *entries*

Specifies the number of log entries that will be displayed. Specify a positive integer n to display the last n entries. Specify the keyword ALL to display all entries in the log. The default is 1,000.

Examples The following subcommand displays the last seven lines of the job log:

```
DISPLAY_JOB_LOG,7
```

The output from this subcommand is:

```
11:48:00.479.PR. Task complete RAP$DISPLAY_MESSAGE_COMMAND
11:48:00.481.PR.     job time =     0.338  monitor time =    0.092
    page faults = 36         max working set = 315
11:48:00.911.PR.
11:48:00.914.PR. ---- SYSTEM ACTIVATION COMPLETE ----
11:48:00.917.PR.
11:48:01.013.PR. Task complete RAP$DISPLAY_MESSAGE_COMMAND
11:48:01.015.PR.     job time =     0.340  monitor time =    0.095
    page faults = 36         max working set = 315
```

DISPLAY_JOB_TABLES Subcommand

Purpose Displays general information about local files or file tables.

Format **DISPLAY_JOB_TABLES** or
DISJT
p_h_name
option
id

Parameters *p_h_name*

Specifies the path handle name of the NOS/VE local file for which data is to be displayed. If you do not specify a path handle name, a list of all task files is provided. If you specify any of the values AVT, CST, LPT, LUT, or MAT for the option field, the value NONE must be specified for p_h_name.

option

Specifies the particular information requested. You can specify the following keywords; the default is that general information is supplied:

AVT

Active volume table information. If you select this option, you must specify NONE for the path handle name field. You must also specify an index location for the identifier field.

CST

CPU state table. If you select this option, you must specify NONE for the path handle name.

GFI

Global file information. If you select this option, you must specify a path handle name.

LPT

Logical PP table information. If you select this option, you must specify NONE for the path handle name field.

LUT

Logical unit table information. If you select this option, you must specify NONE for the path handle name field.

MAT

Mainframe allocation table information. If you select this option, you must specify NONE for the path handle field. You must also specify an index location for the identifier field.

PDE

Path description entry. If you select this option, you must specify a path handle name.

STT

System tape table information.

id

Specifies an index location in the specified table. This field is not valid for keywords other than AVT and MAT.

Examples The following example shows a sample of DISPLAY_JOB_TABLES output if you do not specify a path handle name:

```
path = :$SYSTEM.$SYSTEM.AAM.AAF$44D_LIBRARY.3
    attached =  TRUE    path_handle_name = FS$$__0000498D_A4
        alias = AAF$44D_LIBRARY                    phn = FS$$__000049F2_A5
path = :$SYSTEM.$SYSTEM.AAM.AAF$4DD_LIBRARY.3
    attached =  TRUE    path_handle_name = FS$$__00004ABC_A7
        alias = AAF$4DD_LIBRARY                    phn = FS$$__00004B21_A8
path = :$SYSTEM.$SYSTEM.COMMON.MLF$LIBRARY.3
    attached =  TRUE    path_handle_name = FS$$__00004C50_AB
        alias = AAF$44D_LIBRARY                    phn = FS$$__00004CB5_AC
path = :$SYSTEM.$SYSTEM.OSF$SITE_COMMAND_LIBRARY.3
    attached =  TRUE    path_handle_name = FS$$__0000523B_E8
```

DISPLAY_MEMORY Subcommand

Purpose Displays the contents of a specified address in virtual memory.

Format

DISPLAY_MEMORY or
DISM
  address
  length
  option

Parameters

address

Specifies the starting virtual memory address (as a PVA or symbolic name) of the memory block to be displayed (this is an 11-digit hexadecimal number addressing a specific byte of memory). This parameter is required.

length

Specifies the number of bytes of virtual memory to be displayed. The default is 8.

option

Specifies whether to force the memory to be displayed if the address is beyond the current segment length. The following keywords can be specified; the default is N:

F

Forces the display even if the address, plus the length, extends beyond the current segment length. (This option is only valid for stack segments.)

N

Does not force the display if the address, plus the length, extends beyond the current segment length.

Remarks

- This command is valid even when the PVA specified is not in real memory.
- Console output is formatted for 80-column display. Any other output is formatted for 132-column display.

Examples The following generates a display of memory:

```
DISPLAY_MEMORY,500000500,448

LENGTH      B000

SEGMENT = 005
00000500   0010 CC4A 4D45 2455  4E41 424C 455F 544F     JME$UNABLE_TO
00000510   5F52 4543 4F56 4552  5F43 4154 414C 4F47  _RECOVER_CATALOG
00000520   2020 02D0 0B00 269A  2300 344A 4D00 0445        & # 4JM  E
00000530   4A4D 4524 5441 424C  455F 4C45 4347 5448  JME$TABLE_LENGTH
00000540   535F 4652 4F4D 5F50  524F 4649 4C45 2000  S_FROM_PROFILE
00000550   D00B 0026 ABF5 0071  0000 0000 0000 0000     &    q
00000560   0000 0000 0000 0000  0000 0000 0000 0000
 18 duplicate line(s)
00000690   0000 0000 0000 0000  0000 3006 0000 2C56           0    ,V
000006A0   0000 FFFF 8000 0000  0100 0000 0000 0000
000006B0   0000 0000 0000 0000  0000 0000 0000 0000
```

DISPLAY_MONITOR_FAULT Subcommand

Purpose Displays the monitor fault buffers of the task in which the system core debugger is executing.

Format **DISPLAY_MONITOR_FAULT** or
DISMF

Remarks For more information on the contents of the fault buffer, refer to the NOS/VE System Performance and Maintenance manual, Volume 1.

Examples The following generates a monitor fault buffer display:

```
DISPLAY_MONITOR_FAULT

broken task monitor fault
mntr fault buffer full
p-reg: 0000000000000000
a0: 000000000000
mcr: 0000
ucr: 0000
fault-id: broken task
p-register: 0000000000000000
a0: 000000000000
a1: 000000000000
a2: 000000000000

broken task monitor fault
mntr fault buffer full
p-reg: 0000000000000000
a0: 000000000000
mcr: 0000
ucr: 0000
fault-id: broken task
p-register: 0000000000000000
a0: 000000000000
a1: 000000000000
a2: 000000000000

broken task monitor fault
mntr fault buffer full
p-reg: 0000000000000000
a0: 000000000000
mcr: 0000
ucr: 0000
fault-id: broken task
p-register: 0000000000000000
a0: 000000000000
a1: 000000000000
a2: 000000000000
```

```
broken task monitor fault
mntr fault buffer full
p-reg: 0000000000000000
a0: 000000000000
mcr: 0000
ucr: 0000
fault-id: broken task
p-register: 0000000000000000
a0: 000000000000
a1: 000000000000
a2: 000000000000
```

DISPLAY_OPERATING_SYSTEM_ADDRESS Subcommand

Purpose Displays the symbolic information for a specified operating system address. For example, if the address is found within a data section, the module name, section name, and section offset are displayed. If the address is within executable code, the module, procedure, and procedure offset are displayed.

Format **DISPLAY_OPERATING_SYSTEM_ADDRESS** or
DISOSA
  address

Parameters **address**

Specifies the process virtual address (PVA) for the operating system information to display. (This is an 11-digit hexadecimal number addressing a specific byte of memory.) This parameter is required.

Examples The following entry:

```
DISPLAY_OPERATING_SYSTEM_ADDRESS 600012440
```

generates a display similar to:

```
M=IOM$QUEUE_PP_REQUEST
S=IOP$QUEUE_PP_REQUEST      O=60     S=IOP$QUEUE_PP_REQUEST          O= 60
```

In the display, both procedure names and section names are displayed as S = NAME.

DISPLAY_OPERATING_SYSTEM_SYMBOL Subcommand

Purpose Displays the module name and the virtual address of a specified operating system symbol.

Format **DISPLAY_OPERATING_SYSTEM_SYMBOL** or
DISOSS
  name

Parameters **name**

Specifies the name of the operating system symbol for which the virtual address is to be displayed. The symbol name is a 1- to 31-character name. This parameter is required.

Examples The following entry:

```
DISPLAY_OPERATING_SYSTEM_SYMBOL,PMP$LOAD
```

generates a display similar to:

```
M=LOM$LOADER_EXECUTIVE              S= 1C O= BCAE8
```

This display indicates that procedure PMP$LOAD is located in module LOM$LOADER_EXECUTIVE at segment 1C(16), offset 0BCAE8(16).

DISPLAY_REGISTER Subcommand

Purpose Displays the contents of a specified register of an interrupted task.

Format **DISPLAY_REGISTER** or
DISR
  reg_type
  reg_num
  type

Parameters **reg_type**

Specifies the type of register to be displayed. This parameter is required. You can specify the following keywords:

A

Address register.

X

Operand register.

P

Program address register.

reg_num

Specifies the hexadecimal or decimal number of the register to be displayed. This parameter is valid for the A and X keywords. The. registers are numbered from 0 through 15 decimal or from 0 through F hexadecimal. The default is 0.

type

Specifies how the data of the register contents is to be interpreted.This parameter is valid only when X is specified for the reg_type parameter. Specify one of the following types; the default is HEX:

ASCII or ASC

The register contents is to be interpreted as ASCII data.

DEC

The register contents is to be interpreted as a decimal number.

HEX

The register contents is to be interpreted as a hexadecimal number.

Examples The following entry:

```
DISPLAY_REGISTER P
```

generates a display similar to:

```
P=1 00D 0000010E
```

DISPLAY_SEGMENT_TABLE Subcommand

Purpose Displays the segment descriptor table (SDT) entry for a segment within the current task or within a specified task.

Format **DISPLAY_SEGMENT_TABLE** or
DISST
  number
  gtid

Parameters **number**

Specifies the segment number whose segment descriptor table entry is to be displayed. This parameter is required.

gtid

Specifies the global_task_id of the task whose segment number is to be displayed. It must specify a task in the job that is currently in the system core debugger or in the same job as the task that is currently in the system core debugger. The default is the current task.

Examples The following entry:

```
DISPLAY_SEGMENT_TABLE 5
```

generates a display similar to:

```
DISPLAY SEGMENT TABLE ENTRY   5
ste.v1: regular segment
ste.xp: non executable
ste.rp: read uncontrolled
ste.wp: write uncontrolled
ste.r1: 03
ste.r2: 0D
ste.asid: 18E1
ste.key_lock: 00
fill1: 00
asti: 10E3
```

DISPLAY_SEGMENT_TABLE_EXTENDED Subcommand

Purpose Displays the segment descriptor table extended (SDTX) entry for a segment within the current task or within a specified task.

Format **DISPLAY_SEGMENT_TABLE_EXTENDED** or
DISSTE
  number
  gtid

Parameters **number**

Specifies the segment number whose segment descriptor table extended entry is to be displayed. This parameter is required.

gtid

Specifies the global_task_id of the task whose segment number is to be displayed. It must specify a task in the job that is currently in the system core debugger or in the same job as the task that is currently in the system core debugger. The default is the current task.

Examples The following entry:

```
DISPLAY_SEGMENT_TABLE_EXTENDED 5
```

generates a display similar to:

```
DISPLAY SEGMENT TABLE EXTENDED ENTRY   5
sdtx.open_validating_ring_number: 01
sdtx.segment_reservation_state: NOT RESERVED
sdtx.soft_attr_set:
sdtx.access_rights: sar_write_extend
sdtx.seg_origin: so_task_template
sdtx.locked_page_count: 00
sdtx.segment_lock: lss_none
sdtx.shadow_info.shadowed_segment_kind: ssk_none
 shadow sfid: 00000000
sdtx.shadow_start_page_number: 000000
sdtx.shadow_length_page_count: 000000
sdtx.stream.last_page_fault: 000000000000A094
sdtx.stream.sequential_accesses: 0000000000000000
sdtx.stream.transfer_size: 0000000000000002
sdtx.stream.streaming: FALSE
sdtx.assign active: FALSE
sdtx.offset_requiring_allocation: 00000000
sdtx.segment_kind: sk_transient_file (sk_first_file)
   (generic kind: sk_file)
 sfid: 00010000
```

DISPLAY_STACK_ENVIRONMENT Subcommand

Purpose Displays the segment number and the top of stack values for each ring in which a task is executing.

Format **DISPLAY_STACK_ENVIRONMENT** or **DISSE**

Parameters None.

Remarks When the current system core debugger session is retaining and/or printing (as documented earlier in this chapter), the contents of the segment (from 0 to top-of-stack) are also displayed.

Examples The following entry:

```
DISPLAY_STACK_ENVIRONMENT
```

generates a display at the system console similar to the following:

```
STACK INFO: RING/SEG/HIGH 01 000F 00000D80
STACK INFO: RING/SEG/HIGH 02 0010 00000000
STACK INFO: RING/SEG/HIGH 03 0011 000005E8
STACK INFO: RING/SEG/HIGH 0B 0040 00003398
```

DISPLAY_STACK_FRAME Subcommand

Purpose Displays all or part of the information in a specified stack frame.

Format **DISPLAY_STACK_FRAME** or
DISSF
frame
option

Parameters *frame*

Specifies the number of the stack frame for which the information is to be displayed. Stack frame number 1 is associated with the interrupted procedure, stack frame number 2 is associated with the interrupted procedure's predecessor, and so on. The default stack frame displayed is number 1.

option

Specifies how much of the selected stack frame is to be displayed. You can specify the following keywords; the default is FULL:

AUTO

Displays the memory contents of the specified stack frame.

SAVE

Displays the contents of the save region of the specified stack frame.

FULL

Displays the entire contents of the specified stack frame. This includes information displayed for both AUTO and SAVE.

Examples The following entry:

```
DISPLAY_STACK_FRAME
```

generates a display similar to:

```
STACK FRAME 001          SEGMENT=00F

SAVE AREA

P=1 00D 0000010E          VMID=0
UM=FF77     UCR=0000     MCR=0000

A0=1 00F 00000000        A1=1 00F 00000000
A2=3 011 000004E0        A3=1 00E 00000010
A4=1 00E 00000050        A5=1 003 00000100

X1=00000000  80000000        X2=00000000  00000005
X3=00000000  00002000        X4=00000000  00000000
```

DISPLAY_SYSTEM_LOG Subcommand

Purpose Displays the contents of the system log.

Format **DISPLAY_SYSTEM_LOG** or
DISSL
entries

Parameters *entries*

Specifies the number of log entries to be displayed. Specifying a positive integer n indicates that the last n entries are displayed. You can specify the keyword ALL to display all entries in the log. The default is 1000.

Examples The following entry:

```
DISPLAY_SYSTEM_LOG
```

generates a display similar to:

```
14:07:32.233.$0860_0505_AAA_0000.SY.*********** INSTALLATION DEADSTART ON November 16,1988
14:08:58.264.$0860_0505_AAA_0000.RC.  Recovery not necessary
14:09:11.752.$0860_0505_AAA_0000.PR.----  System Deadstart Completed ---
14:11:36.820.$0860_0505_AAA_0000.SY.diste
14:11:49.969.$0860_0505_AAA_0000.SY.diste
14:12:12.552.$0860_0505_AAA_0000.SY.help setmsf
14:14:47.302.$0860_0505_AAA_0001.CI.LOGIN, USER=$SYSTEM, FAMILY_NAME=$SYSTEM, ACCOUNT=NONE,
  PROJECT=NONE, JOB_CLASS=MAINTENANCE, USER_JOB_NAME=HPA_VE$MONITOR_J
14:14:47.762.$0860_0505_AAA_0002.CI.LOGIN, USER=$SYSTEM, FAMILY_NAME=$SYSTEM, ACCOUNT=NONE,
  PROJECT=NONE, JOB_CLASS=BATCH, USER_JOB_NAME=TERJ_H
14:14:57.706.$0860_0505_AAA_0003.CI.LOGIN, USER=$SYSTEM, FAMILY_NAME=$SYSTEM, ACCOUNT=NONE,
  PROJECT=NONE, JOB_CLASS=BATCH, USER_JOB_NAME=VALIDATION_J
14:14:58.939.$0860_0505_AAA_0004.CI.LOGIN, USER=$SYSTEM, FAMILY_NAME=$SYSTEM, ACCOUNT=NONE,
  PROJECT=NONE, JOB_CLASS=BATCH, USER_JOB_NAME=VALIDATION_J
14:15:00.912.$0860_0505_AAA_0005.CI.LOGIN, USER=$SYSTEM, FAMILY_NAME=$SYSTEM, ACCOUNT=NONE,
  PROJECT=NONE, JOB_CLASS=BATCH, USER_JOB_NAME=VALIDATION_J
14:15:01.791.$0860_0505_AAA_0006.CI.LOGIN, USER=$SYSTEM, FAMILY_NAME=$SYSTEM, ACCOUNT=NONE,
  PROJECT=NONE, JOB_CLASS=BATCH, USER_JOB_NAME=VALIDATION_J
14:15:06.317.$0860_0505_AAA_0006.CI.ved aj
```

DISPLAY_TASK_ENVIRONMENT Subcommand

Purpose Displays the execution control block (XCB) of all tasks in the job in which the system core debugger is executing.

Format **DISPLAY_TASK_ENVIRONMENT** or **DISTE**

Parameters None.

Remarks The system displays selected fields of the XCB for each task when the system core debugger is not displaying output to the console. If you enter this subcommand when displaying output to the console, only the task name, the XCB address, and the global task identification are displayed.

Examples The following displays a task environment:

```
SET_OUTPUT_DESTINATION printer
DISPLAY_TASK_ENVIRONMENT
SET_OUTPUT_DESTINATION console
```

The resulting output is:

```
task name & addr: CLP$TASK_TASKEND 100300003DE0
global task id: 007D01
xp.p: 0000301D001C905E
xp.x0: 1B00000000000000
monitor flags: 0000
system table lock count: 0000000000000000
system flags: 00000000
wait inhib, task terminating, task rethreaded:
  FALSE FALSE FALSE
system give up cpu, subsystem give up cpu:
  FALSE FALSE
parent global task id: 007C01
cpu priority: 06
system error count: 00
link: 100300000100
task control block: 20040000AAE0
task id: 00000002
sdtp: 100300004200000003280000000000000008
sdtx p: 1003000045400000102D0000000000000029
cp time: 00007EC9CF000005B129
page wait info: D00B0016F3E7
quantum: 0000C350
proc malf count: 00
signal(invalid): 000000 0000000000000000000000000000000000
signal(invalid): 000000 0000000000000000000000000000000000
signal(invalid): 000000 0000000000000000000000000000000000
signal(invalid): 000000 0000000000000000000000000000000000
paging statistics:
 pages from disk: 0000000A
 pages reclaimed: 00000088
 new pages assigned: 00000188
 pages from server: 00000000
 page fault count: 000000000218
```

```
task name & addr: $JOBMNTR 100300000100
global task id: 007C01
xp.p: 0000101400012C1E
xp.x0: 0201100F00001A00
monitor flags: 8000
system table lock count: 0000000000000200
system flags: 00000000
wait inhib, task terminating, task rethreaded:
  TRUE  FALSE FALSE
system give up cpu, subsystem give up cpu:
  TRUE  FALSE
parent global task id: 000101
cpu priority: 06
system error count: 00
link: FFFF80000000
task control block: 200400008A00
task id: 00000001
sdtp: 10030000051800000320000000000000008
sdtx p: 1003000007F000000F0E0000000000000029
cp time: 0000A3B842000009FC05
page wait info: 101400008342
quantum: 0000C350
proc malf count: 00
signal(invalid): 000000 000000000000000000000000000000000000000000
signal(invalid): 000000 000000000000000000000000000000000000000000
signal(invalid): 000000 000000000000000000000000000000000000000000
signal(invalid): 000000 000000000000000000000000000000000000000000
paging statistics:
 pages from disk: 000000C3
 pages reclaimed: 0000062D
 new pages assigned: 00000057
 pages from server: 00000000
 page fault count: 0000000005A4
```

DISPLAY_TRACE_BACK Subcommand

Purpose Provides information relevant to stack frames associated with the interrupted procedure and its predecessor procedures.

Format **DISPLAY_TRACE_BACK** or
DISPLAY_CALLS or
DISTB or
DISC
start
count
option
address

Parameters *start*

Specifies the number of the first stack frame to be displayed. Stack frame number 1 is associated with the interrupted procedure, stack frame number 2 is associated with the interrupted procedure's predecessor, and so on. The default is 1.

count

Specifies the number of stack frames to be displayed. The default is 1.

option

Specifies the amount of information to be displayed for each stack frame. You can specify one of the following keywords; the default is FULL:

SHORT

Display the module name and the P-address.

FULL

Display the module name, the P-address, and the PVAs pointing to the top of the stack and the current stack frame save area.

address

Specifies a stack frame's starting virtual memory address as a PVA. If you specify this parameter, the start parameter is ignored.

Remarks The following information is displayed for each stack frame:

Stack frame number
Current P-address of the associated predecessor
PVA of the start of the stack frame
PVA of the stack frame save area
Module name, procedure name, and offset within the procedure (for NOS/VE addresses only).

Examples The following subcommand provides information on interrupted procedures:

```
DISPLAY_TRACE_BACK,1,5,full
```

HANG_TASK Subcommand

Purpose Causes the task to hang in an infinite delay loop when the system core debugger is exited. The task is left in a state where it can be reentered at a later time.

Format **HANG_TASK** or
HANT

Parameters None.

Remarks

- This subcommand acts as a toggle switch. If you enter the subcommand twice before you exit the system core debugger, the hang condition is cleared, and the delay loop is not executed.
- When you hang a task using this subcommand, you can restart the task at the previous point of interruption by issuing the HANG_TASK subcommand again, followed by RUN.

HELP Subcommand

Purpose Accesses the system core debugger HELP file at any time after starting the system core debugger.

Format **HELP**
name

Parameters *name*

Specifies the name of the subcommand for which you want HELP information. You can specify the full name of the subcommand or its abbreviation. The default HELP display is a list of all the system core debugger subcommands and their parameters.

Examples The following entry generates the display shown:

```
HELP DISPLAY_MEMORY

ALIAS        NOMINAL NAME
parameter    parameter    parameter    parameter    parameter ...
=============================================================
DISM         DISPLAY_MEMORY
address      length       option
```

KILL_TASK Subcommand

Purpose Allows the site analyst or operator to abort a specified task.

Format **KILL_TASK** or
KILT
frame
ring
no_ch

Parameters *frame*

Specifies a stack frame that will be damaged to cause the stack to abort. This parameter is optional. The default allows the debugger to determine a "safe" stack frame that can be damaged.

ring

Specifies a ring number in which the first stack frame is to be damaged. This parameter is optional. If you specify the frame parameter, this parameter is ignored.

no_ch

Specifies that the damaged stack frame is to be cut off from previous stack frames. This prevents the task from using condition handlers that may have been established earlier in the stack. This parameter is optional. If you specify the frame parameter, this parameter is ignored.

Remarks

CAUTION

This command could cause the job that contains the current debugged task to abort. If the job is the system job, this command could cause the system to abort.

LIST_BREAKPOINT Subcommand

Purpose

Lists the currently active breakpoints by name with their associated conditions, the segment number, and the first and last byte in which the breakpoint is active.

Format

LIST_BREAKPOINT or
LISB
    name

Parameters

name

Specifies a breakpoint name. The conditions for this breakpoint are displayed. The default system core debugger display is all active breakpoint names and their conditions.

Examples

The following lists a currently active breakpoint:

```
LIST_BREAKPOINT ALPHA1
```

The following lists all active breakpoints:

```
lisb
```

REMOVE_BREAKPOINT Subcommand

Purpose Deactivates a previously active breakpoint.

Format **REMOVE_BREAKPOINT** or
REMB
  name
  cond

Parameters **name**

Specifies the name of the breakpoint to be deactivated. If you specify only the name parameter, all conditions set for that breakpoint are cleared. This parameter is required.

cond

Specifies the particular condition to clear for the named breakpoint. If this is the only condition set, the entire breakpoint is cleared. If more than one condition is set, the other conditions and the breakpoint remain in effect. You can specify the following keywords:

RNI

Read next instruction. This is the most commonly used breakpoint condition.

READ

Read from virtual memory.

WRITE

Write to virtual memory.

BRANCH

Branch to another location.

CALL

Call to another module.

DIVFLT

Division fault.

ARLOS

Arithmetic loss of significance.

AROVFL

Arithmetic overflow.

EXOVFL

Exponential overflow.

EXUNFL

Exponential underflow.

FPLOS

Floating-point loss of significance.

FPINDEF

Floating-point indefinite operand condition.

INVBDP

Invalid business data processing (BDP) instruction.

Examples The following clears a breakpoint condition:

```
REMOVE_BREAKPOINT ALPHA1 EXOVFL
```

The following clears all conditions for breakpoint ALPHA2:

```
REMB ALPHA2
```

REPEAT Subcommand

Purpose Allows the operator or analyst to repeat the previous system core debugger command without retyping any parameters.

Format **REPEAT** or
R

Parameters None.

Remarks This subcommand is valid only if the immediately preceding subcommand was one of the following:

CHANGE_MEMORY
DISPLAY_CALLS
DISPLAY_ENTIRE_SEGMENT_TABLE
DISPLAY_EXECUTION_CONTROL_BLOCK
DISPLAY_INITIATED_JOB_LIST
DISPLAY_JOB_LOG
DISPLAY_JOB_TABLES
DISPLAY_MEMORY
DISPLAY_MONITOR_FAULT
DISPLAY_OPERATING_SYSTEM_ADDRESS
DISPLAY_OPERATING_SYSTEM_SYMBOL
DISPLAY_REGISTER
DISPLAY_SEGMENT_TABLE
DISPLAY_SEGMENT_TABLE_EXTENDED
DISPLAY_STACK_ENVIRONMENT
DISPLAY_STACK_FRAME
DISPLAY_SYSTEM_LOG
DISPLAY_TASK_ENVIRONMENT
DISPLAY_TRACE_BACK
LIST_BREAKPOINT
SUPER_CHANGE_MEMORY

RUN Subcommand

Purpose Causes the system core debugger to exit from the selected task and restarts program execution.

Format **RUN**
option

Parameters *option*

Specifies whether all tasks queued to enter the system core debugger should be dequeued when the current task exits. The keyword ALL dequeues all queued tasks, preventing them from entering the debugger and causing normal execution. By default, the current task exits the debugger and all other tasks remain in the queue to enter the debugger, one at a time.

Remarks The interrupted task is resumed at the point of interruption unless the HANG_TASK toggle is set, in which case the task goes into an idle state.

SELECT Subcommand

Purpose Specifies the task or tasks in which breakpoints are to be set, or identifies a new control option for the task or tasks previously selected.

Format **SELECT** or
SEL
option

Parameters *option*

Specifies the control option used to control the tasks in which the system core debugger is active. You can specify the following keywords:

ALLJOBS

Activates the system core debugger in all active jobs.

HIGHRING integer

Specifies the greatest ring number in which system core debugger traps are recognized. Traps occurring on instructions with ring numbers greater than this ring number are ignored. HIGHRING 3 is a predefined control option.

JOB integer or name

Allows activation of the system core debugger in the job with the specified job ordinal or system-supplied name. You can find a job's ordinal by checking the VED AJ display on the system display console. Using the system-supplied name (the abbreviated form of the job name can be used) is recommended over the active job list ordinal. If the job is swapping, the job ordinal may not be reliable because the desired job could swap in at a different job ordinal.

JOBMONITOR

Activates the system core debugger in the job monitor task.

NOJOB integer or name

Prevents the system core debugger from executing in the job previously specified in a JOB keyword.

NOJOBS

Prevents the system core debugger from executing in all active jobs.

NOJOBMONITOR

Prevents the system core debugger from executing in the job monitor task.

NOUSER

Prevents the system core debugger from executing in user tasks.

USER

Activates the system core debugger in user tasks (those that are not job monitor tasks).

Remarks

- The breakpoints are not active in any task when you first call the system core debugger. You must first use the SELECT subcommand to choose the task(s) in which the system core debugger is to activate breakpoints.
- Selections remain in effect until they are explicitly changed with another SELECT subcommand or you clear the breakpoint.
- You can specify only one control option in a SELECT subcommand. You must enter multiple SELECT subcommands to enable more than one control option.

Examples

The following specifies a control option:

```
SELECT JOBMONITOR

SELECT,HIGHRING 11

SELECT NOJOB
```

SET_BREAKPOINT Subcommand

Purpose Identifies the breakpoint name and establishes the specific condition that causes a program interrupt. When the specified condition is met after the task has been restarted, the interrupt occurs. You can optionally specify a virtual address range within which the specified condition causes an interrupt.

Format **SET_BREAKPOINT** or
SETB
  name
  cond
  base
  offset
  length

Parameters **name**

Specifies the user-supplied name for the breakpoint. Breakpoint names are a maximum of eight characters in length. You can specify a maximum of 32 individual breakpoints per task, but only one per subcommand. This parameter is required.

cond

Specifies the condition that causes an interrupt to occur. This parameter is required. You must specify one of the following keywords:

RNI

Read next instruction. This is the most commonly used breakpoint condition.

READ

Read from virtual memory.

WRITE

Write to virtual memory.

BRANCH

Branch to another location.

CALL

Call to another module.

DIVFLT

Division fault.

ARLOS

Arithmetic loss of significance.

AROVFL

Arithmetic overflow.

EXOVFL

Exponential overflow.

EXUNFL

Exponential underflow.

FPLOS

Floating-point loss of significance.

FPINDEF

Floating-point indefinite operand condition.

INVBDP

Invalid business data processing instruction.

base

Specifies the address of the breakpoint as a process virtual address (PVA) or as a symbolic name. This is an 11-digit hexadecimal number addressing a specific byte of memory. This parameter is required when you are establishing a new breakpoint. You cannot specify this parameter when you are adding conditions to an existing breakpoint.

offset

Specifies the number of bytes from the specified address that the breakpoint becomes effective. The address plus the offset yields the first byte address of the virtual memory address range of the breakpoint. This parameter is optional when you are specifying a new breakpoint. You cannot specify this parameter when adding conditions to an existing breakpoint. The default is 0.

length

Specifies the number of bytes for which the breakpoint is valid. The address plus the offset plus the length –1 yields the last byte address of the virtual memory address range. This parameter is optional when you are specifying a new breakpoint. You cannot specify this parameter when adding conditions to an existing breakpoint. The default is 1.

Examples The following establishes program interrupt conditions:

```
SET_BREAKPOINT,ALPHA1,EXOVFL,10000000000(16),304F(16)

SETB ALPHA2 EXUNFL 00700000028

SETB ALPHA1 DIVFLT

SETB BB1 RNI PMP$EXIT
```

SET_OUTPUT_DESTINATION Subcommand

Purpose Determines output destination for all subsequent system core debugger subcommands.

Format **SET_OUTPUT_DESTINATION** or
SETOD
disposal
title

Parameters *disposal*

Specifies the disposition of the output generated by subsequent subcommands. You can enter one of the following keywords; the default is RETAIN_AND_PRINTER:

CONSOLE or C

Output displayed at the system console.

RETAIN or R

Output retained in the catalog $SYSTEM.DUMPS.

PRINTER or P

Output automatically listed at a central site line printer.

RETAIN_AND_PRINTER or RAP

Output listed at central site line printer and retained in catalog $SYSTEM.DUMPS.

title

Specifies a set of characters that appears on the first page of the printed output. A string must be enclosed in apostrophes. The default is a blank string.

Examples The following specifies that the file should be retained:

```
SET_OUTPUT_DESTINATION retain
```

The following specifies the first line of printed output:

```
SET_OUTPUT_DESTINATION,,'output from debug session 4/30-4'
```

The following specifies the console as the destination for output:

```
SET_OUTPUT_DESTINATION console
```

SET_PAGE_WAIT Subcommand

Purpose

Sets the number of lines to be displayed on the console before waiting to continue the display.

Format

SET_PAGE_WAIT or
SETPW
  number

Parameters

number

Specifies the number of lines to be displayed on the system console before the display waits to continue. The default is 0, which means page wait is not in effect. In this case, all output of a command is displayed in a continuous stream, regardless of the number of lines.

Remarks

- When page wait is in effect and the display is waiting to continue, the string <OVER> is displayed. To continue the display, enter a carriage return.
- If the string <OVER> is being displayed, output can be terminated by entering any characters followed by a carriage return. Output can also be terminated at any time by using the STOP key on a Control Data Viking-X 721 console or the END key on a Zenith console. This is especially helpful during extensive output that is not needed after certain information has been displayed during a session that has page wait turned off.
- If you never enter the SET_PAGE_WAIT subcommand, the number of lines lines displayed is equal to the number of lines specified on the SET_SYSTEM_ATTRIBUTE command using the DEBUGGER_PAGE_WAIT_LINES attribute (if it is nonzero). If the attribute equals 0, the number of lines displayed is equal to the number of lines displayed in the current debugger window. Therefore, the number of lines can change as the debugger window changes size.
- This subcommand affects output displayed to the the system console only; output directed elsewhere is not affected.

Examples

The following command sets the number of displayed lines to 15:

```
SET_PAGE_WAIT 15
```

The following command turns off page wait (allows continuous scrolling of output):

```
SET_PAGE_WAIT 0
```

SUPER_CHANGE_MEMORY Subcommand

Purpose Changes the contents of the specified location in virtual memory exactly as the CHANGE_MEMORY subcommand with the additional capability to change segment access attributes to allow the change and then replace the original segment access attributes.

Format **SUPER_CHANGE_MEMORY** or
SUPCM
  address
  value
  count

Parameters **address**

Specifies the virtual memory address (as a PVA) where the new value is to be entered. This is an 11-digit hexadecimal number addressing a specific byte of memory. This parameter is required.

value

Specifies a number as the value to replace the bytes at the specified address. This parameter is required.

count

Specifies the number of bytes for which the new value is entered. The count can be a maximum of 8. The default is 1.

Remarks This subcommand is valid even when the PVA specified is not in real memory.

CAUTION

This subcommand may change the contents of any virtual memory in the task address space. The subcommand also updates the address on the disk image when the page is aged out to disk. Use this subcommand with extreme caution.

Examples The following changes the contents of a specified virtual memory location:

```
SUPER_CHANGE_MEMORY 00700004af1 7B(16)

SUPCM,10000000028,00,8
```

– (Decrement Last DM Subcommand)

Purpose	Decrements the most recent DM subcommand starting address by the most recent value of the length parameter. This allows you to page backward through virtual memory.
Format	– or **M**
Parameters	None.

+ (Increment Last DM Subcommand)

Purpose Increments the most recent DM subcommand starting address by the most recent value of the length parameter. This allows you to page forward through virtual memory.

Format + or
P

Parameters None.

ANALYZE_DUMP Utility 10

ANALYZE_DUMP Utility 10

The ANALYZE_DUMP utility aids in analyzing NOS/VE system failures and mainframe hardware failures. The ANALYZE_DUMP utility analyzes a tape created by the Express Deadstart Dump (EDD) utility. On dual-state systems, the NVE subsystem may also produce a dump tape if NOS/VE fails. This tape is compatible with tapes produced by EDD and is therefore acceptable as input to the ANALYZE_DUMP utility.

You can use the ANALYZE_DUMP utility as an interactive or batch job.

You can use the ANALYZE_DUMP utility to display the following information:

- Central memory, displayed in numeric and ASCII format, and accessed in virtual or real address mode.
- Peripheral processor (PP) memory.
- Maintenance registers for the input/output unit, memory, and processors.
- Formatted display of exchange packages.
- Process information resulting from an analysis of stack segments.

Additional subcommands provide access to buffer controlware and control store information in the dump. In addition, functions are available to help you create SCL procedures that perform operations on the dump contents.

ANALYZE_DUMP Utility Subcommands and Functions

Detailed descriptions of the subcommands and functions of the ANALYZE_DUMP utility reside in both an online manual and a document that you can list.

To access the online manual, either enter the HELP command while in the utility or execute the following command from outside the utility:

```
/explain manual=analyze_dump
```

To print a copy of the ANALYZE_DUMP utility documentation, enter:

```
/print_file file=$system.manuals.line_printer_manuals.analyze_dump
```

The following sections provide summary lists of ANALYZE_DUMP subcommands and functions.

List of Subcommands

The following is an alphabetical list of the subcommands of the ANALYZE_DUMP utility initiated by the ANALYZE_DUMP command:

CHANGE_DEFAULT
Changes the default setting of certain parameters on ANALYZE_DUMP utility subcommands.

CHANGE_PROCESSOR_REGISTER
Changes the setting of the processor registers which are used for virtual address translation.

COMPARE_CONTROL_STORE

Compares the control store record read from the dump with a control store record previously saved on a file.

COPY_BUFFER_CONTROLWARE

Makes a binary copy of the channel buffer controlware record from the dump and writes it to the specified output file.

COPY_CONTROL_STORE

Makes a binary copy of the control store record from the dump and writes it to the specified output file.

COPY_MEMORY

Copies central memory to a file.

COPY_PP_MEMORY

Copies peripheral processor (PP) memory to a file.

DISPLAY_BUFFER_CONTROLWARE

Displays the channel buffer controlware.

DISPLAY_CALL

Produces a formatted display of the dynamic call chain for a selected task.

DISPLAY_CHANNEL_CONDITIONS

Displays the channel status, if available, for each channel.

DISPLAY_CIO_REGISTERS

Displays the CIO channel registers.

DISPLAY_CONTROL_STORE

Displays the control store part number and revision level.

DISPLAY_DFT_BUFFER

Displays the DFT/OS buffer.

DISPLAY_DUAL_STATE_BUFFER

Displays the dual-state buffer (EICB).

DISPLAY_DUMP_INFORMATION

For an EDD dump, displays information describing the conditions under which the dump was taken. For a *RUN dump, displays the memory dump option and the version level of the *RUN utility.

DISPLAY_DUMP_RECORD

Displays the contents of a specified development dump record.

DISPLAY_DUMP_RECORD_LIST

Returns a listing of all development dump record identifiers.

DISPLAY_ENVIRONMENT_INTERFACE

Displays the environment interface buffer.

DISPLAY_EXCHANGE_PACKAGE

Produces a formatted display of the contents of a selected exchange package.

DISPLAY_MAINTENANCE_REGISTERS

Displays the contents of the processor maintenance registers, I/O maintenance registers, and/or memory maintenance registers.

DISPLAY_MEMORY

Displays central memory in numeric or ASCII format.

DISPLAY_PP_MEMORY

Displays the contents of peripheral processor (PP) memory.

DISPLAY_PP_REGISTERS

Displays the contents of the A, P, Q, and K registers, if available, for each peripheral processor.

DISPLAY_REGISTER_FILE

Displays the contents of the processor register file.

QUIT

Terminates the ANALYZE_DUMP utility.

List of Functions

The following is an alphabetical list of the functions associated with the ANALYZE_DUMP utility:

$ANALYZE_DUMP_OUTPUT

Returns the value specified by the OUTPUT parameter of the ANALYZE_DUMP command.

$ANALYZE_DUMP_TITLE

Returns the value specified by the TITLE parameter of the ANALYZE_DUMP command.

$AVAILABLE

Indicates whether the specified element is present in the dump.

$BIT

Indicates whether the specified bit is set in the specified integer.

$BUFFER_CONTROLWARE

Returns the contents of a channel buffer controlware word; the result is an integer.

$BUFFER_CONTROLWARE_STRING

Returns the contents of a channel buffer controlware word; the result is a string.

$CHANNEL_AVAILABLE

Indicates whether the specified channel buffer controlware is in the dump.

$CONTROL_STORE

Returns the contents of a processor control store (controlware) word; the result is a string.

$CONTROL_STORE_BYTE

Returns the contents of a specified byte within a processor control store (controlware) word; the result is an integer.

$DUMP_RECORD

Returns the contents of a specified development dump record; the result is an integer.

$DUMP_RECORD_AVAILABLE

Indicates whether a specified development dump record is present on the dump tape.

$DUMP_RECORD_LENGTH

Returns an integer that specifies the length of the specified development dump record.

$MAINTENANCE_REGISTER

Returns the contents of the specified maintenance register.

$MEMORY

Returns the contents of central memory; the result is an integer.

$MEMORY_STRING

Returns the contents of central memory; the result is a string.

$MODULE

Returns the module name corresponding to a given address.

$NIL_PVA

Indicates whether the process virtual address (PVA) is nil.

$OFFSET

Returns the segment offset portion of a given address.

$PP_AVAILABLE

Indicates whether the specified PP memory is in the dump.

$PP_MEMORY

Returns the contents of a word of PP memory.

$PROCESS_REGISTER

Returns the contents of a process register as defined by an exchange package.

$REAL_MEMORY_ADDRESS

Translates a process virtual address (PVA) or a system virtual address (SVA) into a real memory address (RMA).

$REGISTER_FILE

Returns the contents of a processor register file word; the result is an integer.

$REGISTER_FILE_STRING
Returns the contents of a processor register file word; the result is a string.

$RING
Returns the ring portion of a given PVA.

$SECTION
Returns the section name and offset into the section of a given PVA.

$SEGMENT
Returns the segment portion of a given PVA.

$SIZE_BUFFER_CONTROLWARE
Returns the size of the buffer controlware for the specified channel.

$SIZE_CONTROL_STORE
Returns the size of the control store (controlware) for the specified processor.

$SIZE_REGISTER_FILE
Returns the size of the register file for the specified processor.

$SYMBOL_ADDRESS
Converts a name into a PVA by accessing the debug table.

ANALYZE_DUMP Command

Purpose Initiates the ANALYZE_DUMP utility.

Format **ANALYZE_DUMP** or
ANAD
DUMP_FILE=file
RESTART_FILE=file
DEBUG_TABLE=file or *keyword*
TITLE=string
OUTPUT=file
PROCESSING_OPTIONS=list of keyword
STATUS=status variable

Parameters *DUMP_FILE* or *DF*

Specifies the dump file to be analyzed. This file can be positioned.

If you omit this parameter, NOS/VE assumes that the file specified by the RESTART_FILE command contains the dump information. If you omit this parameter and the restart file is empty or nonexistent, an error status is returned.

RESTART_FILE or *RF*

Specifies the restart file.

The restart file is either an input file or an output file, depending on whether you specify the DUMP_FILE parameter. If you specify the DUMP_FILE parameter, a restart file is written to this file. The restart file is accessed during the execution of the ANALYZE_DUMP utility. The restart file contains a copy of the dump file, but has been reformatted to allow efficient access and a quick restart capability. If you do not specify a DUMP_FILE parameter, the RESTART_FILE parameter specifies a file created from a dump by a previous execution of the ANALYZE_DUMP utility. The default restart file is $LOCAL.RESTART_FILE.

DEBUG_TABLE or *DT*

Specifies the file that contains the debug table. This file can be positioned.

The system generates symbolic names from addresses and addresses from symbolic names. This table is built when the system is generated. It is included in the generated system and is also saved on a file. If the dump is being analyzed on the system in which it was created, the debug table in the running system can be used. Otherwise, specify the name of the file saved from the system generation. Specify the keyword NONE if no debug table is to be accessed. The default is the running system debug table.

TITLE or *T*

Specifies a string to be used in the page header if a header is generated. The string can be from 1 to 25 characters long. The default header is ANALYZE_DUMP Version 1.0.

OUTPUT or *O*

Establishes the default output file for all the subsequent ANALYZE_DUMP utility subcommands. This file can be positioned.

You can override this default with the OUTPUT parameter on each subcommand. The default is $OUTPUT.

PROCESSING_OPTIONS or *PO*

Specifies what information in the dump file is written to the restart file. You can specify the following keywords; the default is ALL:

CRITICAL

ANALYZE_DUMP reads the entire dump but constructs a restart file containing only the memory critical for analysis. Select this option to process a dump that was taken with the "All Memory" option when the "Critical memory only" option was available. If the "Critical memory only" option was not available when the dump was taken, this parameter is ignored and the dump is processed as usual.

HARDWARE or H

Only the information relating to hardware analysis is written to the restart file.

ALL

All information in the dump file is written to the restart file.

STATUS

Returns the completion status for the utility session.

Remarks

- The ANALYZE_DUMP utility does not request a tape. You must enter the REQUEST_MAGNETIC_TAPE command prior to entering the ANALYZE_DUMP command.
- When the dump resides on more than one tape volume, use the EVSN parameter of the REQUEST_MAGNETIC_TAPE command to specify each tape volume.
- Specifying PROCESSING_OPTIONS=HARDWARE significantly reduces the disk space required to hold the restart file.

Examples

- The following example assumes that the restart file is not yet formatted:

```
/request_magnetic_tape file=$local.dump ..
../type=mt9$6250 external_vsn='dmp001'
/analyze_dump dump_file=$local.dump restart_file=$user.dmp001;quit
```

- The following example assumes that you are using a restart file that is already reformatted:

```
/analyze_dump restart_file=$user.dmp001
ad/
```

Part IV: Fault Tolerance Configuration and Failure Analysis

Fault Tolerance 11

Fault Tolerance 11

Part IV, Fault Tolerance Configuration and Failure Analysis, explains how to configure NOS/VE Version 1.5.1 to maximize system availability. The information in chapters 11, 12, and 13 is intended for site analyst personnel who are responsible for:

- Defining the disk storage and magnetic tape peripheral configuration for NOS/VE.
- Establishing recovery procedures to be used in the event of an error in the operating system or a failing disk or tape unit.

These two tasks are closely related. The way in which disk peripherals are configured on NOS/VE has a significant influence on the fault tolerance abilities of the system and the recovery procedures that may be required later. For this reason, we recommend that you become familiar with the information in all three chapters before you begin either task.

Chapter 11, Fault Tolerance, describes the fault tolerance features of the NOS/VE operating system and suggests how to configure your system to make optimal use of these features. The term *fault tolerance* refers to a wide range of system features intended to minimize the negative impact of system errors and to minimize the extent of the repair effort required to recover from errors.

Chapter 12, Failure Analysis, contains a set of decision trees to help you diagnose many types of system problems and determine the appropriate repair solution.

Chapter 13, Repair Solutions, describes the repair solutions referenced in chapter 12.

Terminology

The following is a glossary of terms and definitions used in Part IV:

Active Set

Mass storage set, members of which have been active since the last deadstart. See Active Volume, Mass Storage Set.

Active Volume

Volume that has been initialized and added to a set, and is mounted on a disk unit that was in the ON state during or since the last deadstart. An active volume may be assigned to files or catalogs.

Alternate Access

Two or more channels or controllers that are physically cabled to another element and that are used interchangeably by NOS/VE to access that element. Refer to the Alternate and Redundant Path Configurations section in appendix E, Supported Hardware Products, for more information on alternate access.

Call CE

Refers to the process of informing Control Data of a hardware failure and a subsequent repair action by a customer engineer (CE).

If the system is not operational after a repair, perform a continuation deadstart. If the continuation deadstart is unsuccessful, use the information in chapter 12, Failure Analysis, to determine a course of action. To inform Control Data of a hardware failure, call the CYBER Software Support Hotline.

Catalog

Directory of permanent files and other (sub) catalogs maintained by the system on behalf of the user. The catalog describes the residence, access control, access statistics, and attributes of the files and catalogs contained in it. A catalog resides on a volume that belongs either to class J or to class L. If a catalog volume is lost, NOS/VE cannot locate the files or subcatalogs that are described by the missing catalogs.

Central Memory Reload

Process that restores central memory from an EDD dump after a mainframe repair action that required power to be shut off. Refer to the CYBER Initialization Package (CIP) Reference manual for more information on restoring central memory. Refer also to Dump NOS/VE and Express Deadstart Dump.

Continuation Deadstart

Process that returns the system to production from an unscheduled interruption or a scheduled interruption initiated by the TERMINATE_SYSTEM command.

The process restarts the NOS/VE system and optionally recovers data from central memory. If this is the first continuation deadstart following a mainframe repair action in which the contents of central memory were saved on tape, refer to Central Memory Reload.

If you have a CYBER 930 series machine, refer to the section on starting the system in the CYBER 930 Guide to Operations manual. For all other CYBER mainframe models, refer to the NOS/VE Operations manual section for continuation deadstart information.

Critical

Identifies an element in the configuration whose failure cannot be tolerated by the NOS/VE system. The effect of not tolerating the failure is that the system will go into step mode or fail. However, a critical volume other than the system device can be omitted from the configuration at a continuation deadstart. The system device is the only disk unit that cannot be omitted from the configuration at a continuation deadstart. Refer also to Critical Element and Service Critical.

Critical Display Window

Window in the system console screen used for the display of information and failure data critical to system operation and maintenance.

Critical Element

Element whose failure causes a system interruption. The critical elements of a NOS/VE system are: the mainframe, the NOS/VE operating system software, a critical volume, or any channel or controller providing sole access to a critical volume.

Critical Volume

System device, any volume containing $SYSTEM catalogs (mass storage class J in the system set), and any volume that is a member of the critical mass storage class (class Q in any set).

Dedicated Fault Tolerance (DFT)

Dedicated program that detects, isolates, analyzes, records, contains, corrects, repairs, and reports mainframe faults. DFT also reconfigures the mainframe to avoid certain faults without repairing the cause.

Dump NOS/VE

Process that saves mainframe memory on magnetic tape for use in later hardware or software failure analysis. Refer also to Express Deadstart Dump and Perform Central Memory Reload. Report software problems that cause service interruptions to Control Data.

When a problem occurs, contact Control Data either through the CYBER Software Support Hotline or by using the SOLVER database to submit a programming system report (PSR). Contact your Control Data sales representative for more information on SOLVER and for your site's SOLVER identification code and password.

If there is a VEOSxxxx fault symptom code in the system message line or in the critical display window of the system console, please record the fault symptom code and any accompanying text on the PSR.

If you have a dual-state system, there are two ways to dump NOS/VE memory. You may take an Express Deadstart Dump (EDD) or you may use the NOS or NOS/BE state to dump a failing NOS/VE system.

If you have a standalone system, you must take an EDD dump. If the system has failed but central memory is still intact, then always take an EDD dump around mainframe repair actions that require power to be shut off. The central memory may be restored during the subsequent continuation deadstart without losing data.

If you have a CYBER 930 series machine, refer to the discussion in the CYBER 930 Guide to Operations manual on saving the contents of system memories on tape. For CYBER models other than the 930 series, refer to the CYBER Initialization Package (CIP) Reference manual for more information.

If you have a dual-state system and central memory will remain intact until your next NOS/VE continuation deadstart, you may use NOS or NOS/BE to take a dump. If NOS or NOS/BE has also failed, take an EDD dump. Refer to the NOS/VE Operations manual for more information about taking a dump using NOS or NOS/BE.

End of Analysis

Either the analysis of the fault provided by this manual is inadequate or there was no fault in the system. If you suspect the NOS/VE system software is at fault, take a dump and contact Control Data. Refer to Dump NOS/VE.

Express Deadstart Dump (EDD)

Express Deadstart Dump is performed from the CYBER Test and Initialization (CTI) Utilities option after a system interruption. An EDD dump writes the contents of central memory to magnetic tape. The magnetic tape can be used later for both hardware and software failure analysis. Additionally, an EDD dump may be used to save and restore mainframe memory around a mainframe repair action. Refer also to Dump NOS/VE and Perform Central Memory Reload.

File

Collection of information referenced by a name and stored as an entry in a system or user catalog.

Installation Deadstart

Process that installs the NOS/VE system to the system device and that may optionally include the restoration of all preexisting files. Refer to the INITIALIZE_SYSTEM_DEVICE system core command in chapter 4, System Core Commands.

Mass Storage Class

Logical grouping of disk volumes to which NOS/VE assigns a subset of files or catalogs. Class assignment is described later in this chapter. NOS/VE has 26 mass storage classes. Each class has a name consisting of a single, alphabetic character.

Mass Storage Set

Collection of one or more mass storage volumes with a unique set name. Each mass storage set defined on a mainframe must have an independent set of mass storage class assignments. The use of multiple sets on a mainframe allows you to allocate permanent file space on the basis of family membership. For more information on the use of mass storage sets, see the Mass Storage Sets section in chapter 8, Permanent File Maintenance.

Missing Volume

Volume that has not been activated at the last deadstart. A volume is assumed to be missing when a catalog refers to a volume that is not known to the system. A volume may be missing because the disk unit on which it is mounted was not described in the physical configuration, because the disk unit has not been turned on since the last deadstart, because there is no channel or controller in the ON state providing access to the volume, or because the volume has been initialized.

Multiple Sets

Mainframe whose mass storage volumes define more than one mass storage set is said to have multiple sets. The multiple set feature allows the separation of families on different groups of mass storage volumes. In particular, it allows a complete separation of the family $SYSTEM from all other families. The term nonsystem set refers to a set other than the system set (the one containing the family $SYSTEM).

Nonsystem Set

In a system with multiple sets, a set other than the system set.

Pause for Operator Input

Allows operator interaction at various steps in the initialization of the system. If you have a CYBER 930 series machine, you must change the menu selection DEADSTART PAUSE FOR OPERATOR INPUT from NO to YES on the SYSTEM OPTIONS menu.

Refer to the CYBER 930 Guide to Operations manual and the NOS/VE Software Release Bulletin for more information on operator intervention. To reconfigure the NOS/VE system device or the CIP device, you must first change the menu selection NOS/VE DEADSTART COMMAND FILE LOCATION that is also on the SYSTEM OPTIONS menu.

For all other CYBER mainframe models, refer to the NOS/VE Operations manual for more information on operator intervention.

Redundant Access

Channel or controller that is physically cabled to another element but is not used to access that element until an uncorrectable fault is detected in the primary channel or controller. Refer to Alternate and Redundant Path Configurations in appendix E, Supported Hardware Products.

Service Critical

Term used to describe mass storage catalogs or files whose absence in the configuration prevents user login or denies basic system service. Service critical product files are installed on a volume belonging to mass storage class K in the system set.

In a multiple set configuration, volumes belonging to sets other than the system set and belonging to classes J and K are also considered service critical. These volumes are required to be operational for certain queued batch jobs.

The failure of a class K volume does not necessitate a system interruption for repair, but the effect of the failure may be far more severe for system service than the loss of most other volumes in the configuration.

Set

See Mass Storage Set.

State

Element such as a channel, controller, or unit has one of three states: ON, OFF, or DOWN. Refer to the description of the CHANGE_ELEMENT_STATE subcommand in chapter 3, Logical Configuration Utility, for more information.

Stepped State

State of inactivity of the NOS/VE system during which repair may be performed on any element of the system that does not require the mainframe to be powered off. The system enters this state automatically when a critical element fails or when commanded by the STEP_SYSTEM command entered in the input line of the critical display window.

System Device

Disk volume that contains the installed NOS/VE system. This may or may not be the volume that contains the CYBER Initialization Package (CIP). The location of the system device is identified in the NOS/VE Deadstart and System Device Configuration Selections menu. Refer to Pause for Operator Input for references to other manuals.

System Segment

System segment is any of the types of files listed below that are created by NOS/VE on behalf of the system's users. These files must be available to execute or to terminate tasks and jobs in the system.

- Stack segments (all rings)
- All NOS/VE system code and data segments
- Job logs
- Global logs

System Operational

System service has either been fully restored or restored to the extent that the fault and the site configuration allow. This does not necessarily conclude the analysis of the failure. You may want to inform Control Data of any software problem you encounter. Refer also to Dump NOS/VE.

System Set

Set containing the $SYSTEM family.

Unavailable Volume

Volume that was activated at or since the last deadstart, but has since become inaccessible. A volume may become inaccessible either because NOS/VE automatically stopped using the volume due to a fault or because the operator or the CE manually changed the state of an element in the configuration to OFF or DOWN.

If NOS/VE detects an uncorrectable fault in a disk unit, it places the volume mounted on the disk unit in an unavailable condition. If NOS/VE detects an uncorrectable fault in a controller or channel, all volumes connected to the failing element are placed in an unavailable condition unless there is an alternate access to the volumes.

Unreconciled

File or catalog that is either not described by a parent catalog or does not exist on the disk volume indicated by the parental catalog.

An unreconciled file or catalog could be one that had been created since the last catalog backup and whose parental catalog had been restored. An unreconciled file could also result from restoring a file that resided on a missing or unavailable volume prior to reinstating the volume; the version of the file on the reinstated volume is extraneous because it is no longer described by a catalog. A system software defect could also cause a file or catalog to be unreconciled. Refer to the description of the system attribute DELETE_UNRECONCILED_FILES in the NOS/VE System Performance and Maintenance manual, Volume 1.

Unstep

Changing the state of the system from a stepped condition to that of normal system service.

Volume

Magnetic recording surface or surfaces on which data is recorded. A volume may belong to one and only one set.

Production Capabilities

This section explains how to configure and operate NOS/VE to maintain system service. By following the recommendations in this section, you enable the system to tolerate the majority of disk hardware faults and to minimize the downtime for faults that cannot be tolerated.

Upgrading to NOS/VE Version 1.5.1, L739

The following are important points to remember as you upgrade to version 1.5.1 at PSR level 739. The details of performing the upgrade to version 1.5.1 are documented in the NOS/VE Installation Handbook.

- The mass storage attributes defined by a REQUEST_MASS_STORAGE command or RMP$REQUEST_MASS_STORAGE program interface are now reinstated when files are restored by the RESTORE_PERMANENT_FILES utility. The attributes of files created by the system installation and upgrade processes, by IM/DM, and by other applications that use REQUEST_MASS_STORAGE are also affected.

  For NOS/VE Version 1.5.1, L739, mass storage attributes are included in the catalog information recorded by the BACKUP_PERMANENT_FILES utility. As a result, the ALLOCATION_SIZE, FILE_CLASS, INITIAL_VOLUME, and VOLUME_OVERFLOW_ALLOWED attribute selections are now reinstated by the RESTORE_PERMANENT_FILES utility when restoring from a version 1.5.1 backup.

  The reinstatement of an attribute depends on who is doing the restoration; restoration of an attribute is governed by the same rules defined for the REQUEST_MASS_STORAGE command. If the person performing the restoration has insufficient privilege to restore the attribute, the file is restored without the attribute value and a warning message is written to the RESPF listing.

  If a system administrator performs the restoration and the INITIAL_VOLUME attribute specifies a volume that is no longer in the configuration, the specification is ignored and a warning message is written to the RESPF listing.

  If a user performs the restoration and the INITIAL_VOLUME attribute specifies a volume that is no longer in the set to which the user is assigned, the specification is ignored and a warning message is written to the RESPF listing.

  Specification of attribute values using the RESPF subcommand SET_RESTORE_OPTIONS takes precedence over values maintained in the backup file.

- At NOS/VE Version 1.5.1, a job that has executed a RESERVE_RESOURCE command, REQUEST_MAGNETIC_TAPE command or RMP$REQUEST_TAPE program request is recovered by active job recovery, provided that there are no tape units currently assigned to the job.

  For previous versions of NOS/VE, a job that had executed a RESERVE_RESOURCE command, REQUEST_MAGNETIC_TAPE command, or RMP$REQUEST_TAPE program request was automatically considered unrecoverable by active job recovery, regardless of whether or not the job had a tape unit assigned to it at the moment of a system interruption.

Coupled with the magnetic tape fault recovery features provided by version 1.4.1 of NOS/VE, the active job recovery improvement allows the site to better manage its tape unit resources. For example, assume the site has a background tape job that takes weeks to run and that runs during non-prime hours (nights or weekends). Also assume that the job only uses one tape unit at a time. In version 1.5.1, it is possible to force the job into giving up its tape unit at the switchover to prime time by doing the following:

1. Make the job's tape unit not ready and select the tape error recovery option to request another mount for the current tape volume. Because the tape unit is not ready, the job will not be able to continue even though you have responded to the failure recovery menu.

   Alternatively, you may wait until the job finishes with the current volume and decline to mount a subsequent volume.

   The job is now in a state where it is not using a tape unit and is therefore recoverable by active job recovery. Because the job uses only one tape unit at a time, it has no tape units reserved to it. Therefore, all tape units are available for use during prime time hours.

2. Enter the SOU command SWAP_OUT_JOB.

   The job's mount request will continue to be displayed in the TAPE_MOUNT VEDISPLAY. Ignore the request until nonprime hours.

3. When it is time to continue the job, enter the SOU command SWAP_IN_JOB.

4. Remount the volume on a tape unit and assign the tape unit to the job, if necessary.

   NOTE

   It is also possible to apply the previous example to a job that uses the RESERVE_RESOURCE command to schedule the use of multiple tape units. However, during prime hours the tape units remain reserved to the swapped job. Although the "reserved" tape units may be used by other jobs, a job that requires more than one tape unit at a time might be blocked, depending on the number of tape units configured and the number required by each job.

- The permanent file maintenance procedures are updated to allow execution within the system job. This is particularly important in situations where service-critical files or catalogs are missing or unavailable, consequently preventing batch job submittal.

- Restrictions on the naming of recorded VSNs of magnetic tape volumes are removed: the VSN prefix can now be of type string, as well as types name and integer. These types are supported by all of the permanent file maintenance procedures and by the process that reinitializes the system device.

Fault Tolerance Features

The fault tolerance features in this release are summarized in the following four groups:

- Configuration dependent features
- Other disk fault tolerance features
- Disk volume defect management features
- Magnetic tape fault tolerance features

Configuration-Dependent Features

These features require critical files to be partitioned from noncritical files to take advantage of the feature. Refer to the Configuration Recommendations and Site Preparation sections later in this chapter for more information on configuring volumes.

- Multiple mass storage sets can be used to allow a mainframe to provide backup access to another mainframe's permanent file base. For more information, refer to Restoring Permanent Files by Sets in chapter 8, Permanent File Maintenance.
- Replacement or reinitialization of the system device's volume may be accomplished during a deadstart without the loss of files or catalogs belonging to users other than $SYSTEM on other volumes.
- $SYSTEM files, catalogs, and subcatalogs can be physically isolated from those of other users; this improves the ability of the system to tolerate the loss of volumes that do not contain these files.
- If a critical volume belonging to class Q or J becomes unavailable due to the failure of a channel, controller, or disk unit, the system automatically places itself in a stepped condition. While the system is in a stepped state, you may perform repair and then enter the UNSTEP_SYSTEM command to continue service.
- If you have two controllers connected to a $9836_1 or $9853_x unit and the active controller fails, you can manually down the faulty controller using the LCU subcommand CHANGE_ELEMENT_STATE. NOS/VE will automatically reconfigure to the redundant controller if one is defined in the physical configuration. For more information, refer to the description of the CHANGE_ELEMENT_STATE LCU subcommand in the Logical Configuration Utility chapter of this manual.
- If you have redundant channels connected to the any of the controllers listed below and the active channel fails, you may manually down the faulty channel using the LCU subcommand CHANGE_ELEMENT_STATE. NOS/VE will automatically reconfigure to a redundant channel if one is defined in the physical configuration. The channel chosen is the first one in the controller's IOU_CONNECTIONS list that is in the ON state. For more information, refer to the description of the DEFINE_ELEMENT PCU subcommand in the Physical Configuration Utility chapter of this manual.

$7155_1
$7155_1x
$10395_11
$FA7B4_D
$FA7B5_A

- If a noncritical volume becomes unavailable due to the failure of a channel, controller, or disk unit, the system places the tasks referencing the volume into a waiting state until the volume is repaired and made available again. Tasks executing in the system job are not placed in a waiting state and may automatically restart or may require manual restarting.

- It is possible to perform online repair and reinstatement of any noncritical element without a system interruption. However, a system interruption is required to repair the system device or to initialize an active volume.

- You can omit any volume except the system device from the configuration at a continuation deadstart; however, you may need to restore files or catalogs in order to use the system.

- If a noncritical volume is damaged or you cannot wait for repair of a disk unit, you may restore files to remaining volumes without requiring a system interruption. Refer to the RESTORE_UNRECONCILED_FILES command in chapter 8, Permanent File Maintenance, and Restoring Files in chapter 13, Repair Solutions, for more information on restoring files.

- If a catalog volume becomes unavailable, catalogs can be restored to another volume without losing files and catalogs described by the catalog backup.

  One system interruption is required to restore catalogs.

  You may restore catalogs from the most recent catalog-only backup or the most recent partial backup or full backup that contains catalogs. Refer to the RESTORE_UNRECONCILED_CATALOGS command in chapter 8, Permanent File Maintenance, and Restoring Catalogs in chapter 13, Repair Solutions, for more information on catalog restoration. Refer to Site Preparation later in this chapter for more information on catalog backups.

- HPA/VE provides online failure analysis.

- MALET/VE provides online diagnosis of a failure concurrent with customer use of the system.

Other Disk Fault Tolerance Features

The remaining fault tolerance features do not require critical files to be partitioned from noncritical files to take advantage of the feature. The features are provided to maintain system production when a disk channel, controller, or unit fails. See Disk Volume Defect Management Features later in this chapter for a discussion of features that allow you to cope with failures on the recording surface of a disk unit.

- When you are performing a continuation deadstart, you may specify whether or not unreconciled files and catalogs are deleted. This option allows a disk unit to be downed without losing files and catalogs. By default, unreconciled files and catalogs are not deleted. Refer to the NOS/VE System Performance and Maintenance manual, Volume 1, for more information about the system attribute DELETE_UNRECONCILED_FILES.

- You may back up catalogs separately from files by specifying ALL for the parameter NUMBER_OF_CYCLES in a backup using the BACKUP_PERMANENT_FILES utility subcommand EXCLUDE_HIGHEST_CYCLES. This allows catalogs to be backed up more frequently and with less expense than performing a partial backup. Refer to chapter 8, Permanent File Maintenance, for more information on the EXCLUDE_HIGHEST_CYCLES subcommand. Refer to the CREATE_CATALOG_BACKUP command in the NOS/VE Operations manual for more information on catalog backups.

- You may backup files without having to include catalog information on your full or partial backups. This can save time and capacity in performing those backups.

  NOTE

  If you exclude catalog information from your full or partial backups, you should only do so because you are backing up catalogs separately. Refer to the description of the BACKUP_CATALOGS parameter of the CREATE_FULL_BACKUP and CREATE_PARTIAL_BACKUP commands in the NOS/VE Operations manual.

- If a mass storage class has no more members with available space, the action taken by NOS/VE depends upon the mass storage class. When class Q has no more space, NOS/VE automatically attempts to add another volume to this class. If there are no more volumes to add to class Q, NOS/VE places the task requesting the space into a waiting state until space becomes available. As a result, the system may no longer be operational.

  When a class other than Q is out of space, the task is put into a waiting state until space becomes available. Refer to repair solution 6 in chapter 13, Repair Solutions, for more information about this process.

- If a volume has been added to class Q or a class other than Q is out of space, NOS/VE automatically selects the VEDISPLAY MASS_STORAGE (VED MS) display at the system console to force the attention of the operator to this problem.

- Users may status their job by pressing the network control character and S (that is, %S or ESC S) to determine why their task is blocked. A task waiting for a hardware repair or an out-of-space condition is denoted in the message. Users may enter terminate break, if they do not want to wait.

  The operator may use the DISPLAY_JOB_STATUS (DISJS) command to see which jobs are blocked by a failure. The operator may use the VED MS display to determine which volumes are unavailable or out of space.

  NOTE

  If a task is waiting for an unavailable system segment, the user will not be able to check the job status or terminate the task.

  If a user does a pause break and enters the DISPLAY_JOB_STATUS command rather than using %S, the message displayed indicates that DISJS was entered rather than indicating the waiting condition.

- You can place files on a specific disk volume or on a specific mass storage class by using the REQUEST_MASS_STORAGE command or the SET_RESTORE_OPTIONS subcommand of the RESTORE_PERMANENT_FILES utility.

- All lines that are recorded on the critical display window of the system console are time-stamped.

- NOS/VE provides a confidence test for all mass storage subsystems. Failure data is stored in an error log. For more information on failure data, refer to the HPA/VE Reference manual. Each actively configured channel, controller, and unit is tested. A redundant channel or controller access is not tested; refer to appendix E, Supported Hardware Products, for more information about which channels may be considered redundant.

  Data is looped through each controller's buffer memory and through a reserved cylinder on each volume. The test is run during each deadstart, when you unstep the system, and whenever a channel, controller, or unit is turned back on using the LCU subcommand CHANGE_ELEMENT_STATE.

- For 887, 9836, and 9853 subsystems, NOS/VE provides a confidence test of each actively configured channel, controller, and disk unit; a redundant channel or controller access is not tested. Refer to Alternate and Redundant Path Configurations in appendix E, Supported Hardware Products, for more information about which channels may be considered redundant. Data is looped through each controller's buffer memory and through a reserved cylinder on each disk unit. The test is run during each deadstart, at UNSTEP_SYSTEM (if volume is reenabled), and whenever a channel, controller, or disk unit is turned back on using the LCU subcommand CHANGE_ELEMENT_STATE. Refer to chapter 3, Logical Configuration Utility, for more information on the CHANGE_ELEMENT_STATE subcommand.

- For 834, 836, 887, 9836, and 9853 subsystems, NOS/VE provides online diagnosis of failures by automatically invoking subsystem diagnostics if disk retries are unsuccessful. The diagnostic result is reported to the critical display window of the system console as described in Failure Data in Critical Display Window in chapter 12, Failure Analysis.

- It is possible to save the contents of central memory before a mainframe repair action and to restore memory following the repair action. This may allow you to recover the data for files that were being modified and that were not written to disk prior to the mainframe fault. To save memory, take an EDD dump. Refer to Dump NOS/VE and Perform Central Memory Reload in the Terminology section of this chapter for more information on saving central memory contents.

- It is possible to add a former set member back to its original set without losing the volume's files or catalogs. If the state of a disk unit is changed to DOWN or OFF during a continuation deadstart, you are asked whether you want to remove the volume mounted on the unit from its set. If you choose to remove the volume from its set for some reason and then decide you made a mistake, you may add the volume back to the set using the LCU subcommand ADD_VOLUME_TO_SET. Refer to chapter 3, Logical Configuration Utility, for more information on the ADD_VOLUME_TO_SET subcommand. You may add the volume back to the set without losing its contents only if you do not delete unreconciled files during a continuation deadstart or do not restore the volume's files or catalogs first.

- If a unit cannot be accessed during a continuation deadstart, the operator is informed and is given the option of fixing the problem or continuing without the volume. If the operator chooses to continue without the volume, the unit is automatically downed by NOS/VE. If the volume that could not be activated is the only member of a particular mass storage class, the operator is requested to assign some other volume to this class to complete the deadstart. If the system device fails during deadstart, you must either repair the system device or create a new system device. Refer to repair solution 3 in chapter 13, Repair Solutions, for more information on volume assignment.

Disk Volume Defect Management Features

The following features allow the site to manage disk volume defects that may occur after a volume has been formatted and initialized. Refer to Volume Defect Management Recommendations in appendix E, Supported Hardware Products, for more information on defect management.

- At a continuation deadstart, NOS/VE moves a catalog from a volume if the volume is no longer a member of the catalog class (J or L) that is required for the catalog and if active set validation is selected. You may want to move catalogs if a correctable media defect on a catalog volume is a concern. For more information, refer to Repair Solution 6 in chapter 13, Repair Solutions, and to the description of the VALIDATE_ACTIVE_SETS system attribute in the System Performance and Maintenance manual, Volume 1.

- The LCU subcommand INITIALIZE_MS_VOLUME and the system core command INITIALIZE_SYSTEM_DEVICE have an option to retain mass storage flaws during initialization of a volume. Refer to chapter 4, System Core Commands, for more information on the INITIALIZE_SYSTEM_DEVICE command. Product-dependent flawing capabilities that are provided in controllers or disk units are not used by NOS/VE. If flaw retention is selected when initializing an 895 volume, the 5-minute reformatting (soft sectoring) is not performed.

- Once a volume has been initialized using the INITIALIZE_MS_VOLUME subcommand, we recommend that you not reformat it by using offline maintenance tools, by changing hardware switch settings, or by entering commands using a maintenance panel. Rather than reformatting the volume, we recommend that you flaw a media defect. This will save your files and your time. Refer to the discussion of setting flaws that follows.

- Reformatting a volume destroys any flaw information recorded on the volume by NOS/VE. If you do reformat a volume, you must initialize the volume again using the INITIALIZE_MS_VOLUME subcommand, add the volume to a set using the LCU subcommand ADD_VOLUME_TO_SET, and restore the volume's files. Remember to specify FALSE for the RETAIN_DEVICE_FLAWS parameter of the INITIALIZE_MS_VOLUME subcommand (or INITIALIZE_SYSTEM_DEVICE) when initializing a volume after reformatting it.

- You can manually define, remove, and display media flaws on disk volumes using the LCU subcommands DEFINE_MS_FLAW, REMOVE_MS_FLAW, and DISPLAY_MS_FLAWS. Because NOS/VE automatically flaws an uncorrectable media defect, the DEFINE_MS_FLAW subcommand is provided to flaw a correctable media defect in one sector. Refer to the discussion of media management in chapter 3, Logical Configuration Utility, for more information on media flaws.

  When flawed in this manner, one DAU is rendered unusable. This is implemented for all disk products supported by NOS/VE. Flaws can be defined before a volume is initialized or activated. Flaws can be removed once a volume becomes active.

  Flaws may be displayed in tabular format or in command source format. Specifying DISPLAY_MS_FLAWS DISPLAY_OPTION=SOURCE creates a file containing the DEFINE_MS_FLAW subcommands corresponding to the defects currently flawed for a volume. You may place these subcommands in your configuration prolog to ensure that the defects are flawed if it is necessary to initialize the volume without selecting flaw retention. To remove CIP from a volume, you must first release CIP (from the CTI Manual Options display), then initialize the volume and specify RETAIN_DEVICE_FLAWS=FALSE. Refer to the system core command INITIALIZE_SYSTEM_DEVICE and the LCU subcommand INITIALIZE_MS_VOLUME for more information.

- If an uncorrectable media defect occurs during access to a sequential or byte-addressable file, abnormal status is returned to the GET or PUT request, giving the program the choice of whether to terminate the task or take some other action. Also, if an uncorrectable media defect occurs during execution of a COPY_FILE command, the command terminates with abnormal status.

- If an uncorrectable media defect is detected during a write to mass storage, NOS/VE automatically flaws the defect and the data is automatically relocated to a different allocation on the same volume.

- If a correctable media defect is manually flawed and the defective space was previously assigned to a file, the data is automatically relocated to a different allocation on the same volume when the file is first attached after the flaw is defined.

- You can define a flaw prior to initializing a volume. The DEFINE_MS_FLAW system core command and the LCU subcommand by the same name allow a flaw to be defined prior to initializing the system volume or some other volume, respectively. Refer to the INITIALIZE_SYSTEM_DEVICE system core command and the INITIALIZE_MS_VOLUME LCU subcommand. Defining a flaw prior to initializing a volume allows you to prevent the defective sector from being assigned to system files located on the volume.

- You can determine the names of permanent files that may have been assigned to disk sectors that are experiencing an uncorrectable media defect. Optionally, the BACKUP_PERMANENT_FILES utility reads the data of every file backed up to the file $NULL. Any files with uncorrectable media defects are reported in the backup listing. Refer to the NULL_BACKUP_FILE_OPTION parameter of the BACKUP_PERMANENT_FILES utility subcommand SET_BACKUP_OPTIONS in chapter 8, Permanent File Maintenance.

Magnetic Tape Fault Tolerance Features

The following are magnetic tape fault tolerance features:

- NOS/VE automatically changes the state of a magnetic tape unit to DOWN if a hardware failure is detected during the process of reading volume labels for automatic volume assignment. If the request to read the label fails, regardless of whether the tape is in fact labelled, the state of the magnetic tape storage unit is changed to DOWN. The unit's state change is reported to the critical display window of the system console.

- If NOS/VE detects a parity error during the process of reading volume labels for automatic volume assignment, the tape is declared unlabelled. The VED TAPE_STATUS display indicates a "Ready/Read Error" in the unit status column of the display. The volume can be manually assigned to a user requesting an unlabelled tape.

- If NOS/VE detects a parity error at loadpoint while writing a tape volume, the operator is informed and is permitted to substitute a new tape volume or move the previous volume to a different tape unit.

- For a $5698_1x tape subsystem, a confidence test is run for each controller and unit on a channel. A redundant channel or controller access is not tested. Refer to appendix E, Supported Hardware Products, for more information on which channels may be considered redundant.

  For a controller, the internal diagnostics are executed and data is looped through the controller's buffer memory. For a tape unit, the internal diagnostics are executed and data is looped through the unit without accessing any media mounted on the unit. Failure data is stored in an error log. For more information on failure data, refer to the HPA/VE Reference manual.

  The confidence tests are run after a deadstart, after entry of an UNSTEP_SYSTEM command, and whenever a channel, controller, or unit is turned back on using the LCU subcommand CHANGE_ELEMENT_STATE. The testing of a $698_3x unit is deferred until the unit is first accessed after one of these events.

  Diagnostics for a controller or a unit are also executed immediately after the occurrence of a nonmedia fault in the element.

Maintenance of Noncritical Elements

You can repair any noncritical element online and reinstate the element to the configuration without a system interruption. You must place the faulty element in the DOWN state to be repaired. An element that is down may undergo any online repair action that does not require the mainframe's power to be shut off and does not damage the data on the volume.

The LCU subcommand CHANGE_ELEMENT_STATE (CHAES) may be used to change the state of a noncritical disk unit, controller, or channel to DOWN. A channel or controller may be downed only if access to a critical volume is not prevented.

Any noncritical disk controller or disk unit may be downed during a continuation deadstart. After the deadstart, the element may be repaired and reinstated without incurring another interruption.

Whenever a disk unit's state is changed to DOWN or OFF, the volume mounted on the device is no longer available to NOS/VE. It may be necessary to add one or more mass storage classes that were unique to the unavailable volume to other volumes to sustain system service. NOS/VE alerts you to this requirement as you attempt to quit the LCU after making the state change.

Production Impact

The following production impact occurs when repairs are made with or without system interruption:

Impact of Repair without System Interruption

NOS/VE can operate indefinitely with one or more unavailable noncritical volumes. An unavailable volume is one that has been deactivated since the last deadstart was performed. An unavailable volume may be reinstated without losing active jobs, files, or catalogs as long as the reinstatement occurs prior to a system termination or interruption.

If a class C volume becomes unavailable, jobs that have been swapped to mass storage and whose image has been removed from memory cannot be continued until the volume on which the job image resides is reinstated. A job that is swapped but whose image remains in memory may continue to execute; if the job must be swapped again before the volume is reinstated, the swap will occur to a different volume. Any new swap files are assigned to other volumes belonging to class C.

If a class K volume becomes unavailable, the effect of the failure upon system service may be far more severe than the loss of most other volumes in the configuration. It may prevent users from logging in to NOS/VE or using basic commands such as EDIT_FILE, PRINT_FILE, and so forth.

If a class L volume becomes unavailable, it may prevent many or all users from productively using the system. If there is only one class L catalog volume, then all users other than $SYSTEM may not be able to reference permanent files. If there are multiple class L catalog volumes, only those users whose master catalog is on an available catalog volume may be unaffected. However, subcatalogs of the master catalog may be on the unavailable volume. It is necessary to terminate and restart the system before catalogs may be restored.

If a class M volume becomes unavailable, only jobs that have files attached on that volume or that attempt an attachment of a file on that volume are affected. If a file cycle that resides on an unavailable volume is restored, the old version is deleted from its catalog but remains defined until all jobs to which it was originally attached have terminated. Modifications made to the old version are lost, however.

Subsequent to the file restoration, any job that attaches the file cycle attaches the newly restored version. If a file cycle that resides on an unavailable volume is not restored before the volume becomes available, then no data is lost.

NOTE

If you restore files, it is necessary to perform a continuation deadstart later when all volumes are available to regain the mass storage space that the original files occupied. Set the DELETE_UNRECONCILED_FILES system attribute to 1 at the deadstart to reclaim the space. Refer to Restoring Files in chapter 13, Repair Solutions, for more information.

If a class N volume becomes unavailable, only jobs that have their temporary files on the volume are affected. After the volume becomes unavailable, jobs that are started have their files assigned to other members of class N.

Interactive users affected by an unavailable class N volume may enter a DETACH_JOB command and log in again. They may enter an ATTACH_JOB command to recover their work when the volume is reinstated.

Any job that is affected by an unavailable class K, L, M, or N volume is blocked until the volume becomes available. An interactive user may check the job status using the network control character and S (that is, %S or ESC S) to determine that the job is waiting on an unavailable volume. The user may enter terminate break or continue to wait. However, a job that has attached a file on an unavailable class K, M, or N volume is not allowed to terminate until the volume is reinstated and the file can be detached.

NOTE

If the user does a pause break and enters the DISPLAY_JOB_STATUS command rather than using %S, the message displayed indicates that DISJS was entered rather than indicating an unavailable volume.

If a task is waiting for an unavailable system segment, the user will not be able to check the job status or terminate the task. System segments are assigned to disk volumes belonging to class N. If a class N volume is unavailable, there may be jobs in this predicament.

Permanent file cycles residing on an unavailable volume remain defined until one of the following occurs:

- The file cycle is deleted by a user.
- The file cycle is restored using the RESTORE_EXCLUDED_FILE_CYCLES subcommand of the RESTORE_PERMANENT_FILES utility.
- A continuation deadstart is performed with the value 1 specified for the system attribute DELETE_UNRECONCILED_FILES.

A user can determine how many cycles are unavailable by entering the DISPLAY_CATALOG command to display the master catalog. For example, the user may enter the following:

```
display_catalog catalog=$user display_option=contents depth=all
  include_exception_conditions=all
```

Each cycle that is unavailable is identified by the message:

```
-- volume unavailable
```

NOTE

If at all possible, repair the fault that caused the volume to become unavailable before terminating the system. A planned or unplanned termination of the system prior to reinstating the unavailable volumes has the following consequences:

- All active jobs are lost.
- The data of files that were being modified at the time of the fault is lost. Refer to Finding Inconsistent Files in chapter 13, Repair Solutions, for more information.

Impact of Repair with System Interruption

The state of any noncritical disk unit may be changed to DOWN or OFF during a continuation deadstart. A volume mounted on a disk unit that is not in the ON state during the deadstart is referred to as a missing volume and a file on such a volume is called a missing file.

Any unwritten data belonging to a missing file is lost as a result of the continuation deadstart. Any active jobs that had attached files or catalogs on the missing volume are not recovered.

If a class J, K, L, or M volume is missing, a reference to the missing file or catalog from a terminal other than the system console is blocked until the volume is reinstated. An interactive user may status the job using the network control character and S (that is, %S or ESC S) to determine that the job is waiting on a missing volume. The user may terminate break or continue to wait.

NOTE

If the user does a pause break and enters the DISPLAY_JOB_STATUS command rather than using %S, the message displayed indicates that DISJS was entered rather than indicating a missing volume.

If a class Q or N volume is missing, no jobs that begin after the completion of the continuation deadstart are affected.

A missing volume becomes available again when the disk unit on which the volume is mounted is turned back on.

You can restore any missing file or catalog. Any missing file or catalog that is not restored is recovered when the missing volume is reinstated.

NOTE

If you restore files or catalogs, you must perform a continuation deadstart later when all volumes become available to regain the mass storage space that the original files occupied. Set the DELETE_UNRECONCILED_FILES system attribute to 1 at the deadstart to reclaim this space. Refer to the System Performance and Maintenance manual, Volume 1, for more information on the DELETE_UNRECONCILED_FILES system attribute. Refer to Restoring Files in chapter 13, Repair Solutions, for more information on performing a continuation deadstart.

Permanent file cycles residing on a missing volume remain defined until one of the following occurs:

- The file cycle is deleted by a user.
- The file is deleted by the site analyst, family administrator, or a user with the DELETE_UNRECONCILED_FILES parameter of the CHANGE_CATALOG_CONTENTS command. Refer to the NOS/VE Operations manual for more information on the CHANGE_CATALOG_CONTENTS command.
- The file is restored using the RESTORE_EXCLUDED_FILE_CYCLES subcommand of the RESTORE_PERMANENT_FILES utility.
- A continuation deadstart is performed with the value 1 specified for the system attribute DELETE_UNRECONCILED_FILES.

A user can determine how many cycles are missing by entering the DISPLAY_CATALOG command to display the master catalog.

```
display_catalog catalog=$user display_option=contents depth=all
  include_exception_conditions=all
```

Each cycle that is missing is identified by the message:

```
-- media missing
```

NOTE

It is not possible to delete a catalog that resides on a missing volume.

Maintenance of Critical Elements

A mainframe element must be repaired offline or while the system is in the stepped state.

A critical disk element may be repaired while the system is in the stepped condition and reinstated by entering the UNSTEP_SYSTEM command in the input line of the critical display window.

A controller or unit providing access to a $SYSTEM (class J) catalog volume may be downed at a continuation deadstart, repaired online, and reinstated without another deadstart as long as access to the system device is maintained throughout. $SYSTEM catalogs must be restored after the deadstart.

The system device may be downed and a new system device may be initialized without having to initialize other set members. Refer to the RECOVER_SYSTEM_SET parameter of the system core command INITIALIZE_SYSTEM_DEVICE. You must restore $SYSTEM catalogs and files after the deadstart.

Configuration Recommendations

The key to configuring the NOS/VE system for maximum availability is mass storage classes. By properly assigning the mass storage volumes to classes, the site categorizes its volumes into two broad classes: critical and noncritical. Refer to the the description of the LCU subcommand CHANGE_MS_CLASS in chapter 3, Logical Configuration Utility, for more information.

The following guidelines are provided for configuring NOS/VE for reasonable performance and greatest system availability (fault tolerance) using mass storage classes. Individual sites may require a different configuration.

If you have fewer than eight disk volumes, configuring for both fault tolerance and performance may be difficult. The loss of one disk unit may so severely affect performance and storage capacity that continuing service in a degraded mode is not possible. If this is likely to be the case at your site, assign all volumes to all classes (the default).

However, if disk capacity is not a factor, you can configure a good fault tolerance configuration with as few as five volumes. Configure the five volumes as described for volumes 1 to 5 of the Eight-Volume Target Configuration section later in this chapter.

The following recommendations will improve the system's ability to tolerate disk hardware faults and will improve system availability if a fault is not tolerable:

1. Configure as few volumes as possible to classes J and Q. This is to your advantage because the unavailability of any class J volume (in the system set) or any class Q volume (in any set) causes the system to automatically enter the stepped state.

   We recommend that only the system device belong to class Q. If the demand for class Q space exceeds the capacity of the system device, NOS/VE automatically picks an additional volume and assigns it to class Q to maintain service. The operator is informed if this occurs.

   In a multiple set system, we recommend that only one volume belong to class J in each set. The capacity required for $SYSTEM catalogs in any set is small; more than one volume should never be required.

2. Configure a redundant channel to each $7155_1, $7155_1x, $FA7B4_D (834 and 836), $FA7B5_A (9836 and 9853) controller. If the primary channel fails, you may down it and NOS/VE automatically configures to use the redundant channel. Refer to appendix E, Supported Hardware Products, for more information on redundant channels.

3. Configure redundant controllers to each 9836 and 9853 unit. If the primary controller fails, you may down it and NOS/VE automatically configures to use the redundant controller. Refer to appendix E, Supported Hardware Products, for more information about redundant channels.

4. Configure all class Q and J volumes on the same channel(s). NOS/VE automatically steps the system if any class Q or J volume becomes unavailable. Therefore, if class Q or J volumes were on every channel in the configuration, the failure of any channel or controller could take the system into the stepped state.

5. Install CIP on the NOS/VE system device in a standalone system. Since the system device must always be on and available, this guarantees that Dedicated Fault Tolerance (DFT) can always access the CIP device during NOS/VE execution. Note that there is no CIP device if your primary IOU is an I4C.

6. We recommend that volumes be assigned to classes such that:

   - Each mass storage class has at least one member volume that does not belong to any other mass storage class. Specifically, this recommendation applies to classes C, J, K, L, M, N and Q, as these are the classes to which NOS/VE currently directs subsets of catalogs and files. However, you can apply this recommendation to all classses.

   - Volumes belonging to mass storage classes C, J, K, L, and Q do not belong to any class other than A.

   - If your site uses classes U, V, W, X, Y, or Z to allow some users to place permanent or temporary files on these volumes while not allowing other users to do so, remove classes M and N from the volumes that belong to classes U to Z.

   The preceding recommendations make it impossible for an out-of-space condition with one class to induce an out-of-space condition in another class. If you do not have enough disk capacity to follow these recommendations, try the following:

   - Give each volume a unique combination of class memberships. Refer to the Twelve-Volume Optimum Availability Configuration examples later in this chapter.

   - If you have multiple sets configured and you need more class M space in the system set, you may add class M to your class K volume. Likewise, if you need more class K space, you may add class K to your class M volumes.

   - If you have multiple sets configured, you may add class J to one of your class L volumes in any nonsystem set. In a family other than $SYSTEM, there is a negligible amount of space required for catalogs belonging to the user $SYSTEM in the family. Alternatively, you may add class N to your class J volume.

7. Configure as few user catalog (class L) volumes as possible. Ideally, configure one class L volume.

8. Separate the class J volume from the class L catalog volume(s); the latter can be repaired online. In a multiple set system, it is not necessary to separate class J from class L in a nonsystem set. The unavailability of a class J volume in a nonsystem set is not critical to the system.

9. Separate catalog volumes from permanent file volumes. If you lose a catalog volume (class J or L), you may be able to resume service more quickly if you only have to restore catalogs and not files. This is particularly true if you backup catalogs separately from files.

10. Separate service critical $SYSTEM permanent files (class K volumes) from those containing files for other families (classes L and M) and from temporary file (class N) volumes. Although members of these classes may be maintained without a system interruption, separating the class K volumes may allow class L, M, and N volumes to fail without impacting all users and also allows a backup of class K files by volume to be minimal.

In a multiple set system, $SYSTEM permanent files are inherently separated from files and catalogs of other families. Therefore, within each set there is less concern about separating class K from other classes. In a nonsystem set, class K is only used for queued files and validation files belonging to $SYSTEM; none of these class K files are considered critical to families in other sets (including the system set).

11. Configure more than one volume for classes C, K, M, and N. The disk units on which these volumes are mounted can be repaired without a system interruption, but having at least two volumes of each class improves the likelihood that some users will not be affected by the loss of a single volume.

12. In a single set system, configure no more than two volumes belonging to class K. The odds of both units failing concurrently are small; however, the more units there are, the more likely one will fail.

    In a multiple set system, configure only one class K volume per set, if one volume provides sufficient capacity.

13. Designate one disk unit in your configuration for use as a replacement for the system device or the $SYSTEM catalog device. The volume mounted on the designated device should not belong to any of the following classes: J, K, L, and M. Following the latter recommendation minimizes the file and catalog restoration time required to continue production if the system device or the disk unit containing a $SYSTEM catalog volume fails.

14. Configure your system with multiple sets if you want to take advantage of any of the following features:

Feature	Description
Fault containment	A missing or unavailable volume only affects the families whose files are in the set of which the volume is a member.
File containment	All of the users in the families assigned to a set are constrained to use only the space available to the set. The family $SYSTEM may be separated from other families in this way.
Family portability	In a multiple mainframe environment, if the volumes in a set are physically accessible to two (or more) mainframes, the set may be logically switched to a surviving mainframe without having to redeadstart the surviving mainframe. However, removing a set from a mainframe requires an interruption.

NOTE

To allow a mainframe to provide access to another mainframe's file base, each mainframe must be configured so it can access the volumes comprising a set on the other mainframe. If access is not available, a physical recabling of peripherals is required to allow access.

The system set must have at least classes C, J, K, M, and Q assigned to it. Class M contains nonservice-critical $SYSTEM files.

Nonsystem sets must have at least classes J, K, L, and M assigned to them. Class J contains the catalogs for the user $SYSTEM in each family assigned to the set; class K consists of files for the user $SYSTEM in each family assigned to the set and queued input files belonging to the families in the set.

The critical class (class Q) should belong to only the system device or other members of the system set.

The temporary class (class N) may belong to any set or may be omitted from any particular set as long as there are sufficient volumes of this class that belong to another set. For more information about mass storage classes in a multiple set environment, refer to chapter 3 of this manual.

NOTE

The following recommendations may improve your system performance while also configuring for fault tolerance and availability.

15. If dedicating one volume each for $SYSTEM (class J) and user catalogs (class L) wastes space, add class K or class N to your class J catalog volumes and class N to your class L catalog volumes for improved capacity utilization. However, ensure that there are other members of class K and N that do not also belong to classes J or L, respectively. If you add class K to a class J volume, we recommend that you backup $SYSTEM files and catalogs separately from user files and catalogs to minimize restoration time if the volume is lost. See recommendation 9 for more information on separating catalogs from permanent file volumes.

16. If you have multiple disk products in your configuration, put catalogs on the product that best matches your capacity and speed requirements. Refer to appendix E, Supported Hardware Products, for more information about disk product characteristics.

    In the following table, the disk products are sorted by increasing page-fault speed. The second column indicates the maximum number of catalogs (each of minimum size) that may be placed on one volume in the configuration. The actual number depends upon the size of the catalogs you create.

Product	**Maximum Number of Catalogs Per Volume**
834	8,121
844	8,850
836	24,400
885	33,007
9836	25,170
895	32,604
9853	65,180
887	33,411

17. Assign the swap file class (class C) to the disk units with the fastest job swap performance in a configuration that has multiple disk products configured. In this environment, choose the class C volumes in the following order of preference: 887, 9853, 895, 9836, 836, 885, 834, and 844. Refer to appendix E, Supported Hardware Products, for more information about disk product characteristics.

    If possible, configure only one 9836 or 9853 volume to class C per controller to allow optimal use of the controller's buffer memory. The $FA7B5_A controller can process read requests for multiple disk units while waiting for NOS/VE to empty the buffer, if there is enough space remaining in the buffer for any concurrent read requests. If there is only one class C volume per controller, the controller's buffer is more likely to have space available for page fault requests on other volumes.

18. Separate frequently used applications and their files that may have been installed to the same disk volume. Place the applications and files on different volumes and possibly on different channel groupings to balance your input/output activity.

    Use either the REQUEST_MASS_STORAGE and COPY_FILE commands to accomplish this, or use the BACKUP_PERMANENT_FILES and RESTORE_PERMANENT_FILES utilities; the latter has a SET_RESTORE_OPTIONS subcommand that allows specification of the volume to which the file is assigned. Refer to chapter 8, Permanent File Maintenance, for more information on these utilities.

19. Keep all classes except A and Q off the system device. This also improves fault tolerance by removing the likelihood that an out-of-space condition could affect the system device. Further, if $SYSTEM catalogs are on a volume other than the system device, it is possible to recover all $SYSTEM files if the system device is initialized.

20. Configure alternate access to 887, 7155 (844 and 885) and 7165 (895) subsystems. If you have sufficient peripheral processors (PPs) and channels, alternate access can improve both performance and availability. Performance can improve because either access may be used to transfer data; therefore, fewer lost disk revolutions and balanced channel utilization result. Availability can improve because if one channel or controller fails, the other channel or controller may be used until the faulty one can be repaired. Refer to Alternate and Redundant Path Configurations in appendix E, Supported Hardware Products, for more information on alternate access.

21. You may want to have users place files that are used concurrently and frequently on different disk volumes to improve performance. For more information on device assignment, refer to appendix D, Assigning Files to a Specific Device.

Eight-Volume Target Configuration: Single Set

The following example configuration of eight volumes provides a balance between availability and fault tolerance. Your site may have to compromise availability for performance or for capacity depending upon how your site uses the system.

Volumes 1 to 4 are on one group of channels and volumes 5 to 8 are on a second channel grouping.

Class								Primary Use
	C	J	K	L	M	N	Q	
Disk1							x	System device
Disk2		x	x					$SYSTEM catalogs/files
Disk3	x			x				User catalogs
Disk4	x				x	x		User temporary and permanent files
Disk5	x				x	x		User temporary and permanent files
Disk6	x				x	x		User temporary and permanent files
Disk7	x				x	x		User temporary and permanent files
Disk8	x				x	x		User temporary and permanent files

NOTE

All classes must have at least one active member volume. For clarity in this example, the other class memberships have been omitted.

The following recommendations are from Configuration Recommendations earlier in this chapter. In the eight-volume target configuration:

- Configuration recommendation 6 cannot be met because there are seven classes and only eight volumes.
- Configuration recommendation 9 is not met because DISK2 belongs to both a catalog class and a permanent file class.
- Configuration recommendation 13 is not met because all volumes belong to one of the classes J, K, L, or M.
- Configuration recommendation 14 is not met because there are not enough volumes for a multiple set configuration.

Twelve-Volume Optimum Availability Configuration: Single Set

The following is a configuration example using twelve volumes. This configuration follows every recommendation for maximum system availability except the use of multiple sets. An attempt has been made to follow applicable performance recommendations. In addition, the numbers of swap and permanent file volumes have been divided equally among two of the channel groups. However, the emphasis is upon availability. The requirements of your site may dictate a compromise between availability and performance.

In the following configuration, volumes 1 to 4 are on one group of channels, volumes 5 to 8 are on a second channel grouping, and volumes 9 to 12 are on a third channel grouping.

Class								Primary Use
	C	J	K	L	M	N	Q	
Disk1							x	System device
Disk2		x						$SYSTEM catalogs
Disk3			x					$SYSTEM files
Disk4				x				User catalogs
Disk5						x		Temporary
Disk6					x			User permanent files
Disk7	x					x		Swap and temporary
Disk8	x				x			Swap and user permanent files
Disk9					x	x		User temporary and permanent files
Disk10					x	x		User temporary and permanent files
Disk11	x							Swap
Disk12	x				x	x		Swap, user temporary and permanent files

NOTE

All classes must have at least one active member volume. For clarity in this example, the other class memberships have been omitted.

Configuration recommendation 13 is not met because multiple sets are not configured.

Twelve-Volume Optimum Availability Configuration: Two Sets

The following is a configuration example using twelve volumes. This configuration follows every recommendation for maximum system availability including the use of multiple sets. An attempt has been made to follow applicable performance recommendations. In addition, the numbers of swap and permanent file volumes have been divided equally among two of the channel groups. However, the emphasis is upon availability. The requirements of your site may dictate a compromise between availability and performance.

In the following configuration, volumes 1 to 4 are on one group of channels, volumes 5 to 8 are on a second channel grouping, and volumes 9 to 12 are on a third channel grouping.

Class								**Primary Use**
System Set:	C	J	K	L	M	N	Q	
Disk1							x	System device
Disk2		x						$SYSTEM catalogs and critical
Disk3			x		x			$SYSTEM files
Disk4	x					x		Swap and temp
Disk5	x							Swap
Disk6						x		Temporary files
Disk7					x			Nonservice-critical $SYSTEM files
User Set:								
Disk8			x		x			Queued files and user permanent files
Disk9					x			User permanent files
Disk10					x			User permanent files
Disk11					x			User permanent files
Disk12		x		x				User and $SYSTEM catalogs

NOTE

All classes must have at least one active member volume. For clarity in this example, the other class memberships have been omitted.

Site Preparation

Maintaining the availability of your system requires planning and effort on the part of the system analyst. Without this involvement, the system will not be able to tolerate an uncorrectable disk unit, controller, or channel fault. Before such a fault occurs, do the following:

- Read the preceding configuration recommendations and physically and logically configure the system to follow most (preferably all) of the fault tolerance recommendations.
- Study the failure analysis and repair solutions provided in chapter 12, Failure Analysis, and chapter 13, Repair Solutions. Pare the analysis down to that which is pertinent for your peripheral configuration, and establish policies for who makes decisions in particular situations.
- Establish a policy of backing up files on volumes belonging to class K after upgrading the system or its applications or after installing a BCU. If you have a single set and members of class K are also members of class M, you will not gain as much from this recommendation as you would otherwise.

NOTE

The majority of NOS/VE's applications and files are not considered service critical and are placed on class M volumes for performance reasons.

If your site has developed applications that you consider service critical, you may also want to place them on the class K volumes. Refer to the REQUEST_MASS_STORAGE command in appendix D, Assigning Files to a Specific Device, for more information on file assignment.

Files on a class K volume are service-critical, NOS/VE permanent files. If one of these volumes fails, you may be able to resume service more rapidly if you can restore from a set of backup tapes that only contain these files. Use the SET_BACKUP_OPTIONS subcommand to exclude catalogs from this backup. Use the EXCLUDE_CATALOG subcommand to avoid backing up unnecessary files. Refer to chapter 8, Permanent File Maintenance, for more information on file backup. In this example, all job files are omitted from the backup.

Use the INCLUDE_VOLUMES subcommand of the BACKUP_PERMANENT_FILES utility to specify the names of the volumes that belong to class K. For example, assume the eight-volume target configuration discussed earlier and enter at the system console:

```
job job_class=system
request_magnetic_tape $local.backup type=mt9$6250 ..
  recorded_vsn=('clsk1' 'clsk2') ring=true
backup_permanent_files backup_file=$local.backup
set_backup_options exclude_catalog_information=true
exclude_catalog $system.$job_input_queue
exclude_catalog $system.$job_output_queue
exclude_catalog $system.$job_swap_queue
include_volumes recorded_vsns=(disk2)
backup_all_files
quit
jobend
```

- Establish a policy to periodically do a full backup of all files. Some sites do this every weekend. Refer to the CREATE_FULL_BACKUP command and the NOS/VE Operations manual for more information.

- Establish a policy to do a partial backup of files modified since your last full backup at the end of every day. Refer to the CREATE_PARTIAL_BACKUP command and the NOS/VE Operations manual for more information.

- Once the version 1.5.1 system is installed or upgraded, all permanent files (except swap files) belonging to the user $SYSTEM in any family are assigned to a class K volume *upon creation*. Therefore, you must have enough class K space for job input/output queues and any other permanent files belonging to $SYSTEM. Furthermore, whenever a permanent file belonging to $SYSTEM is restored, it is assigned to a class K volume by default, even though it may have originally been installed to a class M volume by the installation/upgrade process. Refer to the RESTORE_PERMANENT_FILES utility subcommand SET_RESTORE_OPTIONS in chapter 8, Permanent File Maintenance, for more information on file assignment. Refer to Restoring Files in chapter 13, Repair Solutions, for more information on restoring files.

- Establish a policy of periodically backing up catalogs. The purpose of a catalog backup is to record on magnetic tape the contents of all catalogs, including the location of each contained subcatalog or file on disk.

  A catalog backup may be used for either recovery or retrieval purposes.

  Recovery is performed when a catalog unit fails and the volume's catalogs are restored from a backup. In this situation, information in the catalog backup allows NOS/VE to recover the disk image of files and catalogs on available volumes. Refer to the RESTORE_UNRECONCILED_CATALOGS command and the RESTORE_PERMANENT_FILES utility subcommand RESTORE_MISSING_CATALOGS.

  Retrieval is performed when you restore a catalog that had been deleted from the system. Refer to the RESTORE_CATALOGED_FILES command in chapter 8, Permanent File Maintenance, and the RESTORE_PERMANENT_FILES utility subcommands RESTORE_ALL_FILES, RESTORE_CATALOG, and RESTORE_EXCLUDED_FILE_CYCLES.

  NOTE

  We recommend that you back up catalogs after any installation or upgrade of the system or after performing a file restoration. A catalog backup is only useful for recovery purposes if it accurately describes the residence of files and catalogs.

  It is very important that you back up catalogs after using the ARCHIVE/VE product to duplicate files and *before* you release any file space. If you fail to do so and lose a catalog volume after files are released, you will be unable to retrieve the files automatically.

  NOS/VE does not guarantee that a catalog backup may be used for the purpose of recovering files or catalogs at any release other than the one on which the catalog backup took place. However, a backup taken on version 1.3.1 may be used for retrieval on any future version of NOS/VE or on version 1.2.3.

  The commands CREATE_FULL_BACKUP and CREATE_PARTIAL_BACKUP include catalogs in the backup by default. However, you can (and probably should) backup catalogs more frequently than you would backup files.

We recommend that you backup catalogs separately from files. If you use a catalog-only backup, rather than using a partial or full backup to restore a missing catalog volume, it should take less time because you should have fewer tapes to mount and read.

The other advantage of a catalog-only backup is that it can be taken more often than a file backup. Unlike a file backup, it is not necessary to idle the system to perform a complete catalog backup. The BACKUP_PERMANENT_FILES utility waits for a busy catalog and all catalogs are backed up in one pass. Catalogs accessed by either a user or the BACKUP_PERMANENT_FILES utility remain busy for only a few milliseconds at a time, so there is little real-time impact when performing the catalog backup. The more frequently you backup catalogs, the fewer files and catalogs you risk losing if a catalog volume is lost.

The frequency with which you backup catalogs depends upon your users' rate of cycle creation and rate of change of permits, access logs, and file attributes. The frequency also depends on your willingness to take risks with your users' catalogs and files. If you value the accuracy of the catalog information, you may need to backup catalogs many times a day.

Use the CREATE_CATALOG_BACKUP command to create a catalog-only backup. Refer to the CREATE_CATALOG_BACKUP command and the NOS/VE Operations manual for more information.

- To reliably restore catalogs and files, it is necessary that you know which of the catalog-only partial backups and full backups was most recently performed. Save the listings of each backup you make so that you can determine the date and time of each backup and the magnetic tape media used for each backup. Refer to the Restoring Files and Restoring Catalogs sections in chapter 13, Repair Solutions, for more information on partial and full backups.

- If you have multiple sets, we recommend that you backup sets individually using the BACKUP_PERMANENT_FILES subcommand BACKUP_SET. If catalogs or files for a set require restoration, you do not have to process as many tapes as if all sets were in the same backup.

- If you want to use multiple sets and anticipate a need to prepare a file backup for another mainframe running a system at a version before 1.4.1, one of your sets must be named NVESET. Refer to the NOS/VE Software Release Bulletin for details.

- If your system prolog references any files, it is recommended that these files be placed on a class K (service critical) volume. This allows login to complete even if a non-service critical volume becomes missing or unavailable. You may place files on class K volumes using the REQUEST_MASS_STORAGE command, which is described in appendix D of this manual.

- The following table lists the released default values for a number of the system attributes. We recommend that you define these attributes in your DCFILE file. You should use the released values for these attributes at all times unless you are directed to change them by the instructions in chapter 12, Failure Analysis. Ensure that these attributes are set to their released values at all times unless you are directed to change them. Refer to chapter 5, Deadstart File Software, for more information about DCFILE creation.

Attribute	Value
DELETE_UNRECONCILED_FILES	0
IGNORE_IMAGE	0
RECOVER_AT_ALL_COSTS	0
VALIDATE_ACTIVE_SETS	0

- To determine the classes in use at the last deadstart, add the following commands to a system initialization prolog or epilog. Refer to repair solution 6 in chapter 13, Repair Solutions. Refer to chapter 6, Site Tailoring, for further information on system initialization.

  ```
  lcu
  request_mass_storage $system.mainframe.classes_at_deadstart ..
    initial_volume=system_device
  display_ms_class output=$system.mainframe.classes_at_deadstart
  quit
  ```

  The output of the display is assigned to the system device to ensure its availability should a noncritical disk unit fail. The name of the file is not important; that is, you need not choose CLASSES_AT_DEADSTART as a file name.

  The display of the mass storage class configuration that was in effect when you last deadstarted is especially useful. Whenever you change the class membership of a volume, you must understand that you are only affecting the assignment of future files to the volume.

  Any volume that is a member of class Q or J when you last deadstarted remains critical, even though you may have subsequently removed the volume from these classes. Take this display into consideration when you change mass storage class assignments and the state of disk elements. Refer to repair solution 6 in chapter 13, Repair Solutions, for more information on critical volumes.

- You may choose to add the following command to a system initialization prolog:

  ```
  display_log output=$system.deadstart_job_log.$next
  ```

  NOS/VE records the reconfiguration menus and the responses to those menus in the job log of the system job. You may choose to save the log after each deadstart for later analysis.

- Reserve and label a number of magnetic tapes for taking dumps of NOS/VE memory for failure analysis and recovery. EDD supports labelled tapes. We recommend that you use labelled tapes to ensure that you mail the correct dump tape with your PSR. Refer to the LCU subcommand INITIALIZE_TAPE_VOLUME in the NOS/VE Operations manual for more information on labelling tapes.

- Whenever you recable the system, print the physical configuration after you resume service. From the System Operator Utility, enter:

  ```
  display_system_configuration output=$local.print
  print_file $local.print
  ```

  Keep this listing or an accurate picture of the latest peripheral configuration near the system console. In particular, note the location of the system device and catalog devices in the configuration and the existence of alternate and redundant paths to these devices.

- Always rebuild your deadstart file on disk or tape to reflect any reconfiguration you may have performed during the last deadstart. The file $SYSTEM.MAINFRAME.CONFIGURATION is created by NOS/VE at the completion of every deadstart. This file contains the PCU subcommands necessary to define the physical configuration in effect at the last deadstart. Enter the following commands from the System Operator Utility prior to rebuilding the deadstart tape:

  ```
  set_working_catalog $system
  copy_file input=mainframe.configuration ..
    output=site_os_maintenance.deadstart_commands.physical_configuration
  set_working_catalog $local
  ```

  If you maintain a configuration prolog on your deadstart tape, change the LCU_MAINFRAME_SUBCOMMANDS file in this prolog to reflect changes in your mass storage class assignments or physical configuration.

Failure Analysis 12

Failure Analysis 12

This chapter discusses alternatives available for reacting to mainframe faults, NOS/VE operating system faults, and disk and magnetic tape peripheral faults. Failures of communications equipment and other peripherals are not addressed by this manual. Adverse environmental and power loss conditions are described in the NOS/VE Operations manual.

Sites without onsite hardware maintenance coverage will benefit most from these chapters. If your site has a resident CE, you may involve the CE in the failure analysis at an earlier point than indicated in this chapter. At various points in the analysis, "Can you wait for repair?" is asked. This question implies such factors as:

- Are CEs available on site?
- Is the problem understood?
- Are replacement parts available?
- Would it be more efficient to postpone the repair and run in a degraded mode? If the CE is on site, recabling may be an alternative to immediate repair.

Two alternatives are given to each question. One alternative is to wait for repair. The other is to postpone the repair, performing some temporary action to return to production. You evaluate the alternatives and instruct the CE or the operator accordingly.

The failure analysis process is in the form of a decision tree and subtrees that investigate a failure and recommend a course of action. The most help is obtained when the current version of NOS/VE has already been installed and in use for some time prior to a failure. These chapters do not address a greater variety of failures that could occur during an installation or a continuation deadstart.

The most effective solution to a peripheral failure depends upon the physical and logical peripheral configuration of the mainframe. You follow this decision tree from the top down (starting at Begin Failure Analysis, see figure 12-1) to find the most effective solution for a particular configuration and circumstance. If you do not start at the top of the decision tree, you may not arrive at the best solution or even an appropriate one.

Each section of this chapter consists of two parts. One part contains a decision subtree and the other part provides information and clarification of the significant aspects of the subtree. Identical labels occur in the subtree and the explanatory text in this chapter and in chapter 13, Repair Solutions.

The decision tree describes actions an operator can take without the aid of a CE. Actions described may include hardware repair actions (for example, making a disk unit ready) and complicated logical reconfiguration processes (for example, editing the physical configuration at a deadstart).

Board replacement, physical recabling, and powering on and off equipment are assumed to be the action of a CE and are not recommended here. For hardware problems, any action requiring an operator to use a key to open a peripheral cabinet is not recommended here. Although some information provided may help a CE repair a problem, in general, this chapter does not describe the CE's troubleshooting process.

Begin Failure Analysis

Figure 12-1 shows the top of the decision tree. The analysis of each new problem must begin here.

Either the site analyst is informed of a problem by users or perceives a problem and initiates the analysis.

IS THE SYSTEM MESSAGE LINE BLANK?

The very top line of the system console is the system message line. If this line is blank, as is normally the case, then the system should be providing service (although perhaps in a degraded mode). If this line is not blank, then a software or hardware problem has occurred and the system is in a stepped state or is attempting to enter a stepped state.

ARE YOU HAVING A PROBLEM WITH MAGNETIC TAPE?

Have your users complained about tape tasks failing or have you noticed a problem with one or more magnetic tape units?

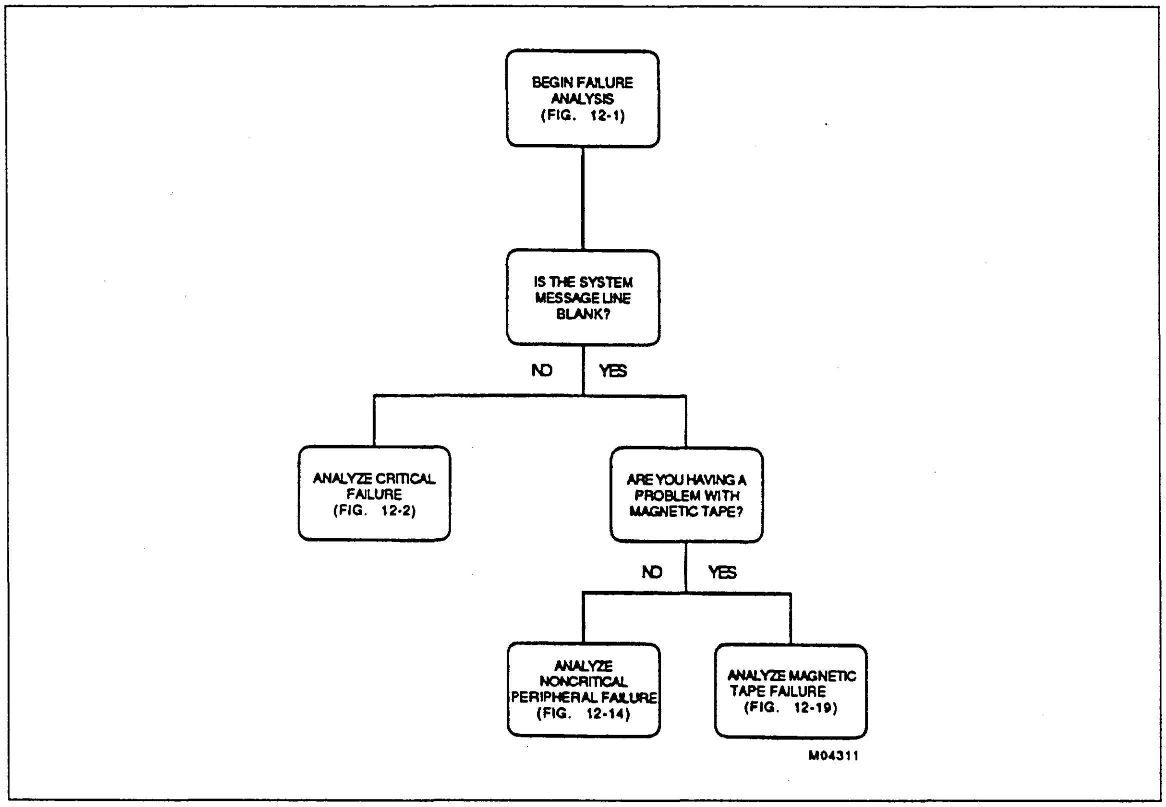

BEGIN FAILURE ANALYSIS (FIG. 12-1)
IS THE SYSTEM MESSAGE LINE BLANK?
NO
YES
ANALYZE CRITICAL FAILURE (FIG. 12-2)
ARE YOU HAVING A PROBLEM WITH MAGNETIC TAPE?
NO
YES
ANALYZE NONCRITICAL PERIPHERAL FAILURE (FIG. 12-14)
ANALYZE MAGNETIC TAPE FAILURE (FIG. 12-19)
M04311

Figure 12-1. Begin Failure Analysis

Analyze Critical Failure

NOS/VE may have intentionally suspended its own execution (put itself into a stepped state) due to the failure of a critical element.

ERR=VEOS9300 IN SYSTEM MESSAGE LINE?

The VEOS9300 fault symptom code indicates that a critical disk volume is no longer accessible due to a failure. The failure data has been recorded on the critical display window of the system console prior to the system going into a stepped state. If this fault symptom code is not present, the mainframe hardware or operating system software is at fault.

ERR=DCmmxxxx or DDmmxxxx IN SYSTEM MESSAGE LINE?

The DCmmxxxx and DDmmxxxx fault symptom codes indicate that a central processor may be in a hung or fault condition. After approximatly 20 seconds, NOS/VE may attempt to disable the CPU. If this is the only processor, NOS/VE may enter a stepped condition; otherwise, the faulty CPU is downed if the condition persists and the processor is not required for dual-state operation.

The second character in the fault symptom code identifies the CPU. The characters DC and DD imply processors 0 (zero) and 1, respectively. The next two characters in the fault symptom code identify the CPU model number. The remaining characters identify the fault condition.

ERR=DCmmZ617 or DDmmZ617 IN SYSTEM MESSAGE LINE?

The DCmmZ617 fault symptom code indicates that DFT timed out NOS/VE after 30 seconds had elapsed without a processor having communicated its status to DFT.

The second character in the fault symptom code identifies the CPU. The characters DC and DD imply processors 0 (zero) and 1, respectively. The next two characters in the fault symptom code identify the CPU model number. The remaining characters identify the fault condition.

This condition may be caused by the need within NOS/VE to serialize a process during which the system console is not refreshed. If the condition lasts more than 5 minutes from the time it is first reported, it is a fault. If the system has resumed service after this fault symptom code is displayed, the displays on the system console will be updating, and you are able to enter console commands again.

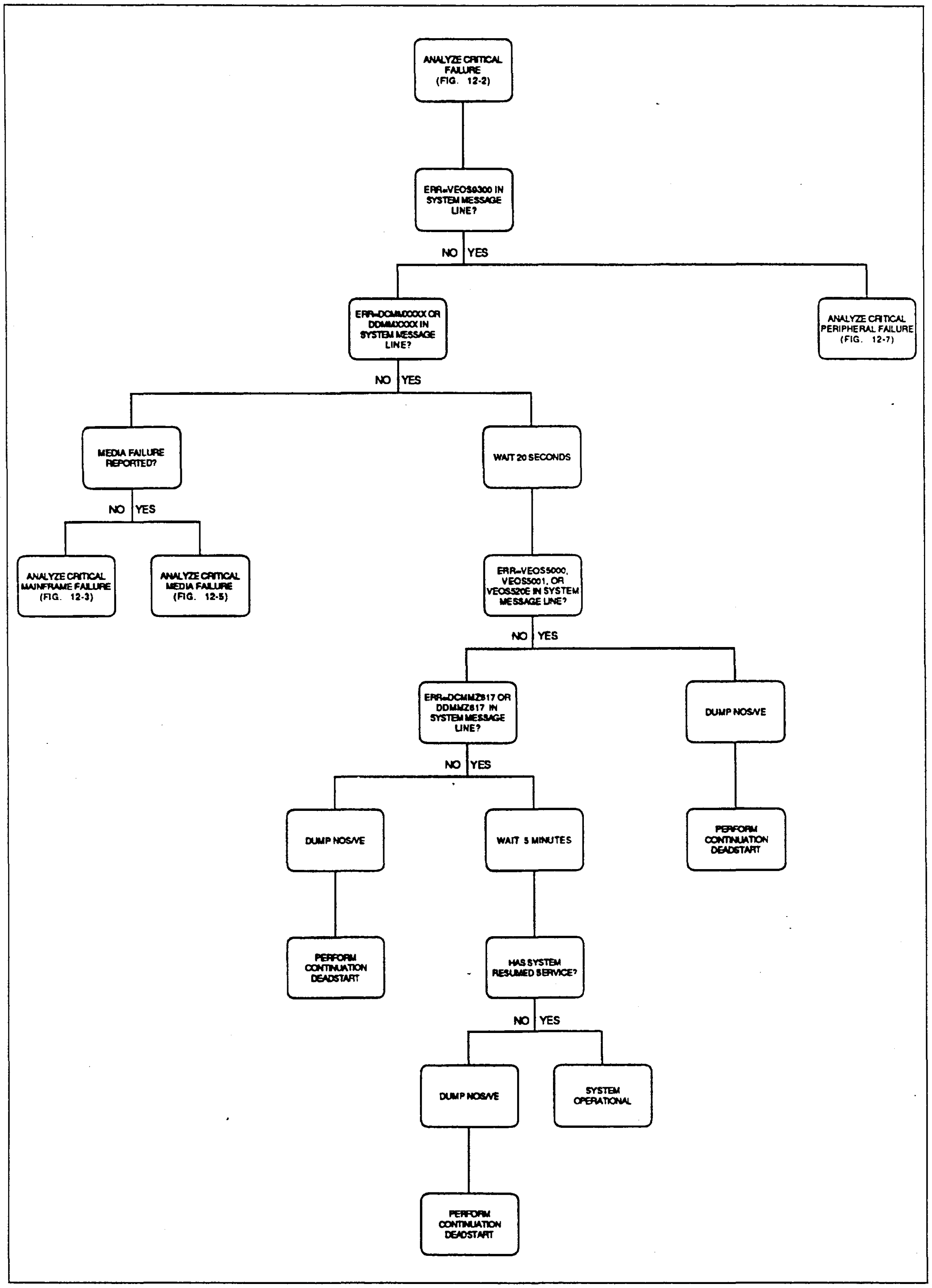

ANALYZE CRITICAL FAILURE (FIG. 12-2)
ERR=VEOS0300 IN SYSTEM MESSAGE LINE?
NO
YES
ERR=DCMMXXXX OR DDMMXXXX IN SYSTEM MESSAGE LINE?
ANALYZE CRITICAL PERIPHERAL FAILURE (FIG. 12-7)
NO
YES
MEDIA FAILURE REPORTED?
WAIT 20 SECONDS
NO
YES
ANALYZE CRITICAL MAINFRAME FAILURE (FIG. 12-3)
ANALYZE CRITICAL MEDIA FAILURE (FIG. 12-5)
ERR=VEOS5000, VEOS5001, OR VEOS520E IN SYSTEM MESSAGE LINE?
NO
YES
ERR=DCMMZ817 OR DDMMZ817 IN SYSTEM MESSAGE LINE?
DUMP NOS/VE
NO
YES
DUMP NOS/VE
WAIT 5 MINUTES
PERFORM CONTINUATION DEADSTART
PERFORM CONTINUATION DEADSTART
HAS SYSTEM RESUMED SERVICE?
NO
YES
DUMP NOS/VE
SYSTEM OPERATIONAL
PERFORM CONTINUATION DEADSTART

Figure 12-2. Analyze Critical Failure

MEDIA FAILURE REPORTED?

Move the cursor to the critical display window of the system console and expand the window (press the shift and F9 (or SUPER) keys at the same time). Refer to Failure Data in Critical Display Window later in this chapter for information about mass storage peripheral failure data in the critical display window.

Locate the UNRECOVERED message group in the critical display window. If you cannot find such a message, then a media failure has not been reported. If the second line of the message contains the words MEDIA FAILURE, then an uncorrectable media defect has been encountered on a disk volume.

HAS SYSTEM RESUMED SERVICE?

If you see the time updating at the system console and you can enter VEDISPLAY commands, the system has probably returned to normal service.

This page intentionally left blank.

Analyze Critical Mainframe Failure

NOS/VE is in a stepped state due to a mainframe or an operating system failure.

POWER OR ENVIRONMENTAL WARNING ENCOUNTERED?

An environmental failure is shown by one of the following messages:

```
VEOS0005 - System IDLED due to LONG POWER WARNING
VEOS0006 - System IDLED due to ENVIRONMENT WARNING
VEOS0009 - System IDLED due to SHORT WARNING
```

ERR=Dxxxxxx IN SYSTEM MESSAGE LINE?

DFT reports mainframe faults to the right half of the system message line. All DFT-detected faults have the prefix ERR=D. A description of mainframe error messages that appear on the system message line is in the file:

$SYSTEM.MANUALS.LINE_PRINTER_MANUALS.MAINFRAME_ERROR_
MESSAGES

WAIT UP TO 4 MINUTES FOR SYSTEM TO RESPOND

Wait 4 minutes from the time shown in the critical display window indicating when the system entered a stepped state. The 4-minute wait exhausts the backup battery's power supply. If power returns to normal or the environmental condition clears within this time, the system writes information to the critical display window showing the change.

DID SYSTEM RESPOND?

The system prompts you when an environmental condition ends within the allotted time.

2

WAS AUTOMATIC_UNSTEP_RESUME SELECTED?

Refer to the NOS/VE System Performance and Maintenance manual, Volume 1, for documentation of the system attribute AUTOMATIC_UNSTEP_RESUME.

IS THE CLOCK UPDATING ON SYSTEM CONSOLE?

If the system console is not being refreshed after 4 minutes, the mainframe has been powered off due to a power failure or an environmental condition.

ENTER COMMAND WHEN PROMPTED

If you see the following line, enter the RESUME_SYSTEM command in the input line of the critical display window:

```
System ready to RESUME manually; use console command.
```

If you see the following line, enter the UNSTEP_SYSTEM command in the input line of the critical display window:

```
System ready to UNSTEP manually; use console command.
```

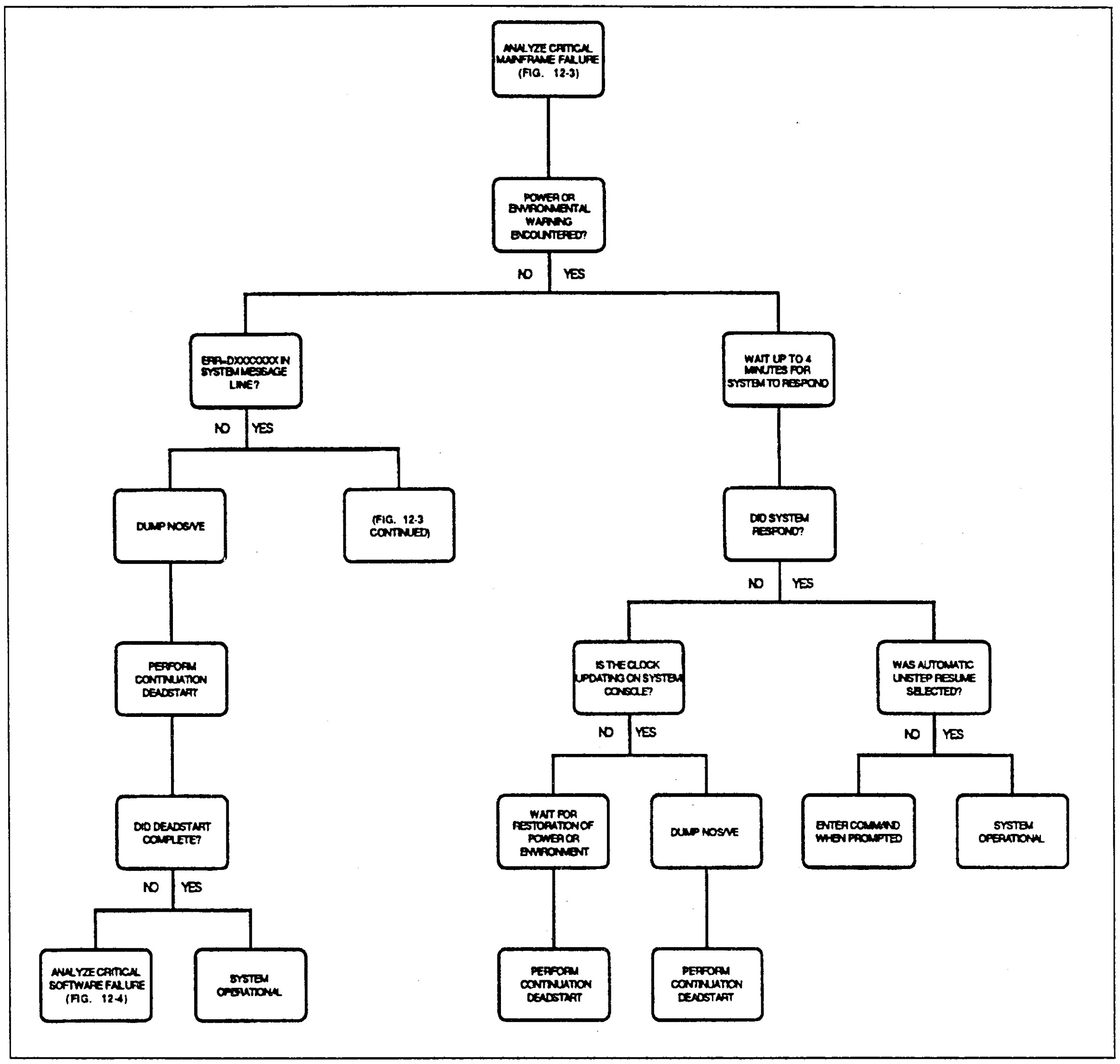

ANALYZE CRITICAL MAINFRAME FAILURE (FIG. 12-3)
POWER OR ENVIRONMENTAL WARNING ENCOUNTERED?
NO
YES
ERR=DXXXXXXX IN SYSTEM MESSAGE LINE?
NO
YES
DUMP NOS/VE
(FIG. 12-3 CONTINUED)
PERFORM CONTINUATION DEADSTART
DID DEADSTART COMPLETE?
NO
YES
ANALYZE CRITICAL SOFTWARE FAILURE (FIG. 12-4)
SYSTEM OPERATIONAL
WAIT UP TO 4 MINUTES FOR SYSTEM TO RESPOND
DID SYSTEM RESPOND?
NO
YES
IS THE CLOCK UPDATING ON SYSTEM CONSOLE?
NO
YES
WAIT FOR RESTORATION OF POWER OR ENVIRONMENT
PERFORM CONTINUATION DEADSTART
DUMP NOS/VE
PERFORM CONTINUATION DEADSTART
WAS AUTOMATIC UNISTEP RESUME SELECTED?
NO
YES
ENTER COMMAND WHEN PROMPTED
SYSTEM OPERATIONAL

Figure 12-3. Analyze Critical Mainframe Failure

(Continued)

(Continued)

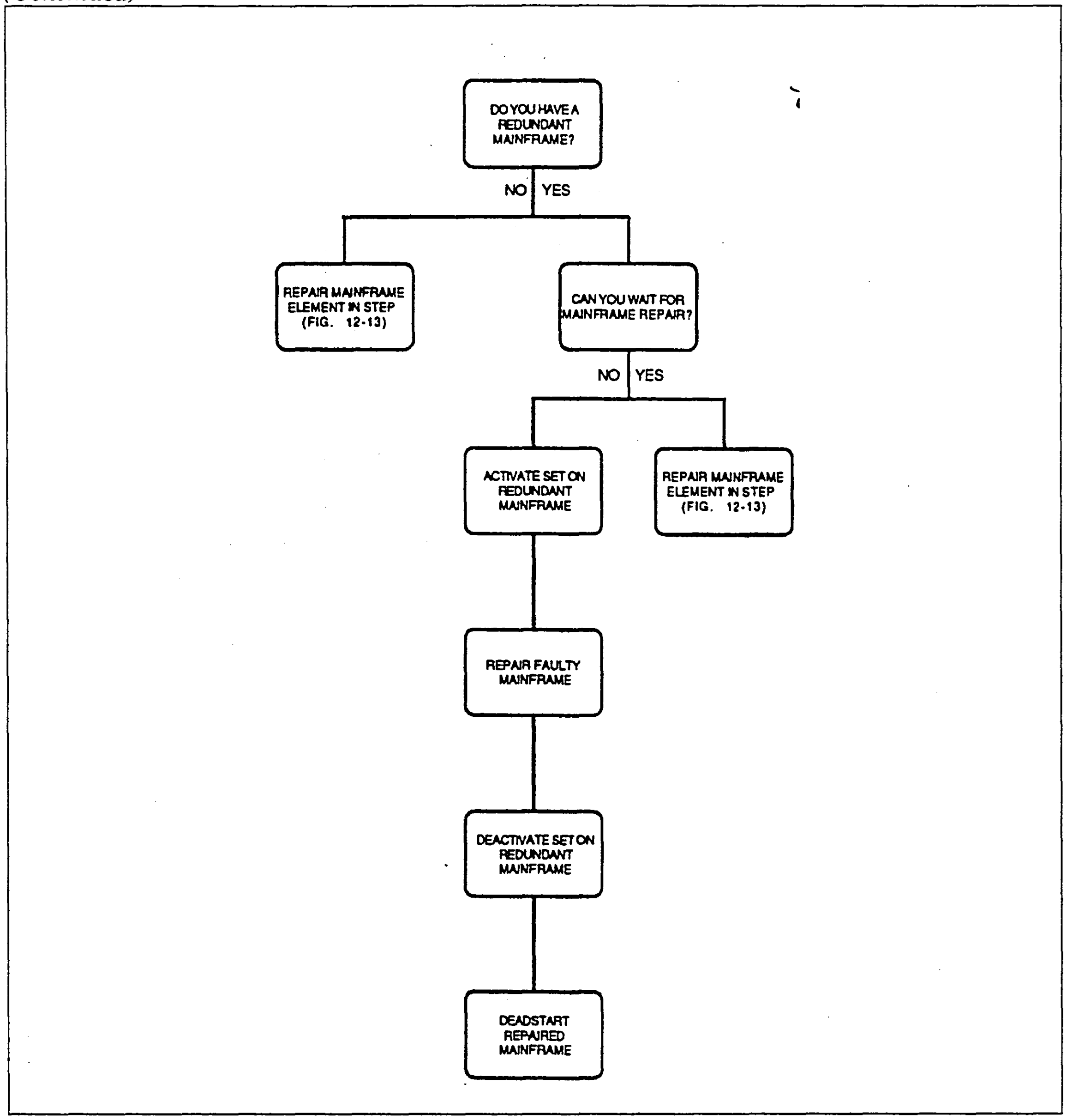

DO YOU HAVE A REDUNDANT MAINFRAME?
NO
YES
REPAIR MAINFRAME ELEMENT IN STEP (FIG. 12-13)
CAN YOU WAIT FOR MAINFRAME REPAIR?
NO
YES
ACTIVATE SET ON REDUNDANT MAINFRAME
REPAIR MAINFRAME ELEMENT IN STEP (FIG. 12-13)
REPAIR FAULTY MAINFRAME
DEACTIVATE SET ON REDUNDANT MAINFRAME
DEADSTART REPAIRED MAINFRAME

Figure 12-3. Analyze Critical Mainframe Failure

DO YOU HAVE A REDUNDANT MAINFRAME?

Have you configured access to one or more mass storage sets from two mainframes? If so, it is possible to switch one or more sets from the failing mainframe to the redundant mainframe.

CAN YOU WAIT FOR MAINFRAME REPAIR?

You can either have a CE repair the faulty mainframe immediately or you can logically switch the permanent files from the faulty mainframe to the redundant one, depending on the urgency of continued service.

ACTIVATE SET ON REDUNDANT MAINFRAME

Using the LCU subcommand CHANGE_ELEMENT_STATE, ensure on the redundant mainframe that all of the disk units, disk controllers, and disk channels that provide access to the sets to be activated are in the ON state. At the system console, enter the following command for each set:

```
activate_set set_name= <set name> validate_set=true
```

REPAIR FAULTY MAINFRAME

While using the redundant mainframe to continue access to the set, call a CE to diagnose and repair the faulty mainframe.

DEACTIVATE SET ON REDUNDANT MAINFRAME

When convenient, use the TERMINATE_SYSTEM command to terminate use of the set. TERMINATE_SYSTEM must be entered from within the System Operator Utility. Selecting pause for operator input, deadstart the redundant mainframe. Select intervention prior to the activation of all volumes during the continuation deadstart. Use the LCU subcommand CHANGE_ELEMENT_STATE to change the state of all elements to OFF that you plan to activate on the repaired mainframe.

DEADSTART REPAIRED MAINFRAME

Selecting pause for operator input, deadstart the repaired mainframe. Select intervention prior to the activation of all volumes during the continuation deadstart. Use the LCU subcommand CHANGE_ELEMENT_STATE to change the state of all elements to ON that you plan to reinstate on the repaired mainframe.

Analyze Critical Software Failure

The operator has already tried unsuccessfully to perform a continuation deadstart to recover from a previous fault that was either a software fault or an undetected hardware fault. This analysis deals only with the inability to perform the continuation deadstart. This problem may or may not be related to the original fault.

FILE SYSTEM FAILURE ENCOUNTERED?

If one of the following messages is displayed, a file system software failure has occurred on one volume. The RECORDED_VSN field identifies the name (six characters or less) given the volume by the operator when the volume was initialized.

```
ERR=VEOS1100- <recorded_vsn> log starts incorrectly
ERR=VEOS1100- <recorded_vsn> log check byte missing
ERR=VEOS1100- <recorded_vsn> invalid log entry
```

TRY RECOVERY AT ALL COSTS

The disk volume identified in the VEOS1100 message has incurred damage to one of its system files that could not be tolerated. This and the subsequent step attempt to avoid the problem without losing all of the files on the volume. Retry the continuation deadstart requesting pause for operator input. At the following prompt:

```
Enter system core commands
```

enter the following system core commands:

```
set_system_attribute recover_at_all_costs 1
auto
```

TRY RECOVERY WITHOUT IMAGE

This recommendation is given so that you can recover all files that were not modified since your last file backup. However, the files that were modified since your last file backup may be inconsistent as a result of recovering the system without an image. Refer to Finding Inconsistent Files in chapter 13, Repair Solutions, for more information on file recovery. Retry the continuation deadstart requesting pause for operator input.

At the following prompt:

```
Enter system core commands
```

enter the following system core commands:

```
set_system_attribute ignore_image 1
auto
```

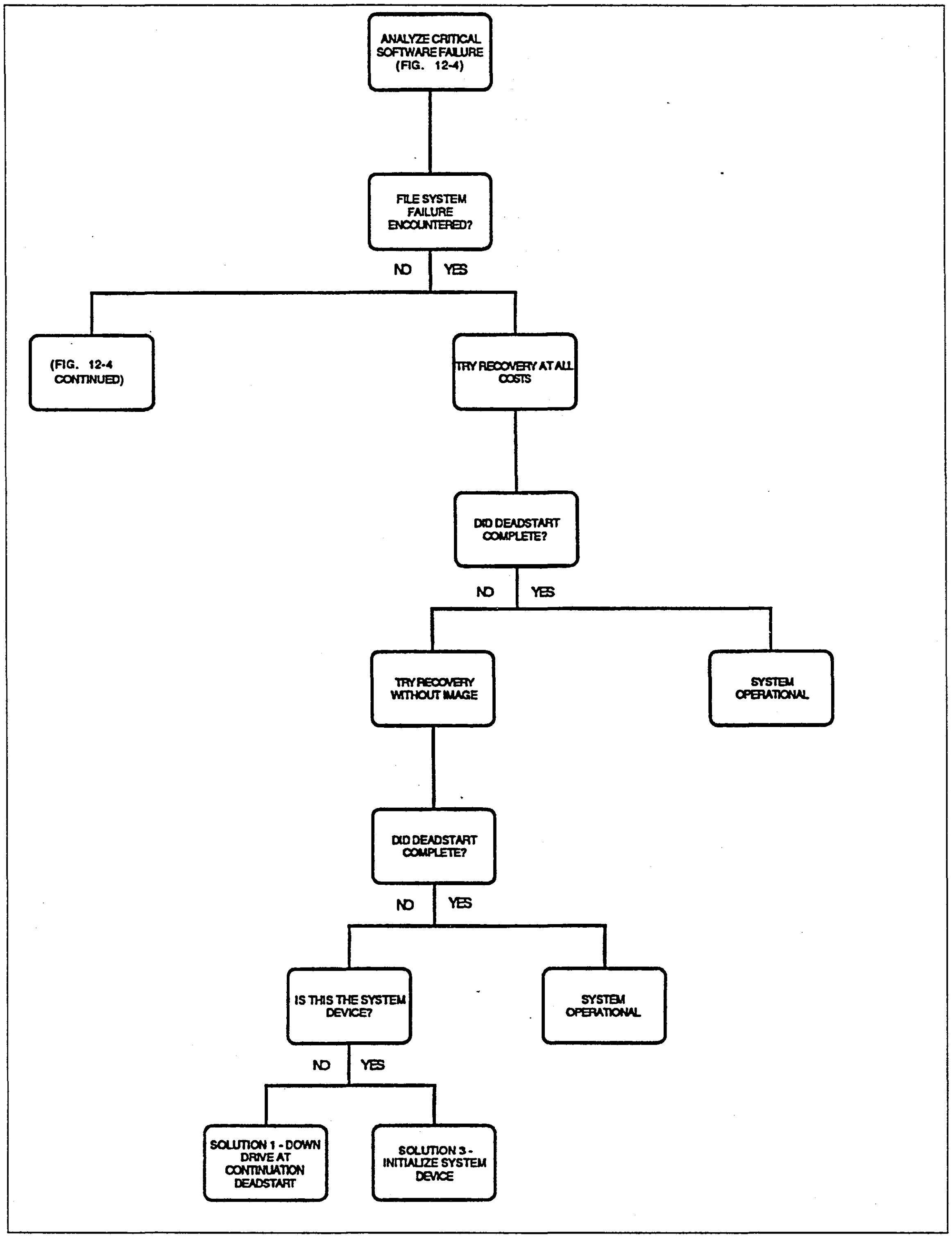

ANALYZE CRITICAL SOFTWARE FAILURE (FIG. 12-4)
FILE SYSTEM FAILURE ENCOUNTERED?
NO
YES
(FIG. 12-4 CONTINUED)
TRY RECOVERY AT ALL COSTS
DID DEADSTART COMPLETE?
NO
YES
TRY RECOVERY WITHOUT IMAGE
SYSTEM OPERATIONAL
DID DEADSTART COMPLETE?
NO
YES
IS THIS THE SYSTEM DEVICE?
SYSTEM OPERATIONAL
NO
YES
SOLUTION 1 - DOWN DRIVE AT CONTINUATION DEADSTART
SOLUTION 3 - INITIALIZE SYSTEM DEVICE

Figure 12-4. Analyze Critical Software Failure

(Continued)

(Continued)

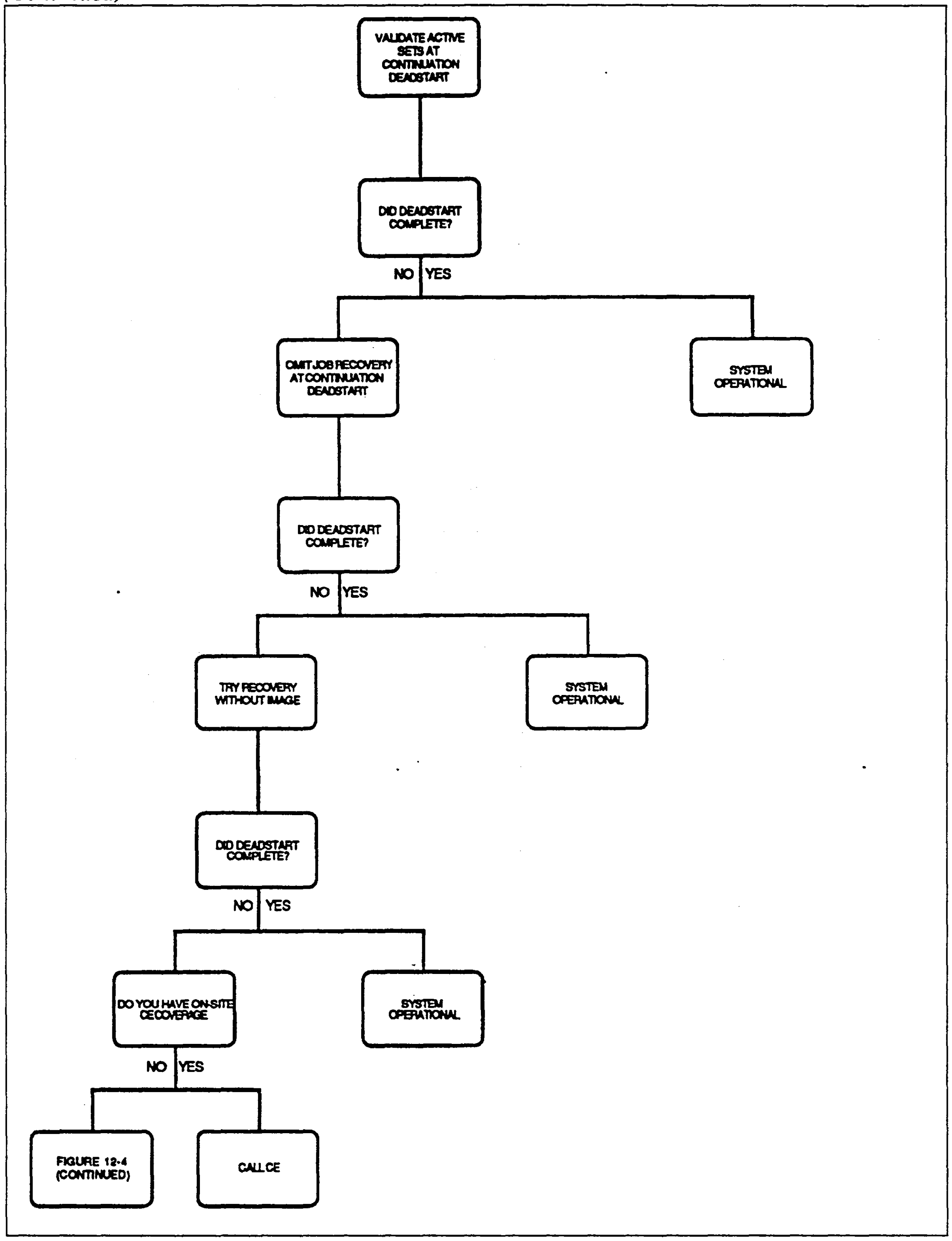


Figure 12-4. Analyze Critical Software Failure

(Continued)

(Continued)

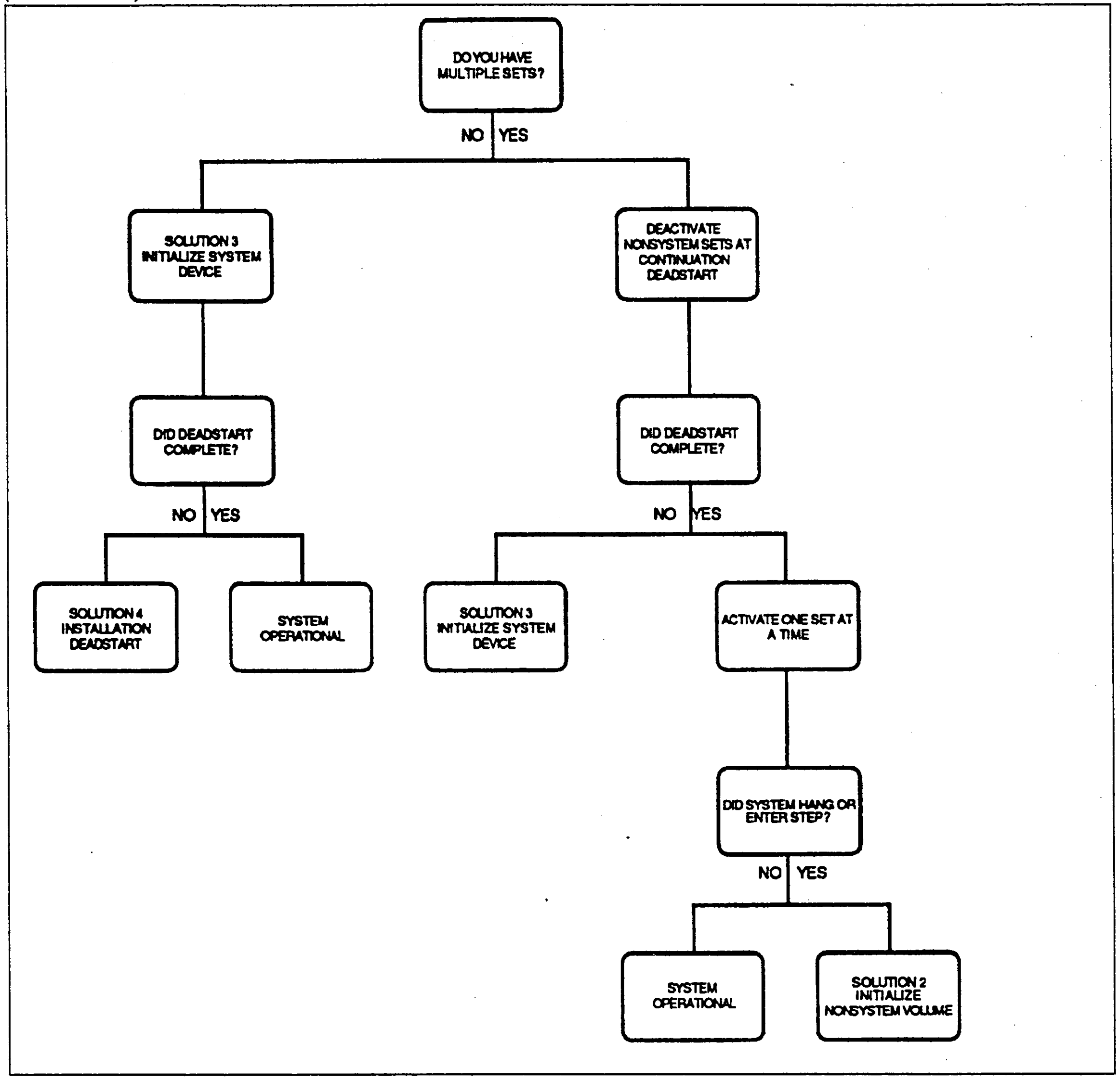

DO YOU HAVE MULTIPLE SETS?
NO
YES
SOLUTION 3 INITIALIZE SYSTEM DEVICE
DEACTIVATE NONSYSTEM SETS AT CONTINUATION DEADSTART
DID DEADSTART COMPLETE?
NO
YES
DID DEADSTART COMPLETE?
NO
YES
SOLUTION 4 INSTALLATION DEADSTART
SYSTEM OPERATIONAL
SOLUTION 3 INITIALIZE SYSTEM DEVICE
ACTIVATE ONE SET AT A TIME
DID SYSTEM HANG OR ENTER STEP?
NO
YES
SYSTEM OPERATIONAL
SOLUTION 2 INITIALIZE NONSYSTEM VOLUME

Figure 12-4. Analyze Critical Software Failure

VALIDATE ACTIVE SETS AT CONTINUATION DEADSTART

The deadstart may not have completed due to a catalog fault. To determine whether this is the problem, retry the continuation deadstart requesting pause for operator input.

At the following prompt:

```
Enter system core commands
```

enter the following system core commands:

```
set_system_attribute validate_active_sets 1
auto
```

OMIT JOB RECOVERY AT CONTINUATION DEADSTART

The deadstart may not have completed due to some fault in the recovery of active jobs. To deselect active job recovery, retry the continuation deadstart requesting pause for operator input.

At the following prompt:

```
Enter system core commands
```

enter the two system core commands:

```
set_system_attribute job_recovery_option 3
auto
```

TRY RECOVERY WITHOUT IMAGE

This recommendation is given so that you can recover all files that were not modified since your last file backup. However, the files that were modified since your last file backup may be inconsistent as a result of recovering the system without an image. Refer to Finding Inconsistent Files in chapter 13, Repair Solutions, for more information on file recovery. Retry the continuation deadstart requesting pause for operator input.

At the following prompt:

```
Enter system core commands
```

enter the three system core commands:

```
set_system_attribute ignore_image 1
set_system_attribute recover_at_all_costs 1
auto
```

DO YOU HAVE ONSITE CE COVERAGE?

So far you have only lost active jobs and files that were in memory at the moment of the original fault. However, if you proceed, you may lose files modified since your last backup. If you have onsite CE coverage, the CE may be able to diagnose a problem with the mainframe that may not have been detected by DFT.

If you found no problem, or you did find a problem but have no onsite CE coverage and prefer to proceed rather than wait for a CE to arrive, you may choose to initialize the system device.

DO YOU HAVE MULTIPLE SETS?

Because you are unable to deadstart, it is useful to temporarily eliminate nonsystem sets to determine whether a problem with one set could possibly be the reason for the inability to deadstart.

DEACTIVATE NONSYSTEM SETS AT CONTINUATION DEADSTART

Perform a continuation deadstart requesting pause for operator input. At the NOS/VE Reconfiguration menu, select the menu option to intervene prior to the activation of existing mass storage set members. For each disk unit whose volume is a member of a set other than the system set, enter the following LCU subcommand:

```
change_element_state element=<name> state=off
```

After all nonsystem set members have been turned off, enter QUIT and attempt to complete the deadstart. If a disk channel or channel pair is dedicated to a set, it is sufficient to change the state of the channel to OFF instead of changing the state of all the units individually.

ACTIVATE ONE SET AT A TIME

Now that you have successfully deadstarted, try to determine which nonsystem set was preventing the deadstart. From the System Operator Utility (SOU), enter the Logical Configuration Utility and repeat the following for each set member of each nonsystem set:

```
SOU/lcu
LCU/change_element_state element=<name> state=on
LCU/quit
```

Enter the following command once for each nonsystem set:

```
SOU/activate_set set_name=<name> validate_set=true ..
SOU../delete_unreconciled_files=false
```

DID THE SYSTEM HANG OR STEP?

If the system could not tolerate the activation of the set, the members of the set must be initialized and restored.

Analyze Critical Media Failure

The NOS/VE operating system is in a stepped state due to a disk media failure that was not corrected or tolerated by the system.

RECORD FAILURE LOCATION

Refer to Failure Data in Critical Display Window later in this chapter for more information on the presentation of disk failure data in the critical display window. Make a note of the RECORDED_VSN of the volume and the address of the media defect on the volume. The address is expressed in terms of the cylinder, track, and sector and the numeric values for these are expressed in a decimal representation in the critical display window. The values provided for the address of the media defect may be given directly to the DEFINE_MS_FLAW subcommand without having to be converted; refer to the Analyze Repetitive Media Failure and Failure Data in Critical Display Window sections in this chapter for more information on disk failure data.

FLAW MEDIA AT CONTINUATION DEADSTART

Perform a continuation deadstart requesting pause for operator intervention. Refer to Terminology in chapter 11, Fault Tolerance, for more information about selecting pause for operator intervention.

At the following prompt:

```
Enter system core commands
```

enter the following system core commands:

```
define_ms_flaw recorded_vsn=xxxxxx c=cccc t=ttt s=ss
auto
```

Complete the continuation deadstart, if possible.

DID DEADSTART COMPLETE?

If you are able to enter operator commands, then the deadstart was successful.

CRITICAL WINDOW DISPLAYING SAME MEDIA FAILURE?

Has the same cylinder, track, and sector been reported for the same volume as before?

DIFFERENT LOCATION BUT SAME VOLUME?

Has a different cylinder, track, or sector been reported as a MEDIA FAILURE on the same volume as before? If so, the disk unit may be the problem; otherwise, the problem may be in a controller that is cabled to the disk units.

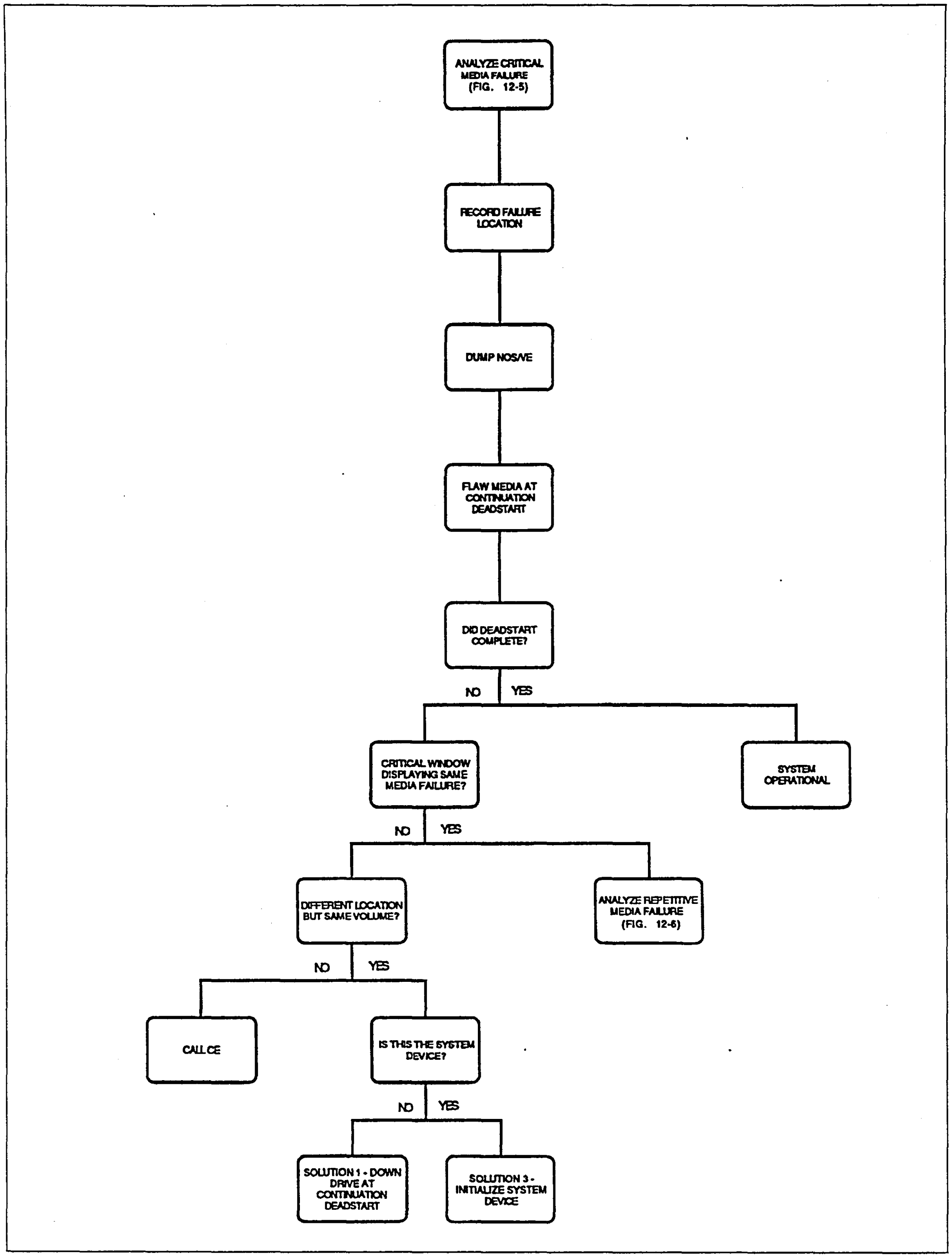

ANALYZE CRITICAL MEDIA FAILURE (FIG. 12-5)
RECORD FAILURE LOCATION
DUMP NOS/VE
FLAW MEDIA AT CONTINUATION DEADSTART
DID DEADSTART COMPLETE?
NO
YES
CRITICAL WINDOW DISPLAYING SAME MEDIA FAILURE?
SYSTEM OPERATIONAL
NO
YES
DIFFERENT LOCATION BUT SAME VOLUME?
ANALYZE REPETITIVE MEDIA FAILURE (FIG. 12-6)
NO
YES
CALL CE
IS THIS THE SYSTEM DEVICE?
NO
YES
SOLUTION 1 - DOWN DRIVE AT CONTINUATION DEADSTART
SOLUTION 3 - INITIALIZE SYSTEM DEVICE

Figure 12-5. Analyze Critical Media Failure

CALL CE

You are experiencing media failures on multiple volumes at the same time. If the disk units are connected to the same controller, it may be that the controller is at fault. You will probably need a CE to diagnose this failure.

IS THIS THE SYSTEM DEVICE?

Determine from the physical address (channel, controller, and unit) whether this is the disk unit containing the NOS/VE system.

This page intentionally left blank.

Analyze Repetitive Media Failure

The operator has previously attempted to perform a continuation deadstart following the report of an uncorrectable disk media failure; the same failure prevented the completion of the continuation deadstart.

IS THIS THE SYSTEM DEVICE?

Determine from the physical address (channel, controller, and unit) whether this is the disk unit containing the NOS/VE system.

HAVE YOU ALREADY TRIED INITIALIZING THIS VOLUME? - DO YOU NEED THE CAPACITY?

If you have sufficient extra capacity to down the disk unit until it is repaired, then do so. You may need to restore files or catalogs, depending upon the class membership of the downed volume. If you need the capacity of this volume, then initialize the volume at a continuation deadstart.

TRY RECOVERY WITHOUT IMAGE

Because the operator has already tried one deadstart that failed during file recovery and the fault was not with the data structures of a specific volume, the fault may be with the central memory image or with the software that performs the recovery of files from central memory. We recommend that you retry the continuation deadstart without file recovery from the memory image. This recommendation is given so that you can recover all files that were not modified since your last file backup; however, the files that were modified since your last file backup may be inconsistent as a result of this recovery. Refer to Finding Inconsistent Files at the end of chapter 13, Repair Solutions, for more information on file recovery. Retry the continuation deadstart requesting pause for operator input.

At the following prompt:

```
Enter system core commands
```

enter the following system core commands:

```
set_system_attribute ignore_image 1
auto
```

FLAW MEDIA DEFECT

Manual flawing is recommended as a precaution because earlier attempts to flaw the defect did not resolve the problem. If the defect has been previously flawed, you are given a warning message when you try to flaw the same defect. You must know the volume's RECORDED_VSN and the address of the defect on the volume. The address is given in terms of the cylinder, track, and sector; the numeric values for these are given in a decimal representation and may be given directly to the DEFINE_MS_FLAW subcommand. After the system is activated, enter the following commands from within the System Operator Utility (SOU):

```
SOU/lcu
LCU/define_ms_flaw recorded_vsn= xxxxxx c=cccc t=ttt s=ss
LCU/quit
```

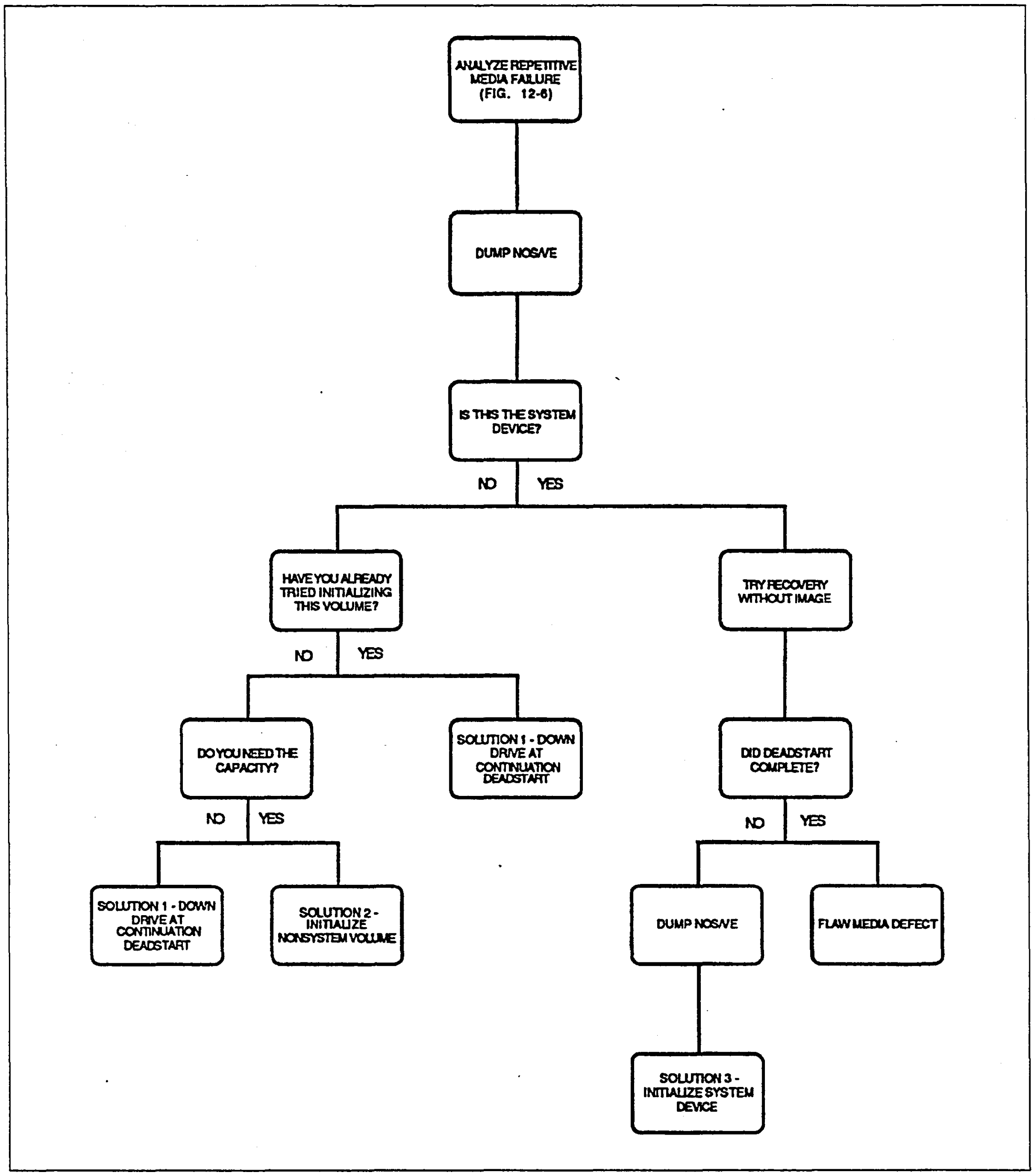

ANALYZE REPETITIVE MEDIA FAILURE (FIG. 12-6)
DUMP NOS/VE
IS THIS THE SYSTEM DEVICE?
NO
YES
HAVE YOU ALREADY TRIED INITIALIZING THIS VOLUME?
TRY RECOVERY WITHOUT IMAGE
NO
YES
DO YOU NEED THE CAPACITY?
SOLUTION 1 - DOWN DRIVE AT CONTINUATION DEADSTART
DID DEADSTART COMPLETE?
NO
YES
NO
YES
SOLUTION 1 - DOWN DRIVE AT CONTINUATION DEADSTART
SOLUTION 2 - INITIALIZE NONSYSTEM VOLUME
DUMP NOS/VE
FLAW MEDIA DEFECT
SOLUTION 3 - INITIALIZE SYSTEM DEVICE

Figure 12-6. Analyze Repetitive Media Failure

Analyze Critical Peripheral Failure

NOS/VE is in a stepped state due to the unavailability of a critical disk volume.

Move the cursor to the critical display window of the system console and expand the window (depress the SHIFT and F9 (or SUPER) keys at the same time).

Refer to Failure Data in Critical Display Window later in this chapter for information concerning mass storage peripheral failure data in the critical display window.

To determine how to answer the questions as to whether the failing element was a channel, controller, or a disk unit, you must locate the UNRECOVERED message group in the critical display window. If the third line of the message contains text such as:

```
xxxxxx DOWNED
```

NOS/VE has stopped accessing (disabled) one or more volumes due to a hardware failure (not a media failure). The words prior to the word DOWNED indicate the kind of peripheral that failed.

WAS "DRIVE DOWNED" REPORTED?

The failing element was a disk unit if the word DRIVE appears or if the word CONTROLLER appears and the failing element was an 887 disk unit.

WAS "CONTROLLER DOWNED" REPORTED?

The failing element was a disk controller if the words CONTROL MODULE or CYBER COUPLER appear.

WAS "CHANNEL DOWNED" REPORTED?

The failing element was a disk channel if the words CHANNEL or ADAPTER appear.

NOTE

Although the action statement in the critical display window indicates that an element was downed, the state of the faulty element may remain ON in the physical configuration.

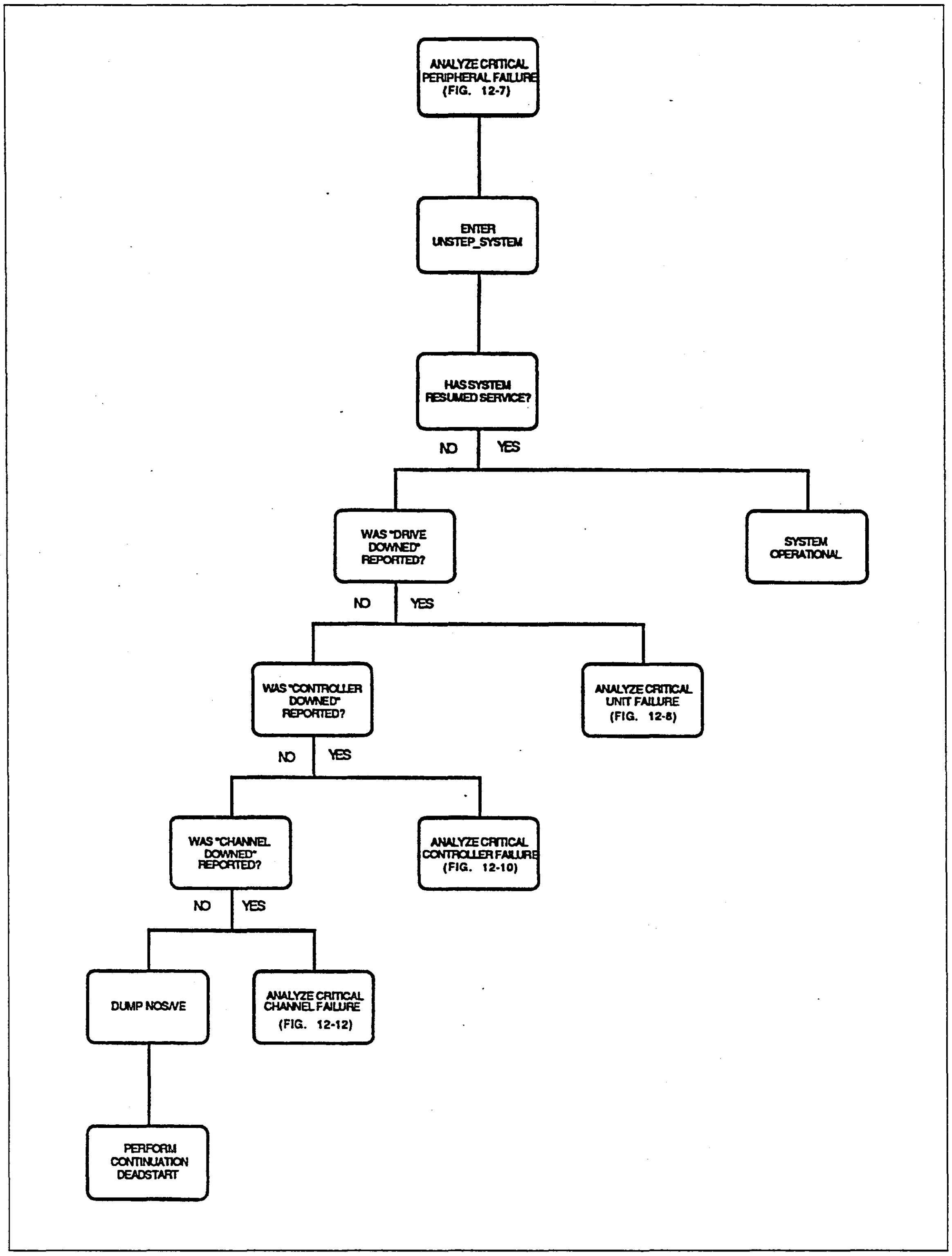

ANALYZE CRITICAL PERIPHERAL FAILURE (FIG. 12-7)
ENTER UNSTEP_SYSTEM
HAS SYSTEM RESUMED SERVICE?
NO
YES
WAS "DRIVE DOWNED" REPORTED?
SYSTEM OPERATIONAL
NO
YES
WAS "CONTROLLER DOWNED" REPORTED?
ANALYZE CRITICAL UNIT FAILURE (FIG. 12-8)
NO
YES
WAS "CHANNEL DOWNED" REPORTED?
ANALYZE CRITICAL CONTROLLER FAILURE (FIG. 12-10)
NO
YES
DUMP NOS/VE
ANALYZE CRITICAL CHANNEL FAILURE (FIG. 12-12)
PERFORM CONTINUATION DEADSTART

Figure 12-7. Analyze Critical Peripheral Failure

Analyze Critical Unit Failure

NOS/VE is in a stepped state because a critical disk unit failed. The state of the disk unit in the physical configuration is still ON; however, NOS/VE has stopped using the volume mounted on the disk unit.

VISUALLY INSPECT DRIVE SWITCH SETTINGS - ARE YOU ABLE TO FIX THE PROBLEM YOURSELF?

Refer to Inspect Critical Drive Switch Settings later in this chapter to determine whether you will be able to repair the disk unit problem yourself.

IS THIS THE SYSTEM DEVICE?

Determine from the physical address (channel, controller, and unit) whether this is the disk unit containing the NOS/VE system.

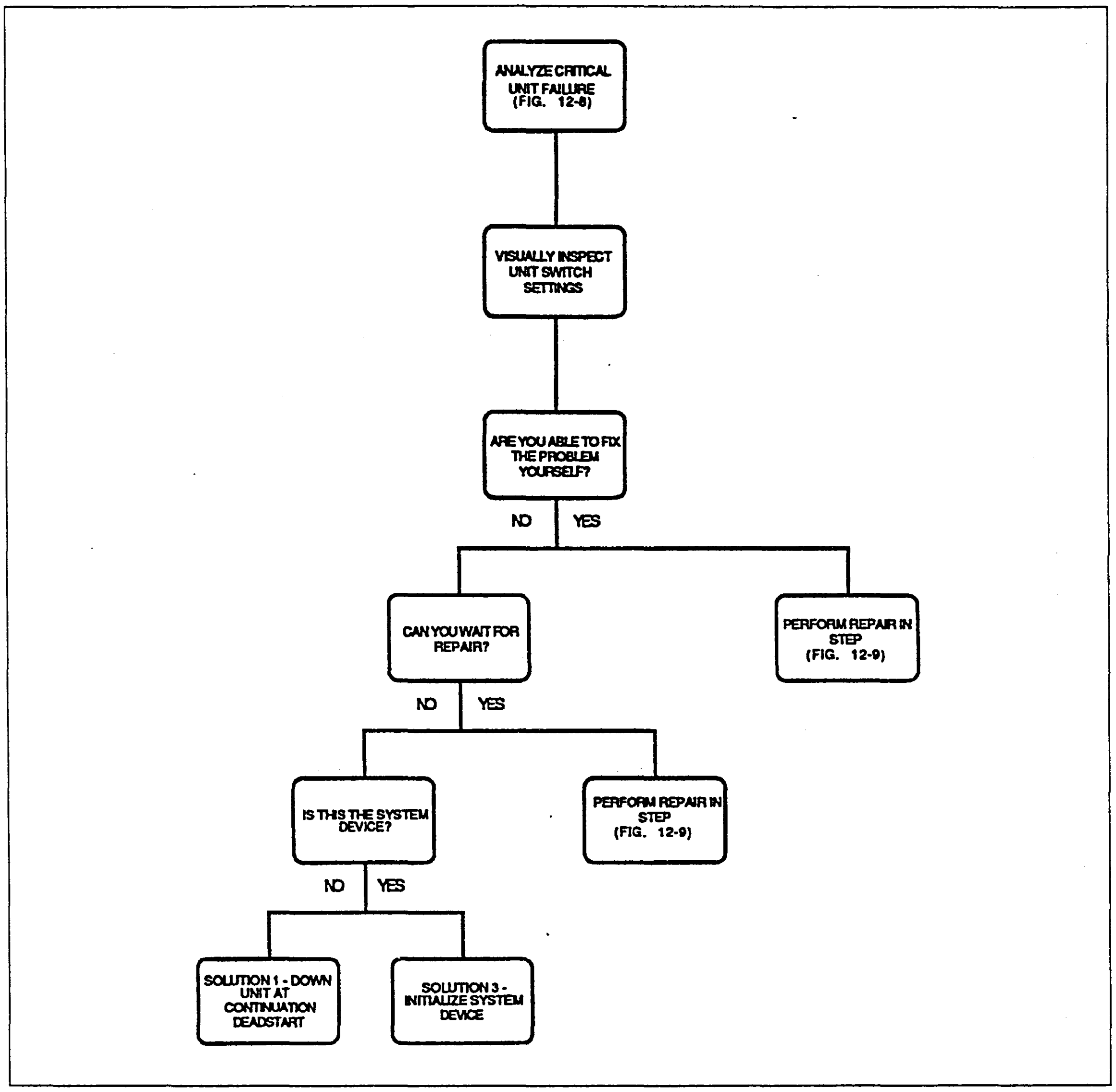

ANALYZE CRITICAL UNIT FAILURE (FIG. 12-8)
VISUALLY INSPECT UNIT SWITCH SETTINGS
ARE YOU ABLE TO FIX THE PROBLEM YOURSELF?
NO
YES
CAN YOU WAIT FOR REPAIR?
PERFORM REPAIR IN STEP (FIG. 12-9)
NO
YES
IS THIS THE SYSTEM DEVICE?
PERFORM REPAIR IN STEP (FIG. 12-9)
NO
YES
SOLUTION 1 - DOWN UNIT AT CONTINUATION DEADSTART
SOLUTION 3 - INITIALIZE SYSTEM DEVICE

Figure 12-8. Analyze Critical Unit Failure

Perform Repair in Step

NOS/VE is in a stepped state due to the failure of a critical disk element. The operator or the CE will attempt to fix the problem while the system is in a steped condition and then continue service after repair. While the system is in a stepped state, any disk controller or disk unit may be powered off, repaired, and powered back on. Subsequently, you may execute an UNSTEP_SYSTEM command to continue service. Refer to the NOS/VE Operations manual for more information on continuing service.

WARNING

The repair process must not:

- Modify central memory.
- Write outside of dedicated maintenance cylinders of the disk volume.
- Replace a disk volume.
- Physically reconfigure the system.

CALL CE IF NECESSARY FOR REPAIR

If the operator has access and authority to change a switch setting on a disk unit or a controller, this change should be made by the operator if a CE is not on site; otherwise, call a CE to perform the repair.

COMPLETE THE REPAIR OF ELEMENT

Either the operator or the CE should now perform the repair of the failing element.

ENTER: UNSTEP_SYSTEM

All elements disabled by the system prior to entering a stepped state are reenabled as a result of the UNSTEP_SYSTEM command. If the problem has been repaired, the system continues service; otherwise, the system reenters a stepped state. Furthermore, the UNSTEP_SYSTEM command causes all mass storage and $5698_1x drivers to execute a confidence test of each path and unit of this type in the configuration.

If the repair was not successful, the system should go back into a stepped condition within 2 to 8 minutes. If a fault in a 9836 or 9853 controller or disk unit was not corrected, it may take the NOS/VE driver that long to determine that a controller is malfunctioning (or powered off).

HAS SYSTEM RESUMED SERVICE?

If you are able to enter operator commands, then the UNSTEP_SYSTEM command was successful.

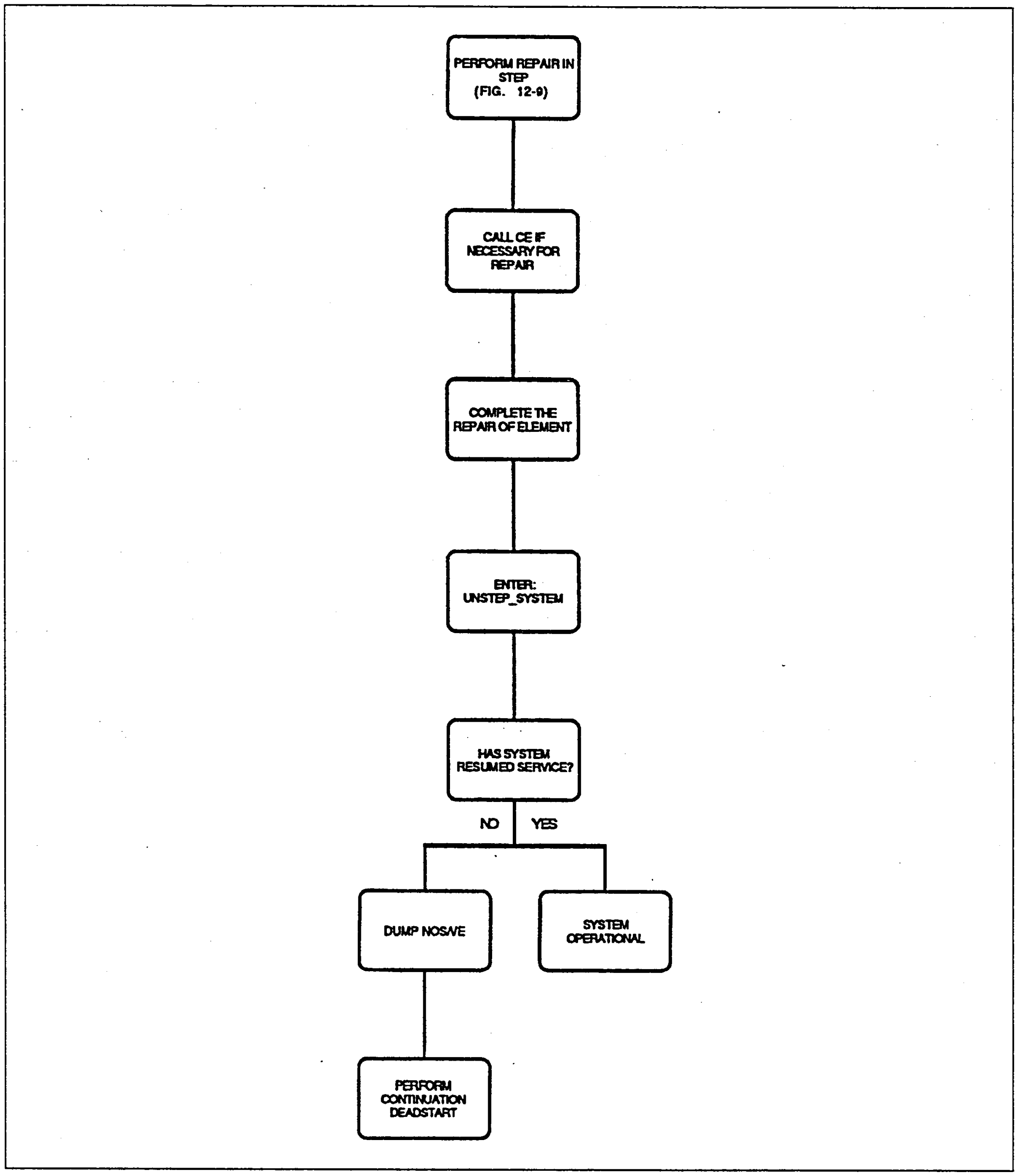

PERFORM REPAIR IN STEP (FIG. 12-9)
CALL CE IF NECESSARY FOR REPAIR
COMPLETE THE REPAIR OF ELEMENT
ENTER: UNSTEP_SYSTEM
HAS SYSTEM RESUMED SERVICE?
NO
YES
DUMP NOS/VE
SYSTEM OPERATIONAL
PERFORM CONTINUATION DEADSTART

Figure 12-9. Perform Repair in Step

Analyze Critical Controller Failure

NOS/VE is in a stepped state because a critical disk controller failed. The disk unit(s) affected were disabled by NOS/VE. The controller and the disk units remain in the ON state in the NOS/VE physical configuration.

VISUALLY INSPECT CONTROLLER SWITCH SETTINGS - ARE YOU ABLE TO FIX THE PROBLEM YOURSELF?

Refer to Failure Data in Critical Display Window later in this chapter for more information on the presentation of disk failure data in the critical display window.

ARE THE UNITS CONNECTED TO THE FAILING CONTROLLER DUAL ACCESS?

Determine if another controller is physically cabled to all the critical disk units that are connected to the failing controller.

IS THE SYSTEM DEVICE CONNECTED TO THE FAILING CONTROLLER?

Determine from the physical address (channel, controller, and unit) whether this is the controller providing the only access to the system device.

895 UNITS INVOLVED?

The 7165-2x product has two CYBER channel couplers (CCCs). Each CCC is connected to one CYBER channel and one storage director. Therefore, if a 7165-2x controller is configured to two channels of the same mainframe and there are no other controllers connected to these channels, downing the controller disables both channels and all the volumes connected to those channels.

If a problem with one CCC or storage director of the 7165-2x is suspected, we recommend that you stop using the channel that provides access to those elements. During the continuation deadstart, use the PCU subutility EDIT_PHYSICAL_CONFIGURATION and the subcommand CHANGE_ELEMENT_DEFINITION to remove the IOU_CONNECTION parameter description of the channel. Refer to chapter 2, Physical Configuration Utility, for more information on changing the element definition.

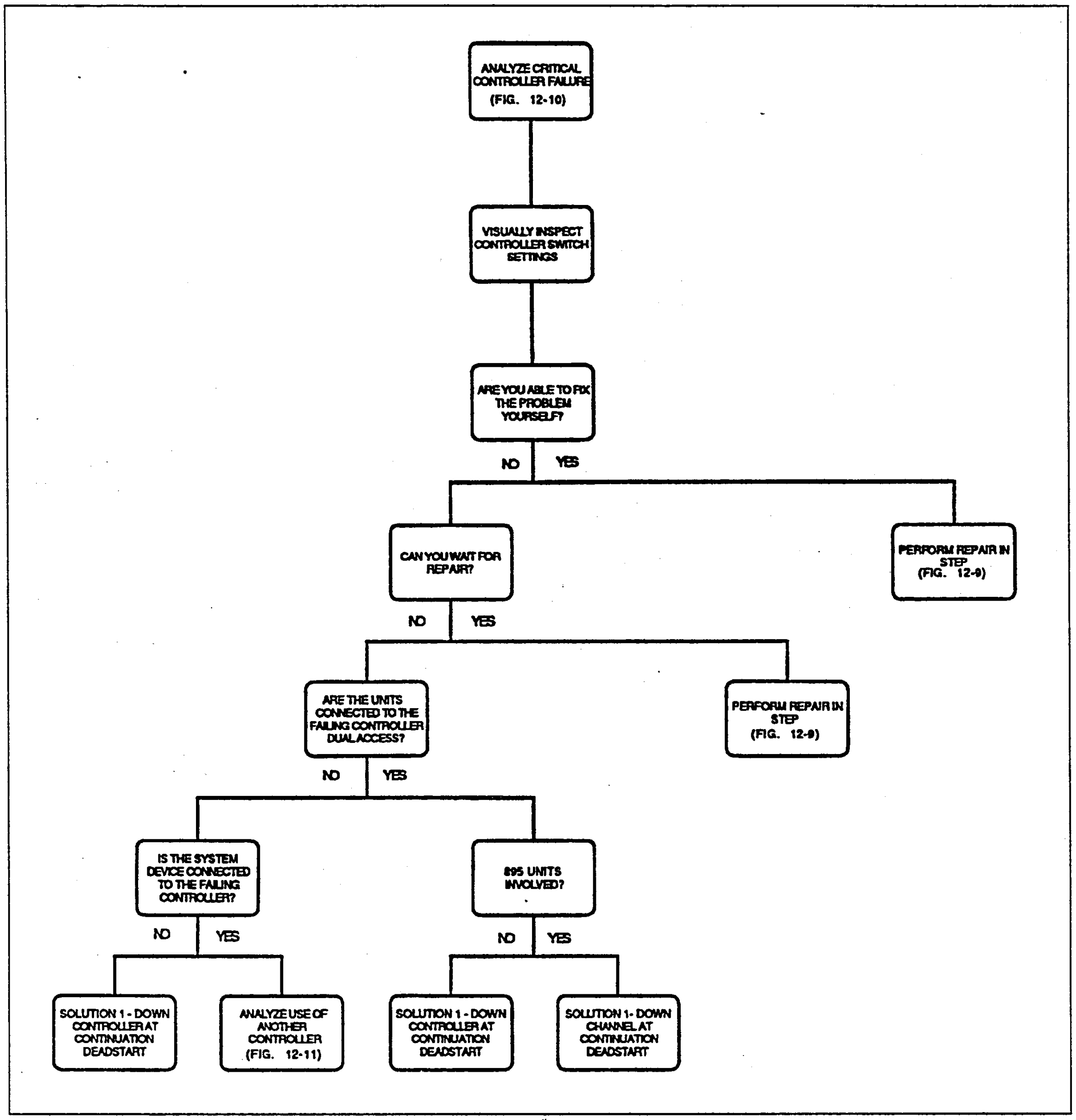

ANALYZE CRITICAL CONTROLLER FAILURE (FIG. 12-10)
VISUALLY INSPECT CONTROLLER SWITCH SETTINGS
ARE YOU ABLE TO FIX THE PROBLEM YOURSELF?
NO
YES
CAN YOU WAIT FOR REPAIR?
PERFORM REPAIR IN STEP (FIG. 12-9)
NO
YES
ARE THE UNITS CONNECTED TO THE FAILING CONTROLLER DUAL ACCESS?
PERFORM REPAIR IN STEP (FIG. 12-9)
NO
YES
IS THE SYSTEM DEVICE CONNECTED TO THE FAILING CONTROLLER?
895 UNITS INVOLVED?
NO
YES
NO
YES
SOLUTION 1 - DOWN CONTROLLER AT CONTINUATION DEADSTART
ANALYZE USE OF ANOTHER CONTROLLER (FIG. 12-11)
SOLUTION 1 - DOWN CONTROLLER AT CONTINUATION DEADSTART
SOLUTION 1- DOWN CHANNEL AT CONTINUATION DEADSTART

Figure 12-10. Analyze Critical Controller Failure

Analyze Use of Another Controller

NOS/VE is in a stepped state because a disk controller failed. Because the system device is only connected to the failing controller and that controller cannot be repaired soon enough, you can either ask a CE to recable the system device to another controller or you can reinstall the system to another disk unit on a second controller. The latter requires a full reload of permanent files. If you have only one controller, it must be repaired before you can resume service.

DO YOU HAVE ANOTHER DISK CONTROLLER?

If the system device is an 836 or 887 disk unit, the answer to this question is no because the controller connects only to one disk unit.

For all other disk unit products, determine if you have more than one controller that can be connected to the system device. If you do, a CE can recable the system device to the alternative controller (it may be necessary to disconnect one of the existing disk units from the controller to make connection to the system device possible).

DO YOU HAVE ONSITE CE COVERAGE?

If you have a resident CE, you may investigate the possibilities of recabling the inaccessible system device to an alternate controller. Otherwise, you must define a new system device or wait for repair to be scheduled.

WOULD RECABLING SYSTEM DEVICE BE FASTER THAN REPAIR?

If the CE is unable to find the fault initially, it may be possible for the CE to recable the system device to another controller. The CE may then perform online repair of the other controller.

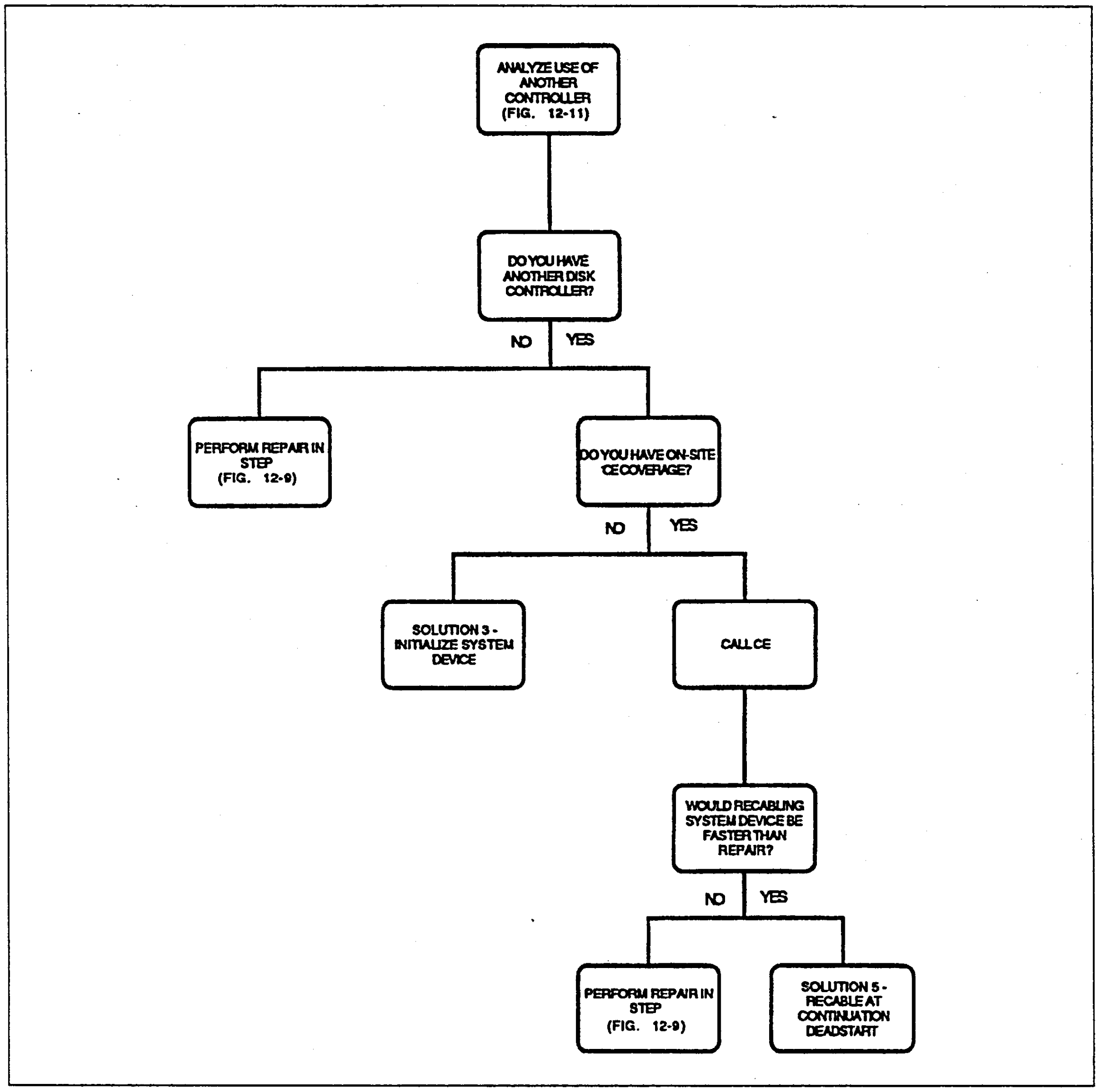

ANALYZE USE OF ANOTHER CONTROLLER (FIG. 12-11)
DO YOU HAVE ANOTHER DISK CONTROLLER?
NO
YES
PERFORM REPAIR IN STEP (FIG. 12-9)
DO YOU HAVE ON-SITE CE COVERAGE?
NO
YES
SOLUTION 3 - INITIALIZE SYSTEM DEVICE
CALL CE
WOULD RECABLING SYSTEM DEVICE BE FASTER THAN REPAIR?
NO
YES
PERFORM REPAIR IN STEP (FIG. 12-9)
SOLUTION 5 - RECABLE AT CONTINUATION DEADSTART

Figure 12-11. Analyze Use of Another Controller

Analyze Critical Channel Failure

NOS/VE is in a stepped state because a disk channel failed. The channel had provided access to a disk unit belonging to class J or Q. The disk units that were affected were disabled by NOS/VE. The channel and the disk units are still on in the NOS/VE configuration.

ARE THE FAILING ELEMENTS CONNECTED TO ANOTHER CHANNEL?

The term *element* refers to either a controller or a disk unit, depending upon the product involved. In either case, are any of the elements that are connected directly to the channel also connected to another channel? The channel may be a redundant channel or an alternate channel. The channel need not be described in the NOS/VE physical configuration because the physical configuration may be edited at a continuation deadstart. Refer to Alternate and Redundant Path Configurations in appendix E, Supported Hardware Products, for more information on access.

IS THE SYSTEM DEVICE CONNECTED TO THE FAILING CHANNEL?

Determine from the physical address (channel, controller, and unit) whether this is the channel providing the only access to the system device.

DO YOU HAVE ANOTHER DISK SUBSYSTEM?

If you have another disk subsystem (different disk channel/controller), you have a choice of reinstalling the system on a disk unit in the other subsystem and restoring all files or waiting for repair.

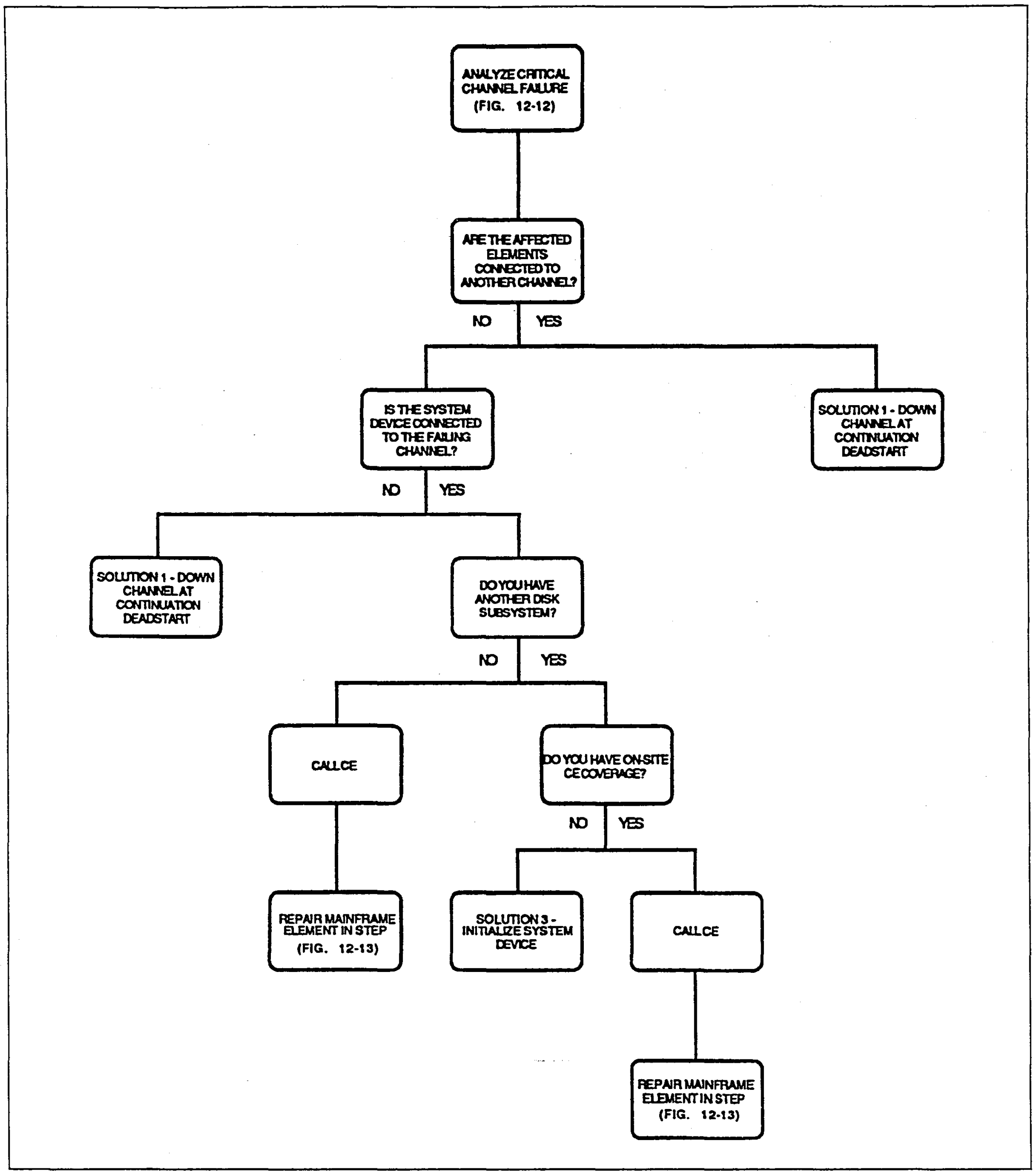

ANALYZE CRITICAL CHANNEL FAILURE (FIG. 12-12)
ARE THE AFFECTED ELEMENTS CONNECTED TO ANOTHER CHANNEL?
NO
YES
IS THE SYSTEM DEVICE CONNECTED TO THE FAILING CHANNEL?
SOLUTION 1 - DOWN CHANNEL AT CONTINUATION DEADSTART
NO
YES
SOLUTION 1 - DOWN CHANNEL AT CONTINUATION DEADSTART
DO YOU HAVE ANOTHER DISK SUBSYSTEM?
NO
YES
CALL CE
DO YOU HAVE ON-SITE CE COVERAGE?
NO
YES
REPAIR MAINFRAME ELEMENT IN STEP (FIG. 12-13)
SOLUTION 3 - INITIALIZE SYSTEM DEVICE
CALL CE
REPAIR MAINFRAME ELEMENT IN STEP (FIG. 12-13)

Figure 12-12. Analyze Critical Channel Failure

Repair Mainframe Element in Step

NOS/VE is in a stepped state because a critical mainframe component or channel failed. If a channel was faulty, it provided the only access to the system device and the operator has chosen to call the CE rather than reinstall NOS/VE to a different system device.

CAN CE PERFORM REPAIR WITH MAINFRAME POWER ON?

If a peripheral cable is faulty, the CE can repair the problem while the system is in a stepped state. However, if the problem is in the mainframe or in the input output unit (IOU), it may be necessary to power off the mainframe or destroy the contents of central memory to perform repair.

If the data in central memory is now intact but would be lost while performing repair, it may be possible to save central memory on a magnetic tape, repair the hardware problem, and restore central memory from magnetic tape at the beginning of the next continuation deadstart.

If mainframe power has been lost or the backup of central memory cannot be performed due to the fault, any modified pages that were in memory at the time of the failure are lost and active jobs are not recovered. This may cause files that were attached for write access at the time of the failure to become inconsistent. Refer to Finding Inconsistent Files in chapter 13, Repair Solutions, for more information on file recovery.

DUMP NOS/VE

If you have not yet been asked to dump NOS/VE during the analysis of this failure, do so now.

PERFORM CENTRAL MEMORY RELOAD

Refer to Terminology in chapter 11, Fault Tolerance, for further information on reloading central memory.

WARNING

A central memory reload can only be reliably performed on the deadstart immediately following the repair of the mainframe.

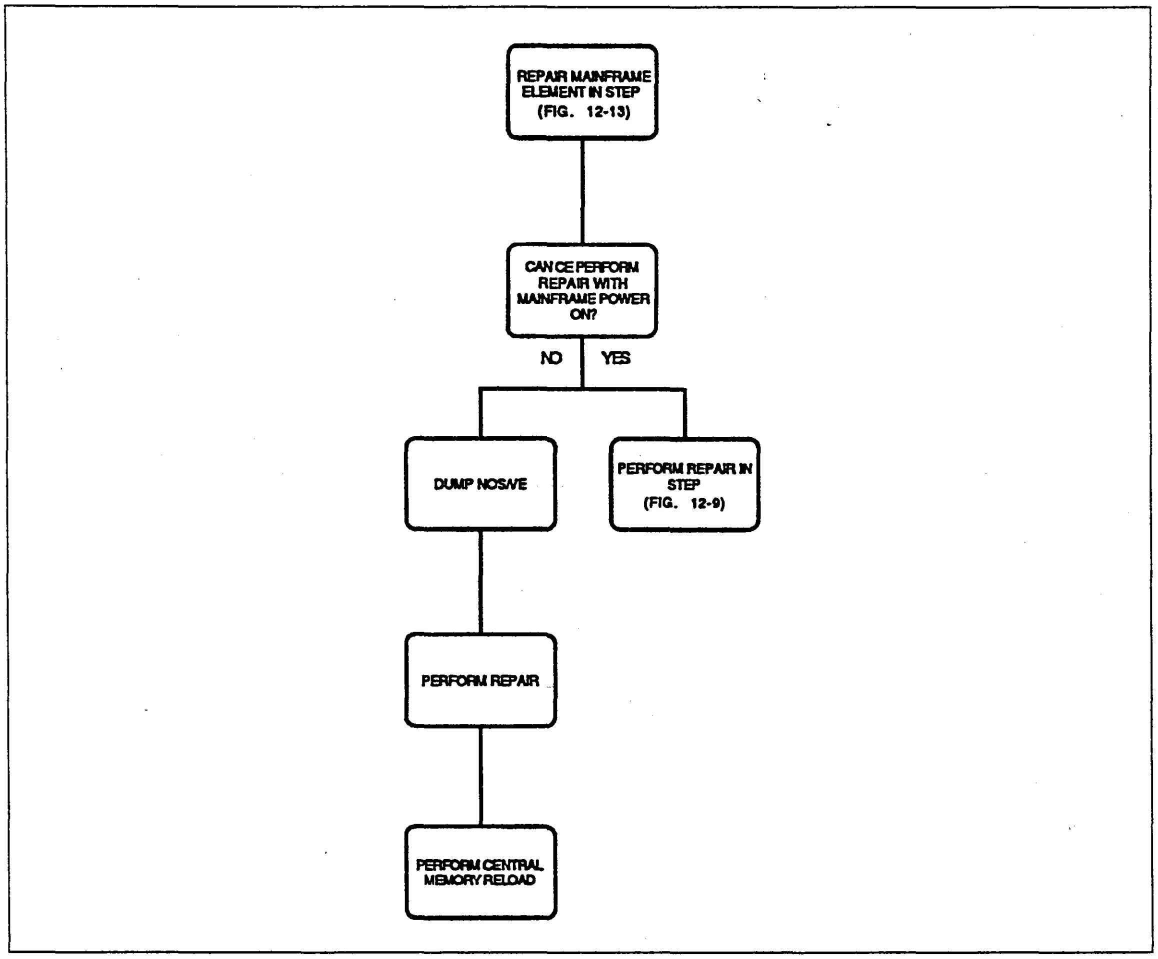

REPAIR MAINFRAME ELEMENT IN STEP (FIG. 12-13)
CAN CE PERFORM REPAIR WITH MAINFRAME POWER ON?
NO
YES
DUMP NOS/VE
PERFORM REPAIR IN STEP (FIG. 12-9)
PERFORM REPAIR
PERFORM CENTRAL MEMORY RELOAD

Figure 12-13. Repair Mainframe Element in Step

Analyze Noncritical Peripheral Failure

NOS/VE is executing jobs but the performance of the system may have degraded noticeably. The operator has confirmed that the system is not in a stepped state and that the problem is not related to the use of magnetic tape equipment. A noncritical mass storage peripheral may be at fault. Although NOS/VE always writes failure data to the critical display window of the system console for an uncorrectable hardware fault (not a media defect), it is possible for this information to scroll off this window before the operator has an opportunity to see it.

The following analyses of noncritical disk failures assumes that the failure data is no longer visible in the critical display window; therefore, other isolation techniques are provided. To be certain that the failure data is no longer visible, move the cursor to the critical display window of the system console and expand the window (press the SHIFT and F9 (or SUPER) keys at the same time). Refer to Failure Data in Critical Display Window later in this chapter for information about mass storage peripheral failure data in the critical display window. Locate the UNRECOVERED message group in the critical display window. If you can find such a message, you may be able to determine which channel, controller, or disk unit is faulty. Refer to Analyze Critical Peripheral Failure in this chapter to determine what to look for in the critical display window. If you are able to determine from the message which element is faulty, return to this analysis and use the information to help answer the following questions.

WERE YOU ABLE TO SEE THE DISPLAY?

If you cannot see the VED MS display the system is inoperable, although some jobs may be executing normally. To use the VED MS display, you must have the SYSTEM_OPERATOR or SYSTEM_DISPLAYS capability and must execute from within the System Operator Utility (SOU).

"CLASSES OUT OF SPACE" DISPLAYED?

Look at the line immediately below the line containing the column headings in the VED MS display. Look for a line like the following: "Classes out of space: M N."

"CLASS Q DEVICES AUTOMATICALLY ADDED" DISPLAYED?

Look at the first two lines below the line containing the column headings in the VED MS display. Look for a line like the following: "Class Q devices automatically added: x." The value x implies that x additional volumes have automatically been assigned to class Q since the last deadstart.

ARE ANY VOLUMES UNAVAILABLE IN THE DISPLAY?

If a volume is unavailable, the words "Volume is not available" appear next to the name of the volume (RECORDED_VSN) and the other information about the volume is suppressed. Scroll the display to ensure all the volumes are available.

END OF ANALYSIS

Either the analysis of the fault provided by this manual is inadequate or there was no fault in the system. If you suspect the NOS/VE system software is at fault, take a dump and contact Control Data.

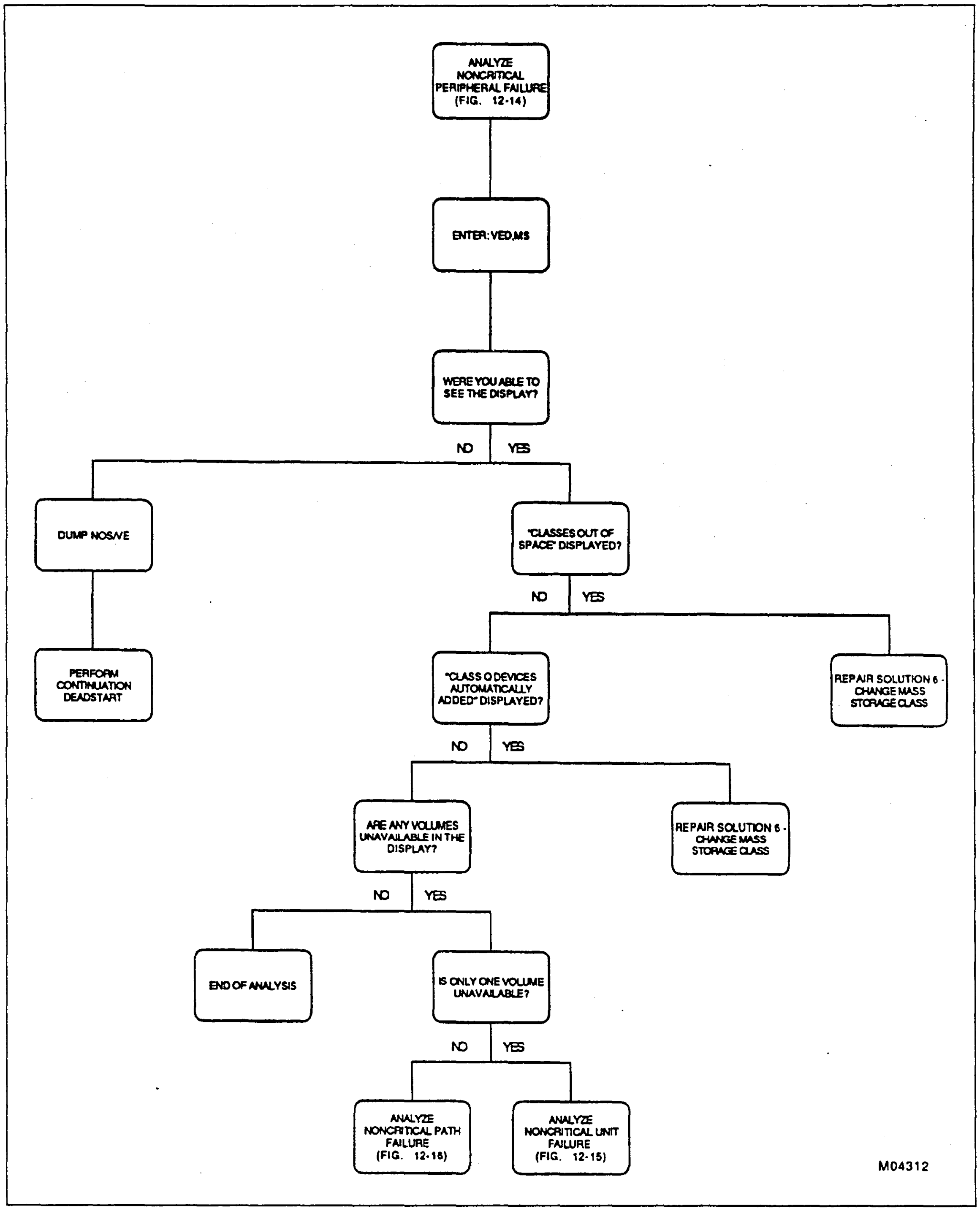

ANALYZE NONCRITICAL PERIPHERAL FAILURE (FIG. 12-14)
ENTER: VED,MS
WERE YOU ABLE TO SEE THE DISPLAY?
NO
YES
DUMP NOS/VE
PERFORM CONTINUATION DEADSTART
"CLASSES OUT OF SPACE" DISPLAYED?
NO
YES
"CLASS O DEVICES AUTOMATICALLY ADDED" DISPLAYED?
REPAIR SOLUTION 6 - CHANGE MASS STORAGE CLASS
NO
YES
ARE ANY VOLUMES UNAVAILABLE IN THE DISPLAY?
REPAIR SOLUTION 6 - CHANGE MASS STORAGE CLASS
NO
YES
END OF ANALYSIS
IS ONLY ONE VOLUME UNAVAILABLE?
NO
YES
ANALYZE NONCRITICAL PATH FAILURE (FIG. 12-16)
ANALYZE NONCRITICAL UNIT FAILURE (FIG. 12-15)
M04312

Figure 12-14. Analyze Noncritical Peripheral Failure

Analyze Noncritical Unit Failure

NOS/VE is not in a stepped state and the operator has determined that one disk volume has been disabled or that multiple disk volumes that are not connected to the same channel have been disabled. If only one volume is unavailable, the fault will be assumed to be with the disk unit rather than its controller. However, if the controller is only cabled to one disk unit, as is required with the 836 and 887 products, the effect of the fault is the same whether it is with the disk unit or the controller.

DOWN UNIT USING CHANGE_ELEMENT_STATE (CHAES)

To change the state of the disk unit to ON or to DOWN, enter the following commands from the System Operator Utility (SOU):

```
SOU/lcu
LCU/change_element_state element=<name> state=<yyyy>
LCU/quit
```

where name is the name of the disk unit and yyyy is either ON or DOWN.

VISUALLY INSPECT AND CHANGE SWITCH SETTINGS

If someone has changed the switch settings for a disk unit or controller during production, this mistake can be corrected while the disk unit is down. The disk unit may then be turned back on.

If NOS/VE noticed the change of the switch setting, this was reported to the critical display window of the system console. The failure analysis for the disk unit may still be visible in the window. If it is, refer to Inspect Critical Drive Switch Settings at the end of this chapter to determine which switch may have been changed; otherwise, go to the disk unit and make the inspection discussed for the disk unit products in Inspect Noncritical Disk Unit Switch Settings later in this chapter.

DOES VED MS STILL SHOW VOLUME UNAVAILABLE?

All the jobs that were blocked by the unavailable volume should resubmit their requests to the disk unit within approximately 30 seconds after you turn the disk unit back on. However, it may take longer than this for all the affected jobs to swap back in; waiting 2 minutes makes it more likely that all the requests were resubmitted before concluding the fault has disappeared.

DETERMINE VOLUME'S CLASS MEMBERSHIP

Determine the class or classes to which the volume has been assigned since the last deadstart. From the System Operator Utility (SOU), enter the following commands to obtain a display of the volume's current mass storage class membership:

```
SOU/lcu
LCU/display_ms_class
LCU/quit
```

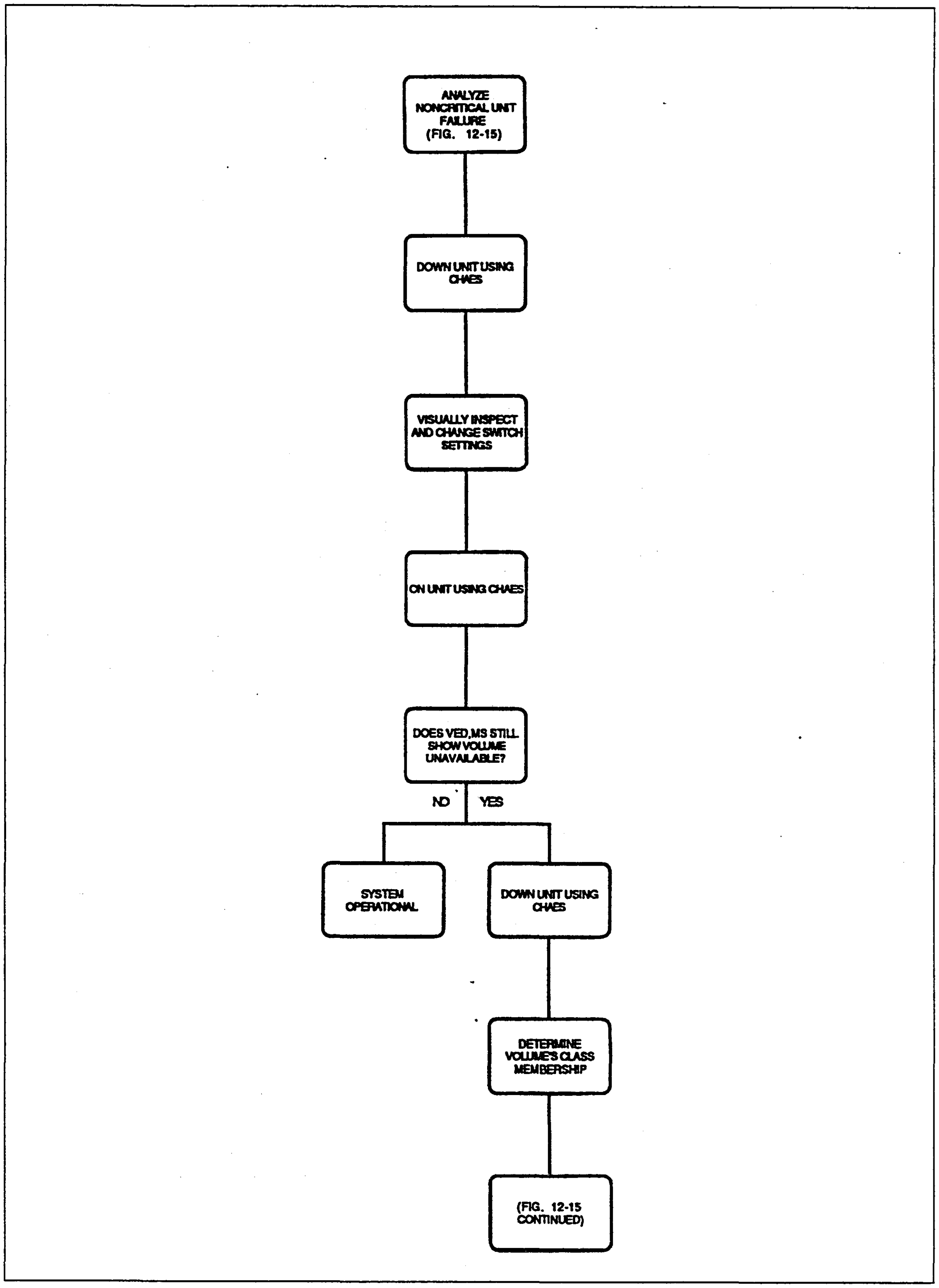

ANALYZE NONCRITICAL UNIT FAILURE (FIG. 12-15)
DOWN UNIT USING CHAES
VISUALLY INSPECT AND CHANGE SWITCH SETTINGS
ON UNIT USING CHAES
DOES VED,MS STILL SHOW VOLUME UNAVAILABLE?
NO
YES
SYSTEM OPERATIONAL
DOWN UNIT USING CHAES
DETERMINE VOLUME'S CLASS MEMBERSHIP
(FIG. 12-15 CONTINUED)

Figure 12-15. Analyze Noncritical Unit Failure

(Continued)

(Continued)

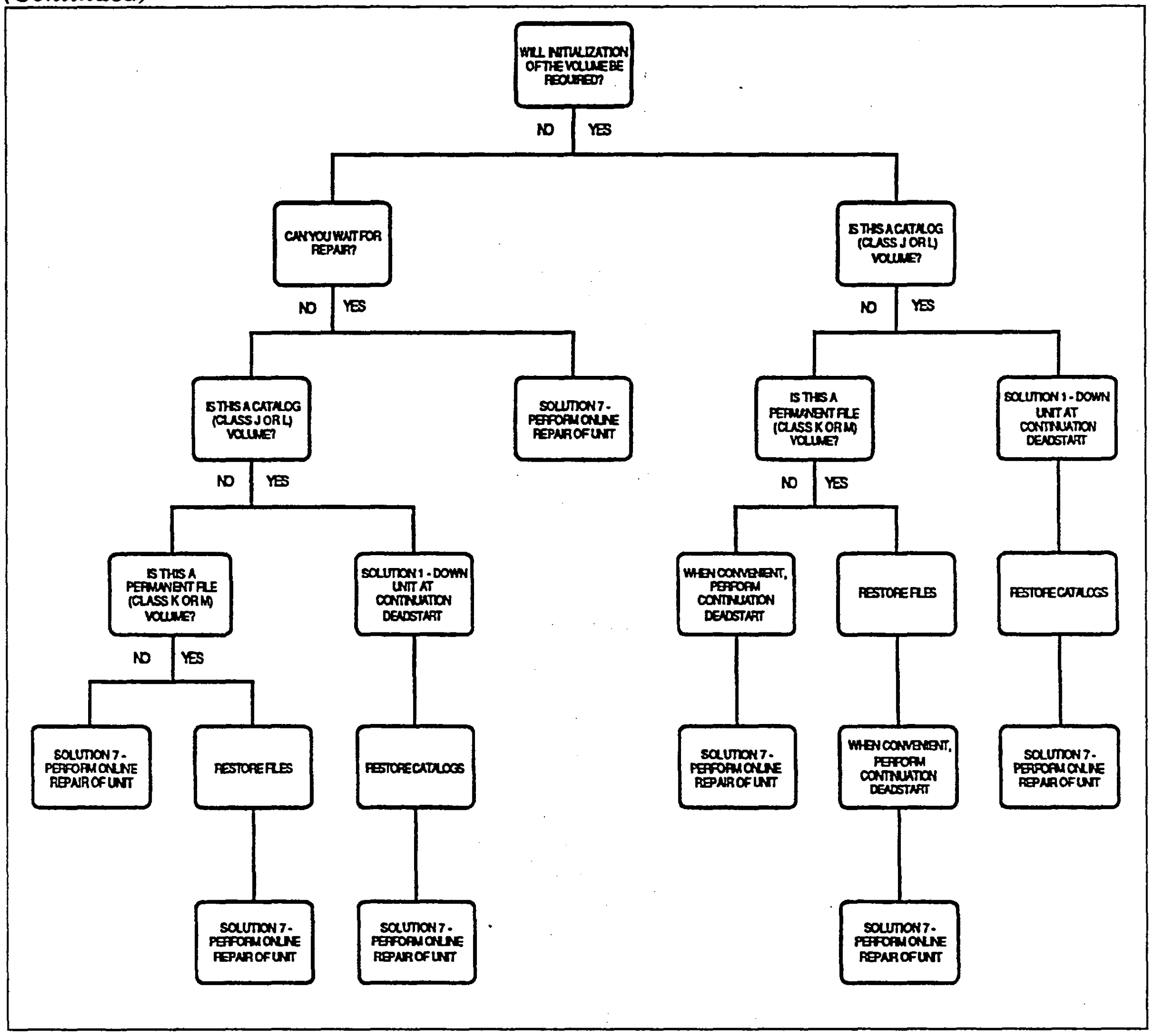

WILL INITIALIZATION OF THE VOLUME BE REQUIRED?
NO
YES
CAN YOU WAIT FOR REPAIR?
IS THIS A CATALOG (CLASS J OR L) VOLUME?
NO
YES
NO
YES
IS THIS A CATALOG (CLASS J OR L) VOLUME?
SOLUTION 7 - PERFORM ONLINE REPAIR OF UNIT
IS THIS A PERMANENT FILE (CLASS K OR M) VOLUME?
SOLUTION 1 - DOWN UNIT AT CONTINUATION DEADSTART
NO
YES
NO
YES
IS THIS A PERMANENT FILE (CLASS K OR M) VOLUME?
SOLUTION 1 - DOWN UNIT AT CONTINUATION DEADSTART
WHEN CONVENIENT, PERFORM CONTINUATION DEADSTART
RESTORE FILES
RESTORE CATALOGS
NO
YES
SOLUTION 7 - PERFORM ONLINE REPAIR OF UNIT
RESTORE FILES
RESTORE CATALOGS
SOLUTION 7 - PERFORM ONLINE REPAIR OF UNIT
WHEN CONVENIENT, PERFORM CONTINUATION DEADSTART
SOLUTION 7 - PERFORM ONLINE REPAIR OF UNIT
SOLUTION 7 - PERFORM ONLINE REPAIR OF UNIT
SOLUTION 7 - PERFORM ONLINE REPAIR OF UNIT
SOLUTION 7 - PERFORM ONLINE REPAIR OF UNIT

Figure 12-15. Analyze Noncritical Unit Failure

Note which of the following classes are currently assigned to the volumes C, K, L, M, and N.

If you have removed one or more mass storage classes from this volume since the last deadstart, you need to know which classes were deleted to proceed reliably in this analysis. If your site followed the recommendations of site preparation in chapter 11, Fault Tolerance, you created a file that contained the mass storage class assignments in effect at the last deadstart.

Enter the following command at the system console to determine what classes were assigned to the volume at the last deadstart. If this file does not exist, proceed with the analysis using the class membership already determined.

```
copy_file $system.mainframe.classes_at_deadstart
```

Note which of the following classes were previously assigned to the volumes C, J, K, L, M, N, and Q.

WILL INITIALIZATION OF THE VOLUME BE REQUIRED?

If you have onsite CE coverage, the CE may have already analyzed the fault and told you that the volume's data has been damaged or the volume (Head Disk Assembly) requires replacement. If this occurred, the volume requires initialization.

If you have no CE onsite or the fault is not yet isolated, assume that the volume does not require initialization.

CAN YOU WAIT FOR REPAIR?

If you must return to production before the fault can be repaired, you may have to restore files or catalogs and possibly perform a continuation deadstart. If any of the unavailable files or catalogs were modified since your last backup, you lose the changes to these files or catalogs if you restore them. If at all possible, wait for the repair.

IS THIS A CATALOG (CLASS J OR L) VOLUME?

A continuation deadstart is required to either initialize a catalog volume or to restore missing catalogs to other volumes.

IS THIS A PERMANENT FILE (CLASS K OR M) VOLUME?

If the volume requires initialization, or if you must have access to the files that were assigned to the volume in order to continue production, you may restore files without waiting for the disk unit's repair. However, if you restore files, modifications made to any of these files since the last file backup are lost.

WHEN CONVENIENT, PERFORM CONTINUATION DEADSTART

NOS/VE does not allow you to initialize a volume that has been active since the last deadstart. You must perform a continuation deadstart before NOS/VE allows the volume to be initialized, but you may not need to do this immediately.

If the volume's capacity is required for meaningful use of the system, you perform a continuation deadstart as soon as the fault is repaired. If you can use the system productively without the volume, defer the continuation deadstart until a scheduled time.

You may perform the continuation deadstart either before or after the fault is repaired. If you perform the continuation deadstart before the fault is repaired, leave the disk unit in the DOWN state and use repair solution 7 from chapter 13, Repair Solutions, to reinstate the disk unit after it is repaired. You must initialize the disk unit in the same LCU session during which you turn on the disk unit or another deadstart is required to initialize the volume.

If you perform the continuation deadstart after the volume is repaired, you must turn on the disk unit and initialize it during the first LCU intervention during that deadstart. Otherwise, another deadstart is required to initialize the volume. Refer to repair solution 2 in chapter 13, Repair Solutions, for more information about initializing a volume during deadstart.

RESTORE FILES

Refer to Restoring Files in chapter 13, Repair Solutions, for more information. Follow the recommendations for restoring unavailable files.

RESTORE CATALOGS

Refer to Restoring Catalogs in chapter 13, Repair Solutions, for more information.

This page intentionally left blank.

Analyze Noncritical Path Failure

NOS/VE is not in a stepped state and the operator has determined that more than one disk unit has been disabled. A disk channel or controller may be the cause.

ARE ALL UNAVAILABLE VOLUMES ON THE SAME CHANNEL?

If all the disk units are on the same channel, it is likely that the problem is channel-related and unlikely to be disk unit-related.

ARE ALL VOLUMES ON THE CHANNEL UNAVAILABLE?

If the channel is connected to at least one disk unit that is available, the problem is unlikely to be a channel problem.

ARE 834, 9836, OR 9853 UNITS INVOLVED?

Because these products can have several controllers per channel, it is possible that all the disk units that are unavailable are being accessed through the same controller and the controller may be at fault.

DISPLAY AND ANALYZE SYSTEM CONFIGURATION

From the Sytem Operator Utility (SOU), enter:

```
SOU/display_system_configuration o=$local.display
SOU/print_file $local.display
```

Find the disk units that have volumes unavailable in the printed display. The display for each disk unit identifies the name of each controller that is cabled to it.

ARE ALL UNAVAILABLE VOLUMES ON THE SAME CONTROLLER? - ARE ALL VOLUMES ON THIS CONTROLLER UNAVAILABLE?

If all the disk units connected to the same controller are unavailable, it is likely that the controller is the cause of the problem.

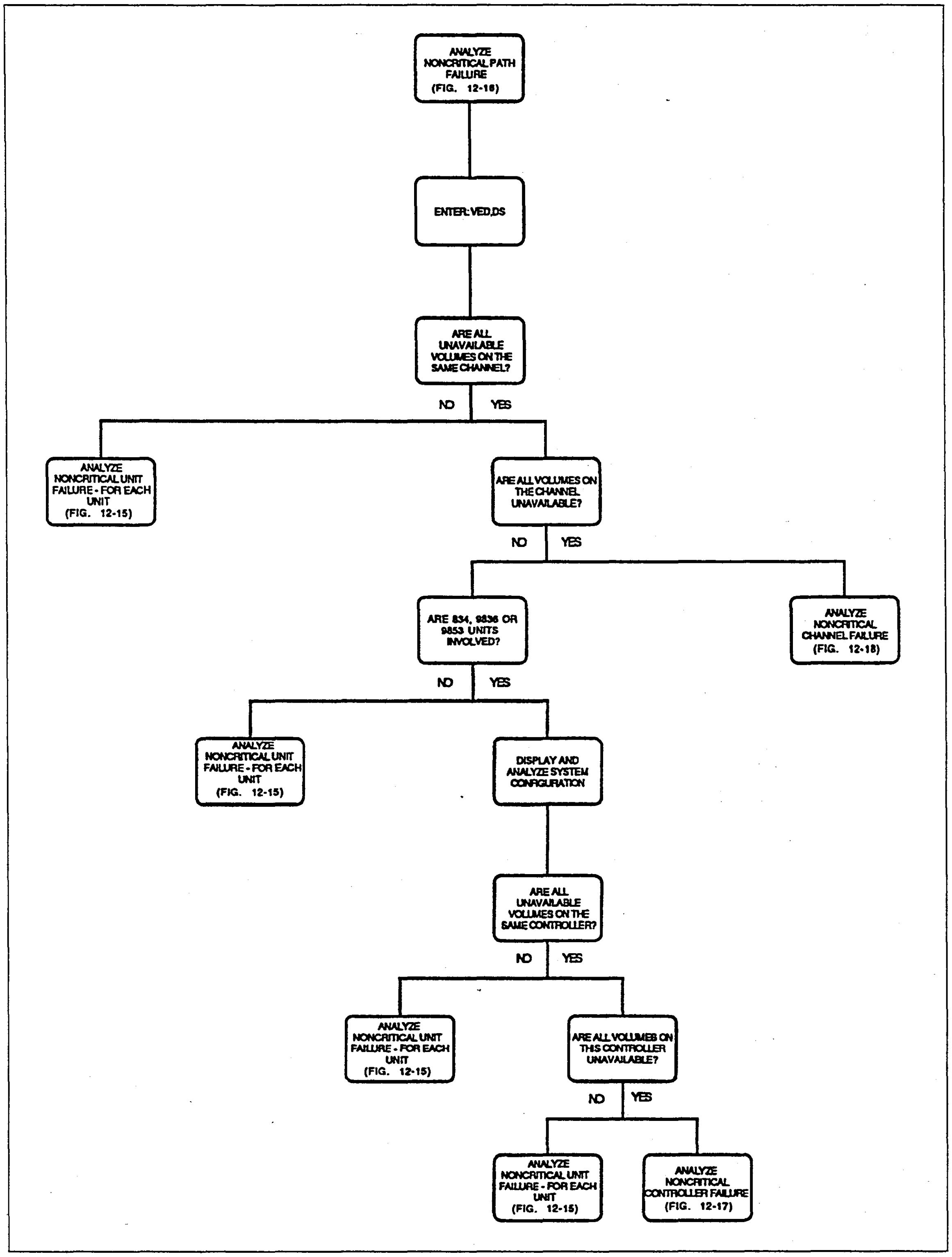

ANALYZE NONCRITICAL PATH FAILURE (FIG. 12-16)
ENTER: VED,DS
ARE ALL UNAVAILABLE VOLUMES ON THE SAME CHANNEL?
NO
YES
ANALYZE NONCRITICAL UNIT FAILURE - FOR EACH UNIT (FIG. 12-15)
ARE ALL VOLUMES ON THE CHANNEL UNAVAILABLE?
NO
YES
ARE 834, 9836 OR 9853 UNITS INVOLVED?
ANALYZE NONCRITICAL CHANNEL FAILURE (FIG. 12-18)
NO
YES
ANALYZE NONCRITICAL UNIT FAILURE - FOR EACH UNIT (FIG. 12-15)
DISPLAY AND ANALYZE SYSTEM CONFIGURATION
ARE ALL UNAVAILABLE VOLUMES ON THE SAME CONTROLLER?
NO
YES
ANALYZE NONCRITICAL UNIT FAILURE - FOR EACH UNIT (FIG. 12-15)
ARE ALL VOLUMES ON THIS CONTROLLER UNAVAILABLE?
NO
YES
ANALYZE NONCRITICAL UNIT FAILURE - FOR EACH UNIT (FIG. 12-15)
ANALYZE NONCRITICAL CONTROLLER FAILURE (FIG. 12-17)

Figure 12-16. Analyze Noncritical Path Failure

Analyze Noncritical Controller Failure

NOS/VE is not in a stepped state and the operator has determined that more than one 834, 9836, or 9853 disk unit has been disabled. A disk controller may be the cause.

DOWN CONTROLLER USING CHANGE_ELEMENT_STATE (CHAES)

The state changes are accepted because all the units that are connected to this controller are unavailable and the system is not in step; therefore, no critical volumes are connected only to this controller. From the System Operator Utility (SOU), enter the following commands (where *name* is the name of the controller):

```
SOU/lcu
LCU/change_element_state element=<name> state=down
LCU/quit
```

DID VOLUMES REMAIN UNAVAILABLE?

NOS/VE automatically reconfigures to a redundant controller if one is available. If the volumes remain unavailable, there is no redundant access.

VISUALLY INSPECT AND CHANGE SWITCH SETTINGS

If someone has changed the switch settings for a controller during production, you can correct this while the controller is down and then turn the controller back on.

Go to the disk controller and perform the following inspection:

- If the controller is a 7155-1x controller, ensure that the channel enable/disable switch for the channel access is in the enabled position.
- If the controller is a 7165-2x controller, ensure that the CCC's knob is in the online position and the storage director's enable/disable switch is in the enabled position.
- If the disk controller has been powered off, you know where the power switch is, and you have access to it, turn on the power.

ON CONTROLLER USING CHANGE_ELEMENT_STATE (CHAES)

If this is a 9836 or 9853 controller, wait up to 8 minutes for the state change to ON to complete; it may take this long to diagnose a controller that has no power or is faulty. From the System Operator Utility (SOU), enter the following commands (where *name* is the name of the controller):

```
SOU/lcu
LCU/change_element_state element=<name> state=on
LCU/quit
```

DID VOLUMES REMAIN UNAVAILABLE?

After changing the state of the controller to ON, wait 2 minutes to ensure that all the jobs that have had requests blocked by the original failure have had a chance to automatically resubmit their requests.

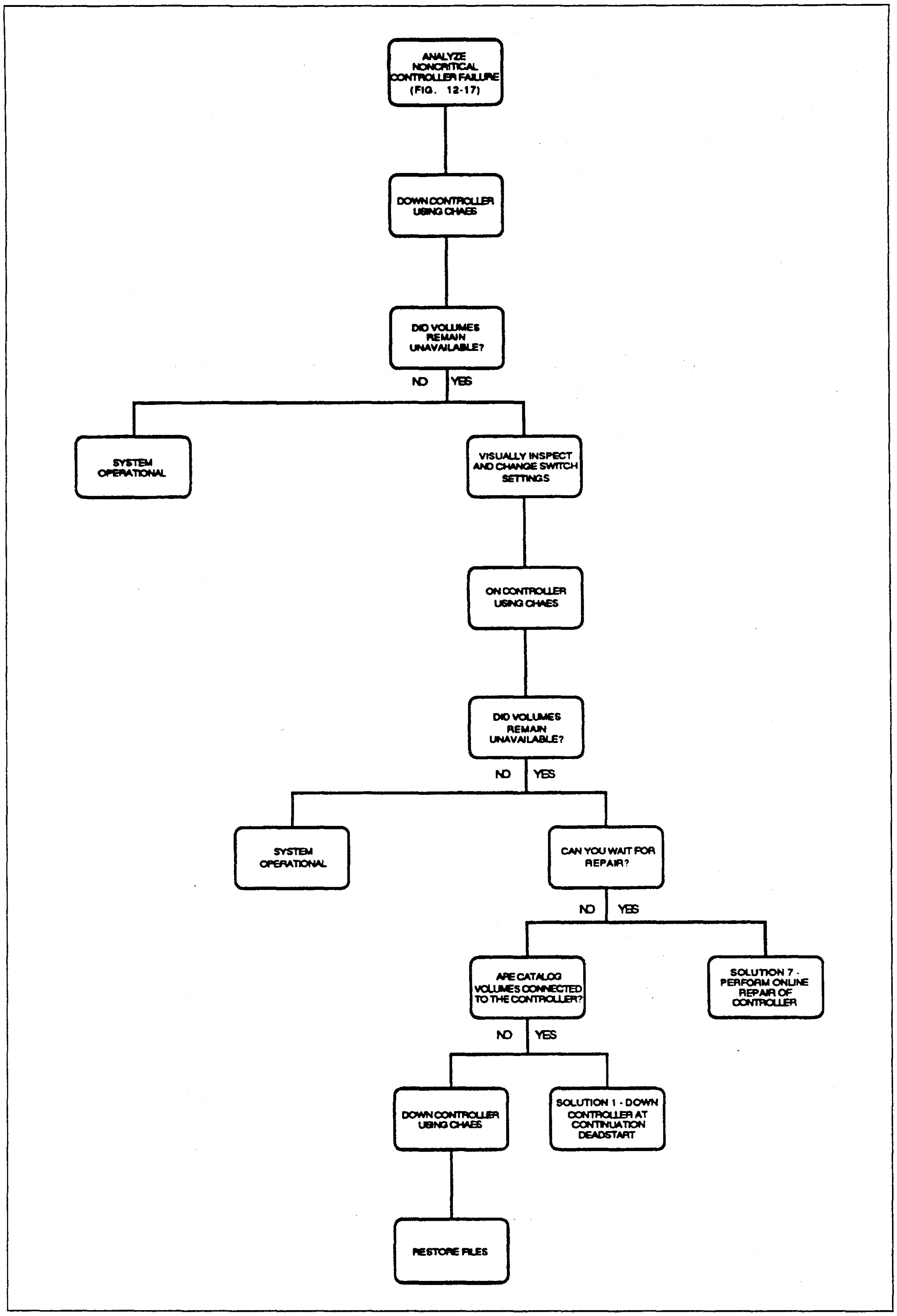

ANALYZE NONCRITICAL CONTROLLER FAILURE (FIG. 12-17)
DOWN CONTROLLER USING CHAES
DID VOLUMES REMAIN UNAVAILABLE?
NO
YES
SYSTEM OPERATIONAL
VISUALLY INSPECT AND CHANGE SWITCH SETTINGS
ON CONTROLLER USING CHAES
DID VOLUMES REMAIN UNAVAILABLE?
NO
YES
SYSTEM OPERATIONAL
CAN YOU WAIT FOR REPAIR?
NO
YES
ARE CATALOG VOLUMES CONNECTED TO THE CONTROLLER?
SOLUTION 7 - PERFORM ONLINE REPAIR OF CONTROLLER
NO
YES
DOWN CONTROLLER USING CHAES
SOLUTION 1 - DOWN CONTROLLER AT CONTINUATION DEADSTART
RESTORE FILES

Figure 12-17. Analyze Noncritical Controller Failure

ARE CATALOG VOLUMES CONNECTED TO THE CONTROLLER?

Make a note of the RECORDED_VSN of each volume connected to the failing controller and then determine whether any of the volumes at one time belonged to class J or L. A system interruption is required to restore catalogs. From the System Operator Utility (SOU), enter the following commands to obtain a display of the volume's current mass storage class membership:

```
SOU/lcu
LCU/display_ms_class
LCU/quit
```

RESTORE FILES

Refer to Restoring Files in chapter 13, Repair Solutions, for more information on restoring files. Follow the recommendations for restoring unavailable files.

This page intentionally left blank.

Analyze Noncritical Channel Failure

NOS/VE is not in a stepped state and the operator has determined that more than one disk volume is unavailable. A disk channel (or controller) may be the cause because all the unavailable volumes are accessed via the same channel.

DOWN THE CHANNEL USING CHANGE_ELEMENT_STATE (CHAES)

Down the channel. State changes are accepted because all the units that are connected to the channel are unavailable and the system is not in step; therefore, no critical volumes are connected only to this channel.

For example, if channel 7 were the channel, enter the following commands from the System Operator Utility (SOU):

```
SOU/lcu
LCU/change_element_state element=ch7 state=down
LCU/quit
```

DID VOLUMES REMAIN UNAVAILABLE?

Look at the VED MS display again.

IS THERE AN ALTERNATE CHANNEL?

Look at the VEDISPLAY DEVICE_STATUS (VED DS) display of the disk units corresponding to the volumes that are unavailable in the VED MS display and determine whether the disk units are connected to another channel. If they are, refer to Alternate and Redundant Path Configurations in appendix E, Supported Hardware Products, to determine whether the other channel displayed is considered to be an alternate or a redundant channel.

ON FAULTY CHANNEL USING CHANGE_ELEMENT_STATE (CHAES)

Turn the channel back on to see if the problem is reproduced. For example, if channel 7 were the channel, enter the following commands from the System Operator Utility (SOU):

```
SOU/lcu
LCU/change_element_state element=ch7 state=on
LCU/quit
```

Then wait approximately 2 minutes to ensure that all the jobs that were blocked by the original failure have had a chance to automatically resubmit their requests.

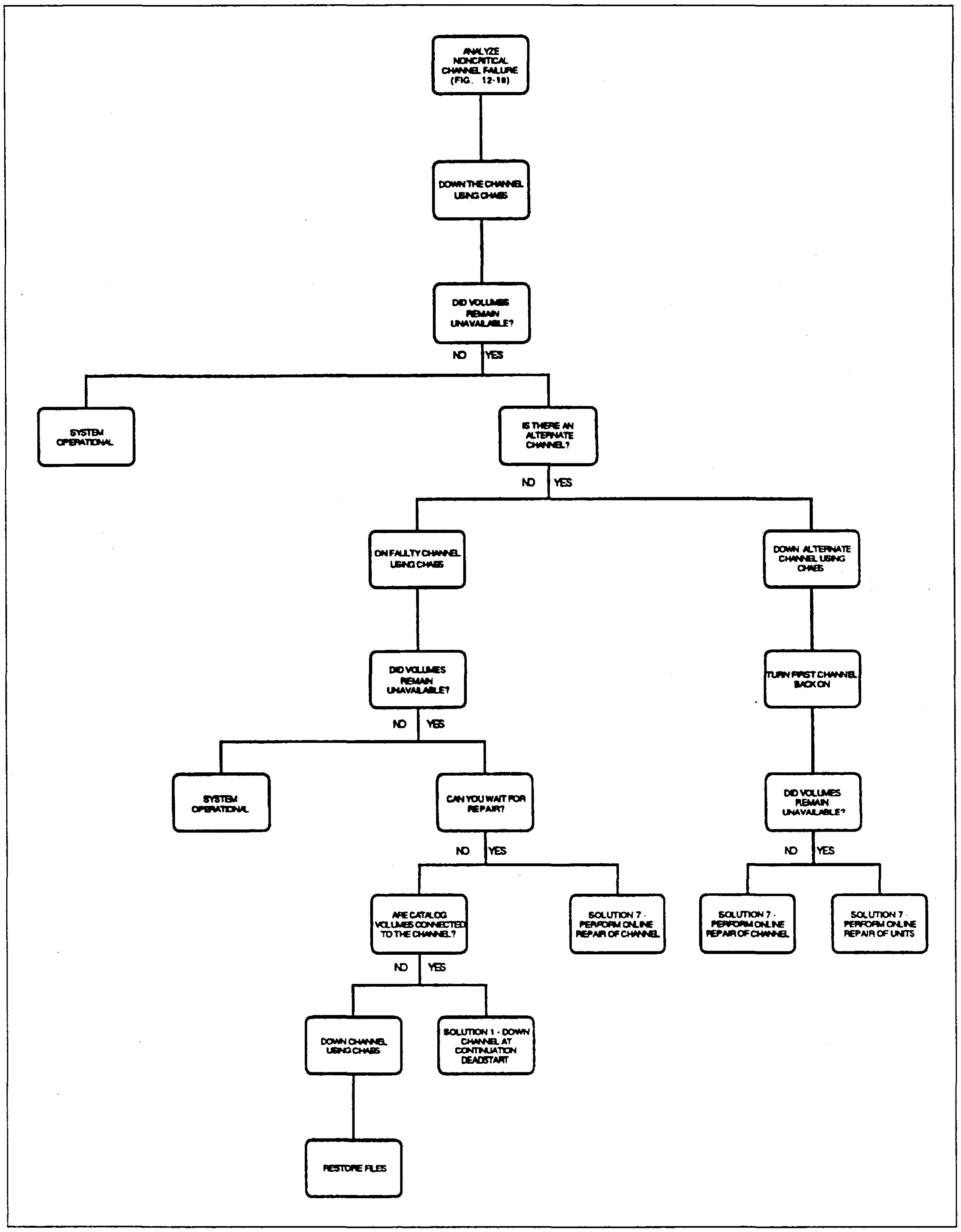

ANALYZE NONCRITICAL CHANNEL FAILURE (FIG. 12-18)
DOWN THE CHANNEL USING CHABS
DID VOLUMES REMAIN UNAVAILABLE?
NO
YES
SYSTEM OPERATIONAL
IS THERE AN ALTERNATE CHANNEL?
NO
YES
ON FAULTY CHANNEL USING CHABS
DOWN ALTERNATE CHANNEL USING CHABS
DID VOLUMES REMAIN UNAVAILABLE?
NO
YES
TURN FIRST CHANNEL BACK ON
SYSTEM OPERATIONAL
CAN YOU WAIT FOR REPAIR?
DID VOLUMES REMAIN UNAVAILABLE?
NO
YES
NO
YES
ARE CATALOG VOLUMES CONNECTED TO THE CHANNEL?
SOLUTION 7 - PERFORM ONLINE REPAIR OF CHANNEL
SOLUTION 7 - PERFORM ONLINE REPAIR OF CHANNEL
SOLUTION 7 - PERFORM ONLINE REPAIR OF UNITS
NO
YES
DOWN CHANNEL USING CHABS
SOLUTION 1 - DOWN CHANNEL AT CONTINUATION DEADSTART
RESTORE FILES

Figure 12-18. Analyze Noncritical Channel Failure

ARE CATALOG VOLUMES CONNECTED TO THE CHANNEL?

Make a note of the RECORDED_VSN of each volume connected to the failing channel and then determine whether any of the volumes at one time belonged to class J or L. A system interruption is required to restore catalogs. From the System Operator Utility (SOU), enter the following commands to obtain a display of the volume's current mass storage class membership:

```
SOU/lcu
LCU/display_ms_class
LCU/quit
```

DOWN ALTERNATE CHANNEL USING CHANGE_ELEMENT_STATE (CHAES)

In a dual-access configuration, the fault that caused volumes to become unavailable could have been in either channel. By trying the channels individually, you may isolate which one is at fault, if any.

RESTORE FILES

Refer to Restoring Files in chapter 13, Repair Solutions, for more information on restoring files.

This page intentionally left blank.

Analyze Magnetic Tape Failure

Ohe operator or an interactive user of magnetic tape has encountered a problem using magnetic tape equipment.

ENTER VED, DS

To use the VED DS display, you must be executing from within a System Operator Utility session with the SYSTEM_OPERATOR or SYSTEM_DISPLAY capability active.

ARE THERE ANY UNITS IN THE DOWN STATE?

Determine whether there are any magnetic tape units in the DOWN state.

DO YOU KNOW WHICH UNITS ARE GIVING YOU TROUBLE?

There are no units in the DOWN state. Therefore, NOS/VE has not detected a hardware problem that would cause NOS/VE to automatically down a unit. You need to determine whether the same unit is failing or whether a defective volume is causing the problem.

IS THERE MORE THAN ONE UNIT INVOLVED?

If more than one unit is involved, a faulty channel or controller could be the cause. Alternatively, a faulty volume repeatedly mounted on different units could make it appear that the hardware is at fault.

IS THERE MORE THAN ONE UNIT IN THE DOWN STATE?

Have you found, in looking at the VED DS display, that there are at least two magnetic tape units in the DOWN state? If so, this may indicate a faulty magnetic tape channel or controller.

END OF ANALYSIS

Either the analysis of the fault provided by this manual is inadequate or there was no fault in the system. If you suspect the NOS/VE system software is at fault, take a dump and contact Control Data.

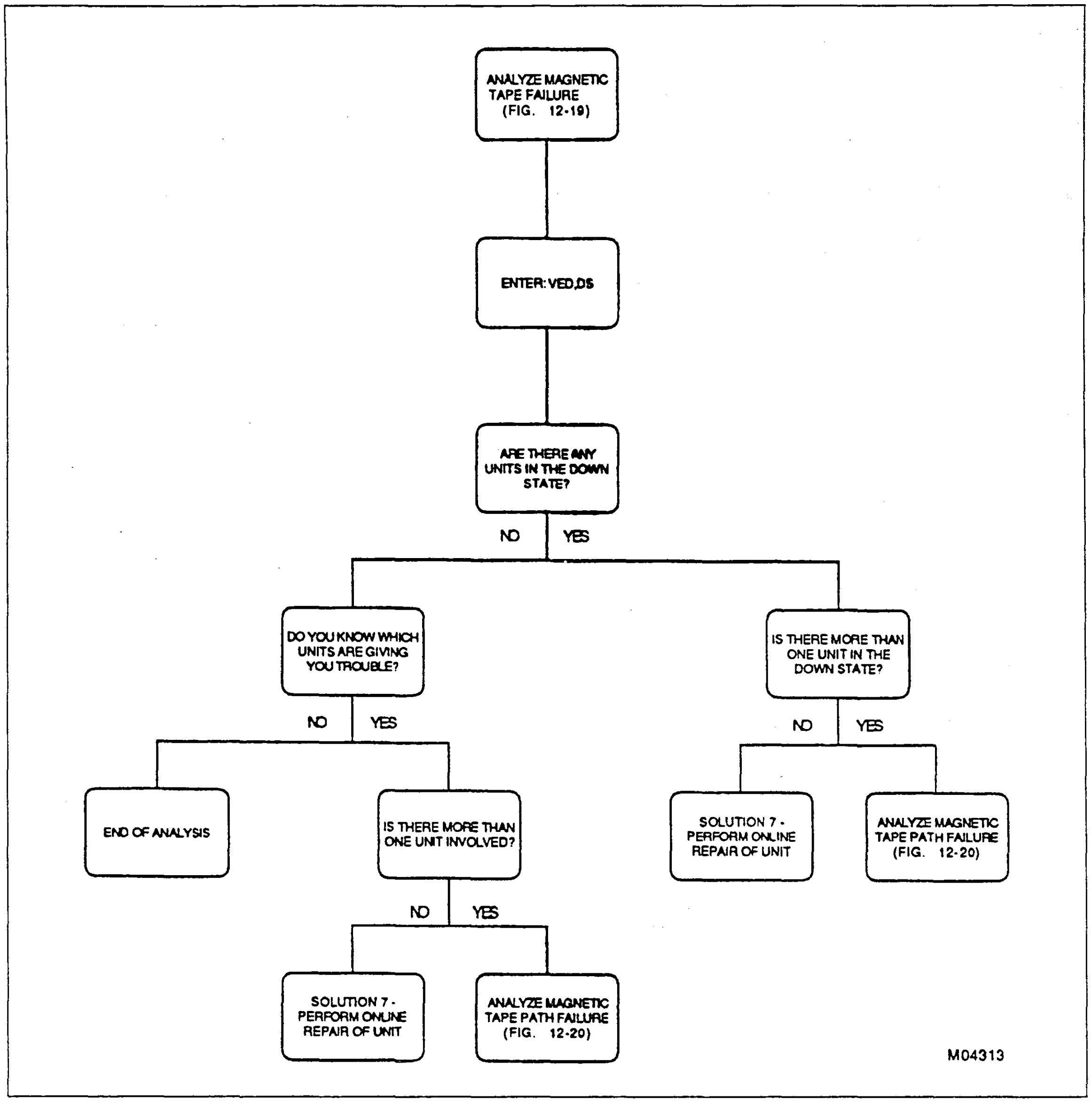

ANALYZE MAGNETIC TAPE FAILURE (FIG. 12-19)
ENTER: VED,DS
ARE THERE ANY UNITS IN THE DOWN STATE?
NO
YES
DO YOU KNOW WHICH UNITS ARE GIVING YOU TROUBLE?
IS THERE MORE THAN ONE UNIT IN THE DOWN STATE?
NO
YES
NO
YES
END OF ANALYSIS
IS THERE MORE THAN ONE UNIT INVOLVED?
SOLUTION 7 - PERFORM ONLINE REPAIR OF UNIT
ANALYZE MAGNETIC TAPE PATH FAILURE (FIG. 12-20)
NO
YES
SOLUTION 7 - PERFORM ONLINE REPAIR OF UNIT
ANALYZE MAGNETIC TAPE PATH FAILURE (FIG. 12-20)
M04313

Figure 12-19. Analyze Magnetic Tape Failure

Analyze Magnetic Tape Path Failure

NOS/VE is not in a stepped state and there are at least two magnetic tape units in the DOWN state.

ARE ALL UNITS ON THE SAME CHANNEL?

Does the VED DS display indicate that all the magnetic tape units with which you are having problems have a common channel connection?

ARE ALL UNITS ON THE CHANNEL FAULTY?

Does the common channel provide access to other units that are down or are causing a problem?

IS THERE ANOTHER CHANNEL TO THE UNITS?

Does the VED DS display indicate that all of the units in question have the same two channels in common? Refer to the discussion of Alternate and Redundant Path Configurations in appendix E of this manual for more information on channel configurations.

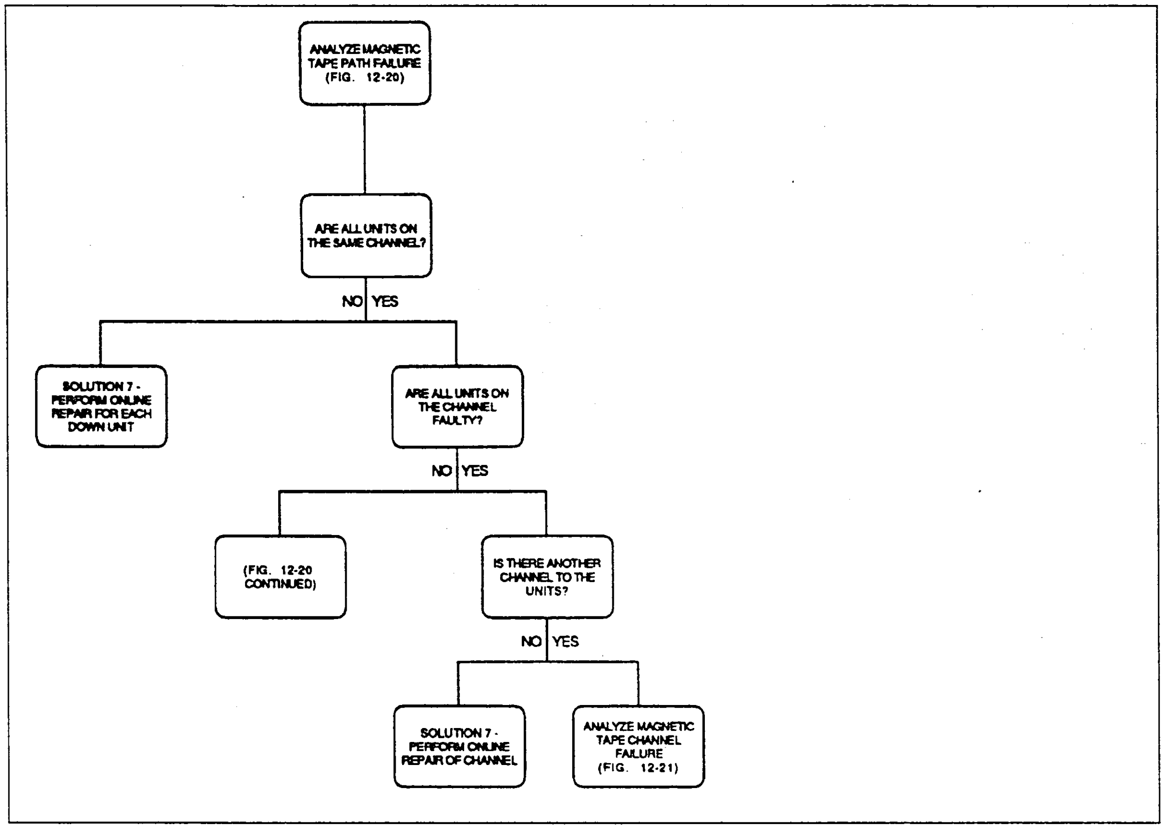

ANALYZE MAGNETIC TAPE PATH FAILURE (FIG. 12-20)
ARE ALL UNITS ON THE SAME CHANNEL?
NO
YES
SOLUTION 7 - PERFORM ONLINE REPAIR FOR EACH DOWN UNIT
ARE ALL UNITS ON THE CHANNEL FAULTY?
NO
YES
(FIG. 12-20 CONTINUED)
IS THERE ANOTHER CHANNEL TO THE UNITS?
NO
YES
SOLUTION 7 - PERFORM ONLINE REPAIR OF CHANNEL
ANALYZE MAGNETIC TAPE CHANNEL FAILURE (FIG. 12-21)

Figure 12-20. Analyze Magnetic Tape Path Failure

(Continued)

(Continued)

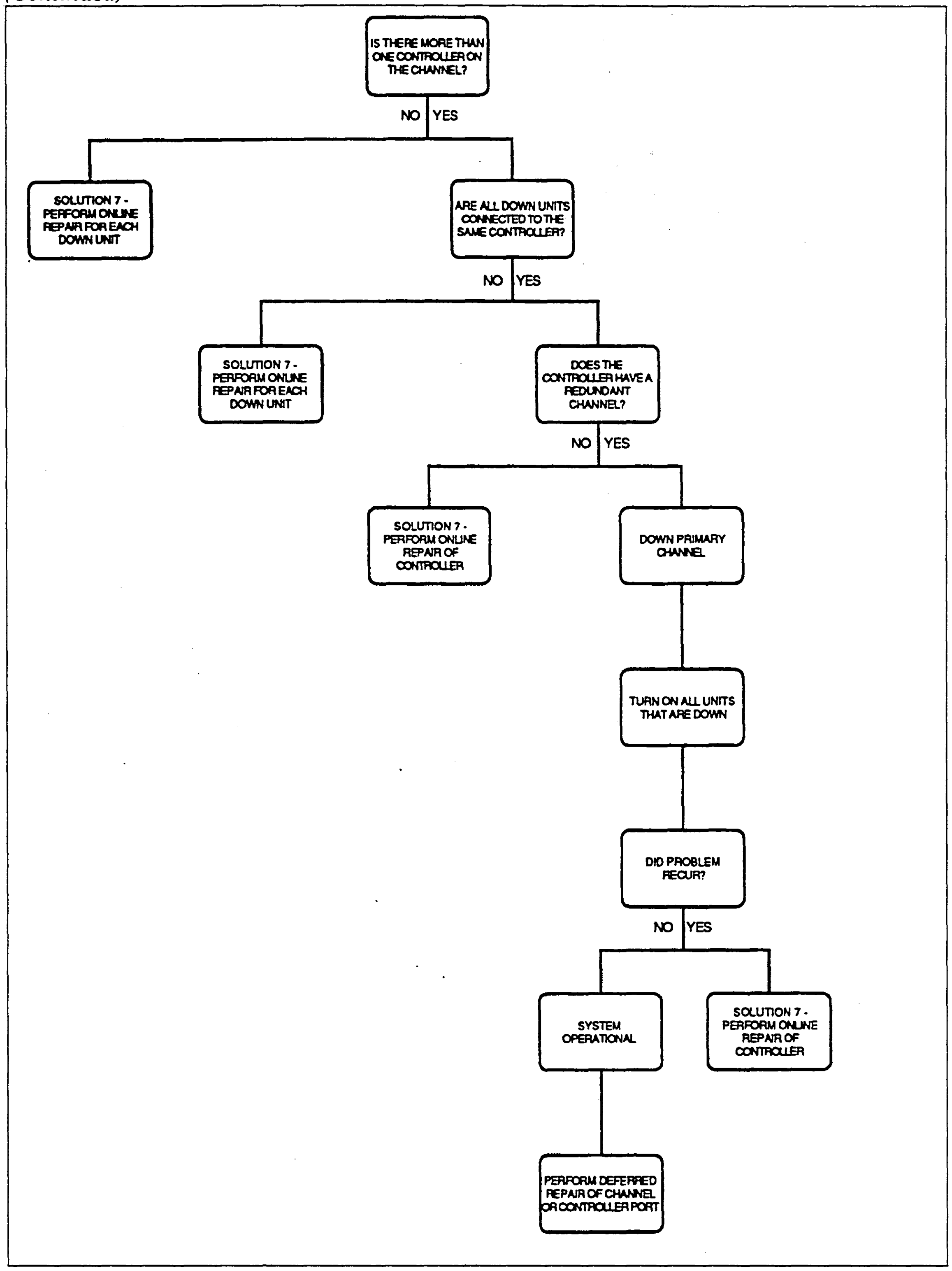

IS THERE MORE THAN ONE CONTROLLER ON THE CHANNEL?
NO
YES
SOLUTION 7 - PERFORM ONLINE REPAIR FOR EACH DOWN UNIT
ARE ALL DOWN UNITS CONNECTED TO THE SAME CONTROLLER?
NO
YES
SOLUTION 7 - PERFORM ONLINE REPAIR FOR EACH DOWN UNIT
DOES THE CONTROLLER HAVE A REDUNDANT CHANNEL?
NO
YES
SOLUTION 7 - PERFORM ONLINE REPAIR OF CONTROLLER
DOWN PRIMARY CHANNEL
TURN ON ALL UNITS THAT ARE DOWN
DID PROBLEM RECUR?
NO
YES
SYSTEM OPERATIONAL
SOLUTION 7 - PERFORM ONLINE REPAIR OF CONTROLLER
PERFORM DEFERRED REPAIR OF CHANNEL OR CONTROLLER PORT

Figure 12-20. Analyze Magnetic Tape Path Failure

IS THERE MORE THAN ONE CONTROLLER ON THE CHANNEL?

There may be more than one $5698_1x controller on a channel. If so, you must determine whether one of the controllers is at fault. Only the $5698_1x magnetic tape product allows multiple controllers per channel.

ARE ALL DOWN UNITS CONNECTED TO THE SAME CONTROLLER?

From the System Operator Utility (SOU), enter:

```
SOU/display_system_configuration output=$local.display
SOU/print_file $local.display
```

Find the units that are down in the printed display. The display for each unit identifies the name of each controller that is cabled to it.

DOES THE CONTROLLER HAVE A REDUNDANT CHANNEL?

Look at the printed display of the controller's connections to determine whether the $5698_1x controller has more than one channel connected to it.

DOWN PRIMARY CHANNEL

Enter the following commands from the System Operator Utility (SOU) to down the first channel listed in the display:

```
SOU/lcu
LCU/change_element_state element=<channel name> state=down
LCU/quit
```

TURN ON ALL UNITS THAT ARE DOWN

Enter the following commands from the System Operator Utility (SOU) to turn on all units connected to the controller that are in the DOWN state:

```
SOU/lcu
LCU/change_element_state element=<unit name> state=on
LCU/quit
```

DID PROBLEM RECUR?

If using a redundant channel to the controller does not solve the problem, the controller requires repair. Otherwise, the controller's channel port or the channel itself may require repair at some future time.

Analyze Magnetic Tape Channel Failure

NOS/VE is not in a stepped state. All of the magnetic tape units that are connected to a particular channel are either in the DOWN state or are failing. All of the units are accessible by two channels.

DOWN ONE CHANNEL USING CHANGE_ELEMENT_STATE (CHAES)

For example, assuming channel 7 is at fault, enter the following commands from the System Operator Utility (SOU):

```
SOU/lcu
LCU/change_element_state element=ch7 state=down
LCU/quit
```

TURN ON ALL UNITS THAT ARE DOWN

For example, assume that channel 7 has access to only two units, T45 and T46. Also assume that the units are down and are also accessed by channel 16. From the System Operator Utility (SOU), enter:

```
SOU/lcu
LCU/change_element_state element=t45 state=on
LCU/change_element_state element=t46 state=on
LCU/quit
```

DID PROBLEM RECUR?

After using the affected magnetic tape units for a period of time, did all of the units automatically change state to DOWN or did you notice the same kinds of symptoms as you saw before downing the one channel? If either case, the problem may be with the other channel or controller.

ON THE CHANNEL USING CHANGE_ELEMENT_STATE

DOWN THE ALTERNATE CHANNEL

From the System Operator Utility (SOU), enter:

```
SOU/lcu
LCU/change_element_state element=ch7 state=on
LCU/change_element_state element=ch16 state=down
LCU/quit
```

BOTH CHANNELS CONNECTED TO SAME CONTROLLER?

From the System Operator Utility (SOU), enter:

```
SOU/display_system_configuration o=$local.display
SOU/print_file $local.display
```

Find the channels in the display. The display for each channel identifies the name of each controller cabled to it.

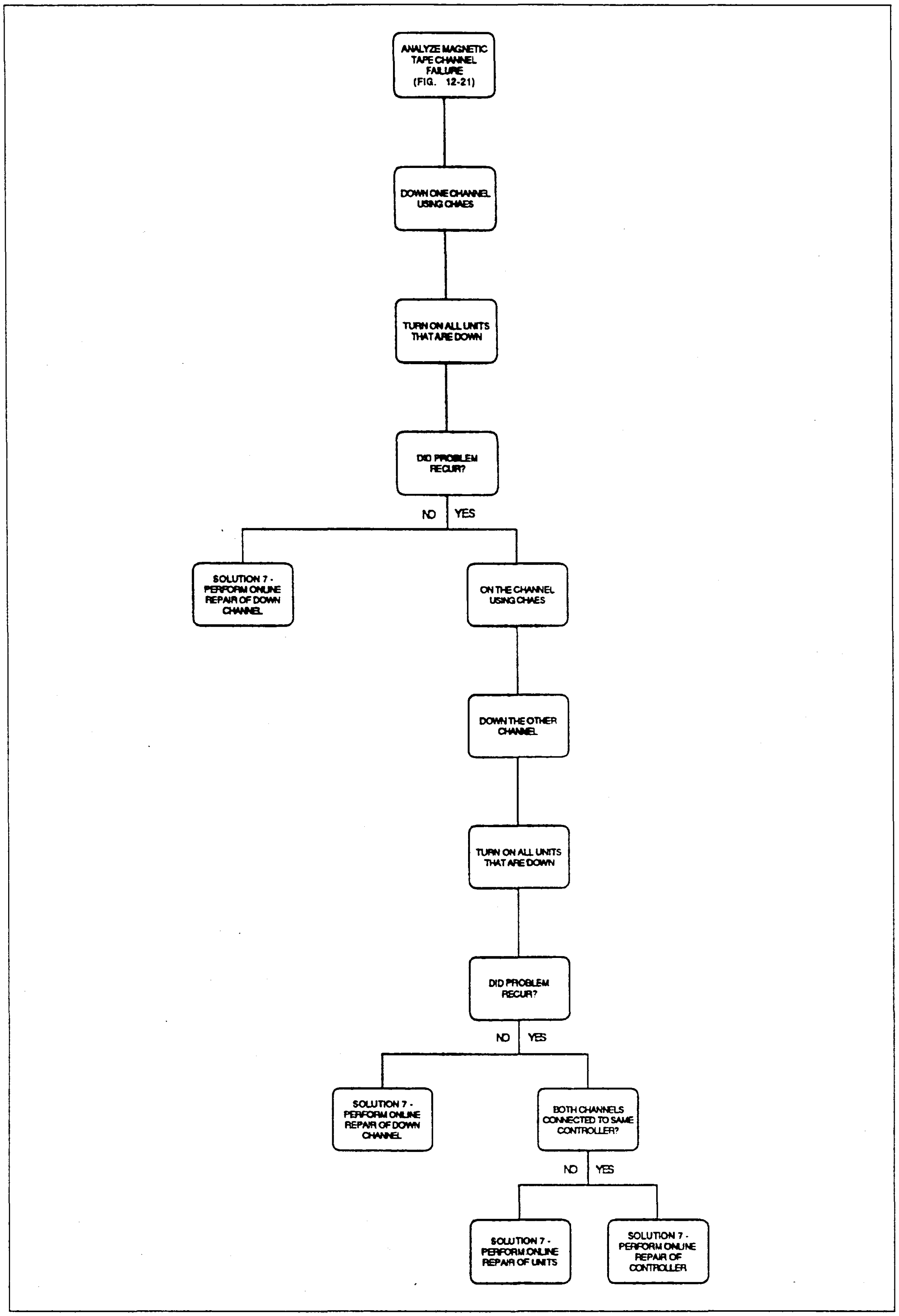

ANALYZE MAGNETIC TAPE CHANNEL FAILURE (FIG. 12-21)
DOWN ONE CHANNEL USING CHAES
TURN ON ALL UNITS THAT ARE DOWN
DID PROBLEM RECUR?
NO
YES
SOLUTION 7 - PERFORM ONLINE REPAIR OF DOWN CHANNEL
ON THE CHANNEL USING CHAES
DOWN THE OTHER CHANNEL
TURN ON ALL UNITS THAT ARE DOWN
DID PROBLEM RECUR?
NO
YES
SOLUTION 7 - PERFORM ONLINE REPAIR OF DOWN CHANNEL
BOTH CHANNELS CONNECTED TO SAME CONTROLLER?
NO
YES
SOLUTION 7 - PERFORM ONLINE REPAIR OF UNITS
SOLUTION 7 - PERFORM ONLINE REPAIR OF CONTROLLER

Figure 12-21. Analyze Magnetic Tape Channel Failure

Failure Data in Critical Display Window

NOS/VE uses the critical display window of the system console to explain the details concerning a failure or an unusual condition. To see the full content of the window, it is necessary to expand it; move the cursor to the critical display window of the system console and expand the window (depress the SHIFT and F9 (SUPER) keys at the same time).

If all the retries of the operation are unsuccessful, NOS/VE writes a final documentation of the failure to the critical display window. This final message consists of several lines. The first line of the group contains the word UNRECOVERED.

It is important that you write down all the information in all the lines contained in the last unrecovered message group in the critical display window, because if the system is in a stepped condition, it is probably due to this failure. You may also record other failures described in the critical display window (for the same channel, controller, and disk unit) that may have occurred just prior to the last one. This may help the CE understand the failure.

The disk failure data in the critical display window takes the following form:

```
hh:mm:ss <  path    >- <vsn > - UNRECOVERED         Cxxxx Txx Sxx
<symptom statement>
<action statement>
<failure data>
```

For example:

```
09:10:33 CH01  CO UOO- VSN101 - UNRECOVERED                   C0351 TOO S02
NO CONTROLLER INTERRUPT
DRIVE DOWNED
SR0400, RP 0008 400f 2005 0103 0018 2950 4d50 492f 434d 2d33
ER0000,    3930 3537 2d33 3231 2020 2020 2020 2020 0727 877a
```

NOTE

All numeric information displayed in the first line of the message group is in a decimal representation. Device status displayed in the last lines of the message group are in either an octal or a hexadecimal representation, depending upon the product.

In this example, the fields have the following definitions:

Table 12-1. Disk Failure Data

Field	Definition
hh.mm.ss	Time when the fault occurred.
< path >	Physical path used when the fault was detected. Several abbreviations describe the peripheral elements in this path: IOUn Name of the input output unit, where n is either 0 or 1. CHnn Nonconcurrent (NIO) channel. CCHn CIO channel of an I4. CCHnp CIO channel and port (A or B) of an I4. CCHnnp CIO channel and port (A or B) of an I4C. Name of the mainframe channel used. This is the same channel name that can be used as input to the CHANGE_ELEMENT_STATE LCU subcommand or to reconfigure the channel in the PCU. The numeric suffix (n or nn) is the decimal representation of the channel number. Cn Identifies the controller by physical address. The n suffix identifies the address of the controller on the channel, (equipment number). For an 887 disk unit (that connects directly to the channel and has a built-in controller per disk unit), this is the address of both the controller and the disk unit. Unn Identifies the disk unit by physical address. The nn suffix identifies the address from the controller to the disk unit (unit number). The nn value is the decimal representation of the unit number and is the same value used to describe the peripheral connection of the controller to the disk unit in the NOS/VE physical configuration.

(Continued)

Table 12-1. Disk Failure Data *(Continued)*

Field	Definition
	The path information is represented differently for certain groups of disk products:

Path	Product
	7155 (844 and 885) and 7165 (895)
CHnn Cn Unn	NIO 170 channel
CCHn Cn Unn	I4 CIO 170 channel
CCHnn Cn Unn	I4C CIO 170 channel
	FA7B4_D (834 and 836)
CHnn Cn Unn	NIO ISI channel
	FA7B5_A (9836 and 9853))
CHn Cn Unn	IO IPI channel
CCHnp Cn Unn	I4 CIO IPI channel
CCHnnp Cn Unn	I4C CIO IPI channel
	887
CCHnp Cn	I4 CIO ISI channel
CCHnnp Cn	I4C CIO ISI channel

Field	Definition
< vsn >	Identifies the RECORDED_VSN of the volume that was being accessed at the time of the failure. The RECORDED_VSN is the identification given to the volume by the LCU subcommand INITIALIZE_MS_VOLUME. This information is required if you use the LCU subcommand DEFINE_MS_FLAW to flaw a media defect on this volume.
UNRECOVERED	Fault was not correctable by the NOS/VE driver. This does not necessarily imply that the fault was not tolerated by the NOS/VE system.
Cxxxx Txx Sxx	Identifies the address of the media that was being accessed at the time of the failure. This is the address of a specific (physical) sector expressed in terms of cylinder, track, and sector. The cylinder, track, and sector numbers are displayed in a decimal representation to facilitate the use of the LCU subcommand DEFINE_MS_FLAW.
<symptom statement>	Description of the fault symptom.

<action statement>	Optional line in the message group that indicates an action taken by NOS/VE for a fault that was not correctable.
	
	One of the following statements may appear in the third line of the message groups. This indicates that one or more disk units were disabled.
	
	`ADAPTER DOWNED`
	`CHANNEL DOWNED`
	`CONTROL MODULE DOWNED`
	`CONTROLLER DOWNED`
	`CYBER COUPLER DOWNED`
	`DRIVE DOWNED`
	`MEDIA FAILURE`

<failure data>	Hardware fault status in an octal or hexadecimal representation, depending upon the product.

Although the word DOWNED is used in these messages, NOS/VE version 1.5.1 does not, in fact, automatically change the state of the failing element to the DOWN state. However, the failing disk unit or the disk units affected by a failing adapter, channel, or controller are put in a disabled state. Disabled can be viewed as a substate of ON because a CE does not have the capability to access the entire surface of a disk unit in a disabled state. The volumes affected are considered unavailable to a user of the system.

To reenable the disk unit(s), it is possible to enter the UNSTEP_SYSTEM command if the system went into a stepped state as a result of the failure. If the system did not go into a stepped state, you must first down the faulty element and then turn it back on using the LCU subcommand CHANGE_ELEMENT_STATE.

Inspect Critical Controller Switch Settings

Refer to the second line of the UNRECOVERED message group in the critical display window. If any of the following symptom statements appear and you have the required access to the switches and they are in the wrong position, you will be able to repair the problem yourself. Refer to Failure Data in Critical Display Window earlier in this chapter for more information on the format of the message group.

FUNCTION TIMEOUT

Implies that a disk controller or adapter is not responding to NOS/VE requests. If the controller is a 7155-1x controller, ensure that the channel enable/disable switch for the channel access is in the enabled position. If the controller is a 7165_2x CCC, ensure that the CCC's knob is in the online position.

INTERVENTION REQUIRED

Implies that a 7165-2x storage director cannot be accessed. Ensure that the enable/disable switch at the storage director is in the enabled position.

CAN NOT SELECT THE CONTROLLER

Implies that an 834, 836, 887, 9836, or 9853 controller cannot be accessed. For an 887 disk unit, ensure that the correct channel port switch on the disk unit is enabled. For the other controllers mentioned, a key is required to access the switch that powers the controller; if you know where the switch is and you have access to it, you will be able to repair this problem yourself.

Inspect Critical Drive Switch Settings

Refer to the second line of the UNRECOVERED message group in the critical display window. If any of the following symptom statements appear and you have the required access to the switches and the switches are in the wrong position, you will be able to repair the problem yourself. Refer to Failure Data in the Critical Display Window later in this chapter for more information on the format of the message group.

Drive not ready symptom statements:

CHAN ENABLE SWITCH OFF OR UNIT NOT CABLED

Indicates that an 885 disk volume is not accessible. Ensure that the CHAN I/CHAN II ENABLE switch corresponding to the channel identified in the message is depressed on the 885 disk unit.

DRIVE NOT READY

Indicates that a 9836 or 9853 disk volume is not accessible. Ensure the start switch is depressed on the 9853 disk unit. A key is required to access the start switch in the 9836 cabinet; if you do not have access to the key, you cannot repair this problem yourself.

DRIVE NOT READY - MIC1

Indicates that an 887 disk volume is not accessible. Ensure that the disk unit's motor enable/disable switch is in the enabled position.

SD - EQUIPMENT CHECK

Indicates that an 895 disk volume is not accessible. Ensure that the enable/disable switch is enabled at the DDC (head-of-string controller).

START SWITCH NOT DEPRESSED

Indicates that an 834 or 836 disk volume is not accessible. Ensure the start switch is depressed on the 834 or 836 disk unit.

UNIT OFF LINE OR NOT CABLED

Indicates that an 844 disk volume is not accessible. If the disk unit has an ONLINE switch, ensure that it is depressed. If the disk unit does not have an ONLINE switch, ensure the knob inside the back of the cabinet is turned to the online position; a key is required to access this knob.

UNIT NOT READY

Indicates that an 844 or 885 disk volume is not accessible. Ensure that the start switch is depressed on the 844 or 885 disk unit.

UNIT RESERVED

Indicates that an 885 disk volume is not accessible. Ensure that the 7155-1x controller's enable/disable switch corresponding to the 885 disk unit is in the enabled position.

You may also correct a disk unit write-protected condition. This condition implies that a volume cannot be written upon because a switch is set to prevent writing to the disk unit.

Drive write-protected symptom statements:

DRIVE WRITE PROTECTED

Indicates that a 834 or 836 volume cannot be written upon. Ensure the write-protect switch is not depressed.

HARDWARE WRITE PROTECTED

Indicates that a 9836 or 9853 volume cannot be written upon. Ensure the write-protect switch is not depressed on the 9853 disk unit. A key is required to access the write-protect switch in the 9836 cabinet; if you do not have access to the key, you cannot repair this problem yourself.

UNIT READ ONLY SWITCH ON

Indicates that an 885 volume cannot be written upon. Ensure the read-only switch is not depressed on the 885 disk unit.

Inspect Noncritical Disk Unit Switch Settings

You have been referred to this section because you determined that a switch setting was noncritical.

If any of the following switches are not in the required position and you have access to the switches, you will be able to repair the problem yourself and you should do so now. Make sure that you only modify the switch settings of the disk unit that you have downed or those of a controller that correspond to the disk unit that you have downed.

For an 834 or 836 disk unit, ensure that the start switch is depressed and the write-protect switch is not depressed.

For an 844 disk unit, ensure that the knob is turned to the online position and the start switch is depressed.

For an 885 disk unit, ensure the following:

- The disk unit's start switch is depressed.
- The disk unit's CHAN I/CHAN II ENABLE switch for each configured channel is depressed.
- The disk unit's read-only switch is not depressed.
- The 7155-1x controller's enable/disable switch corresponding to the 885 disk unit is enabled.

For an 887 disk unit, ensure that the motor enable/disable switch is enabled and that the correct port is enabled on the disk unit.

For an 895 disk unit, ensure that the enable/disable switch is enabled at the DDC (head-of-string controller).

For a 9836 disk unit, ensure that the start switch is depressed and the write-protect switch is not depressed. A key is required to access the disk unit's switches in the 9836 cabinet; if you do not have access to the key, you cannot repair this problem yourself.

Regardless of whether you have changed switch settings, turn the disk unit back on.

Repair Solutions 13

Repair Solutions 13

This chapter contains the documentation of problem solutions recommended in chapter 12, Failure Analysis.

Solution 1: Down Element at Continuation Deadstart

Use this repair solution when the system is no longer operational as a result of one of the following conditions:

- A disk channel cable failure or the failure of the only disk controller that is connected to the channel. Refer to Analyze Critical Channel Failure and Analyze Noncritical Channel Failure in chapter 12, Failure Analysis, for more information on channel or controller failure.
- An 834, 9836, or 9853 disk controller failure; refer to Analyze Critical Controller Failure and Analyze Noncritical Controller Failure in chapter 12, Failure Analysis.
- A disk unit logic failure; refer to Analyze Critical Unit Failure and Analyze Noncritical Unit Failure, Analyze Critical Media Failure and Analyze Repetitive Media Failure in chapter 12, Failure Analysis.
- A head disk assembly (HDA) failure (head crash, faulty media). Use this repair solution to down the disk unit on which the faulty volume is mounted until a replacement HDA is available.

This solution offers the following benefits that would not be realized if you performed an installation deadstart (repair solution 4):

- You need to restore only the number of files or catalogs that puts you quickly back into production, rather than reinstalling NOS/VE and restoring all files and all catalogs.
- You can recover files that were modified since the last backup but prior to the peripheral failure. These files would be lost and cause rework if you use repair solution 4.

This solution may have the following disadvantages compared to performing an installation deadstart (repair solution 4):

- If a NOS/VE system interrupt occurs, the system recovers files and jobs that were in central memory prior to the failure, unless central memory has been lost or compromised. Modified file data that has not yet been written to disk can be recovered. This recovery is only possible if the disk volume(s) to which a file is assigned is accessible when the continuation deadstart is performed.

However, if a volume is omitted from the configuration on a continuation deadstart, modified data cannot be written to the missing volume and the data is lost. When the volume is reinstated, the files that were attached for write at the time of the failure may have inconsistent data. Refer to Finding Inconsistent Files later in this chapter for more information on file recovery.

- Your file base may become inconsistent if you restore files whose modification date does not match the catalog. For example, assume you have two files called SOURCE and OBJECT_LIBRARY. Ordinarily, SOURCE would accurately represent the contents of OBJECT_LIBRARY. However, if SOURCE is on a volume that is not currently accessible and you restore an older version of SOURCE using this repair solution, the two files no longer match.

All the reasons for using this repair solution have the following conditions in common:

- The system device is not affected by the actions taken in this repair solution.
- No physical recabling is required.
- There is a requirement to change the state or the logical connection of an element in the physical configuration. Refer to Changing the Configuration at Deadstart later in this chapter for a description of how to request the ability to change the physical configuration. To change the state of a controller or a disk unit, you must know the name of the element in the physical configuration. If you cannot remember the name of the element, you may use the EDIT_PHYSICAL_CONFIGURATION subutility subcommand DISPLAY_CONNECTED_ELEMENTS to find it.
- Whenever a channel or controller that provides sole access to a class J, K, L, M, or Q volume is downed, or whenever a disk unit that belongs to any of those classes is downed, there is likely a need to restore files or catalogs.

  You might also need to restore files if you used the REQUEST_MASS_STORAGE command to assign permanent files to a volume that does not belong to class K or M. Refer to the Restoring Files and Restoring Catalogs sections in this chapter for more information on file and catalog restoration.

Repair Steps

The following steps describe repairs:

1. Initiate a continuation deadstart requesting pause for operator input. Refer to Reconfiguration at Continuation Deadstart later in this chapter for more information about making configuration changes at deadstart.

2. Select menu option 1 in response to the following display:

```
You have requested a pause for operator intervention.


NOS/VE RECONFIGURATION MENU - CONTINUATION DEADSTART -

You have the following choices for reconfiguration:

1 - Intervene before installing the physical configuration.

2 - Intervene before activating existing mass storage set members.

3 - Intervene after activating existing mass storage set members.

Enter selection, GO, or ? for HELP.
? 1
```

Depending upon the disk product involved, there are several techniques that can be used to reconfigure a faulty element:

a. If the fault is in a disk unit, you can down the disk unit using the following command sequence as an example:

```
PCU/edit_physical_configuration
PCE/display_element_definition  violet_895_2
```

Refer to chapter 2, Physical Configuration Utility, for more information on the DISPLAY_ELEMENT_DEFINITION subcommand. Assume the DISPLAY_ELEMENT_DEFINITION (DISED) subcommand displays the following:

```
DEFINE_ELEMENT E = VIOLET_895_2 ..
  ELEMENT_IDENTIFICATION = $895_2 STATE = ON ..
  SERIAL_NUMBER = 1184 ..
  PERIPHERAL_CONNECTION = ( ..
  ( VIOLET_7165_1 2 )) ..
  VERIFY_ELEMENT_IDENTIFICATION = TRUE
```

You then enter:

```
PCE/change_element_definition violet_895_2 s=down
PCE/display_element_definition violet_895_2
```

Assume the DISPLAY_ELEMENT_DEFINITION (DISED) subcommand displays the following:

```
DEFINE_ELEMENT E = VIOLET_895_2 ..
  ELEMENT_IDENTIFICATION = $895_2 STATE = DOWN ..
  SERIAL_NUMBER = 1184..
  PERIPHERAL_CONNECTION = ( ..
  ( VIOLET_7165_2 2 )) ..
  VERIFY_ELEMENT_IDENTIFICATION = TRUE
```

You then enter:

```
PCE/quit
PCU/install_physical_configuration
PCU/quit
```

b. If the fault is in the channel (or channel adapter) that is used to access an 834, 836, 9836, or 9853 controller or an 887 disk unit, then use the PCU subcommands shown below to reconfigure to a redundant channel.

If the 887 disk unit connects directly to two ports of the *same channel*, use the activity described immediately after the next note. Otherwise, refer to step 2c.

All of these connections require the reversal of the order of the channels identified in the IOU connection of the element.

In the following examples, assume that channel 1 is the faulty channel and TAN_XM_1 is the name of one of the controller(s) connected to the channel. Using the following command sequence as an example, change the IOU_CONNECTION parameter of each element connected to the channel. If you do not remember the names of the elements connected to the channel, use the EDIT_PHYSICAL_CONFIGURATION subutility subcommand DISPLAY_CONNECTED_ELEMENTS; refer to step 2e for an example of its use.

NOTE

The mainframe name in the new IOU_CONNECTION parameter defaults to the name of the mainframe on which the editing is done. The name of the IOU defaults to IOU0 (IOU zero). Therefore, these two fields are normally not required.

However, if you have more than one mainframe, you must qualify the channel with the correct mainframe name. If you have multiple IOUs on the same mainframe, you must qualify the channel and mainframe with the correct IOU name (either IOU0 or IOU1).

At the PCU prompt, enter:

```
PCU/edit_physical_configuration
PCE/display_element_definition tan_xm_1
```

Assume the DISPLAY_ELEMENT_DEFINITION (DISED) subcommand displays the following:

```
DEFINE_ELEMENT E = TAN_XM_1 ..
  ELEMENT_IDENTIFICATION = $FA7B5_A STATE = ON ..
  SERIAL_NUMBER = 221 ..
  IOU_CONNECTION = ( ..
  ( CH1 2 $SYSTEM_9303_0102 IOU0 )  ..
  ( CH19 2 $SYSTEM_9303_0102 IOU0 )) ..
  VERIFY_ELEMENT_IDENTIFICATION = TRUE
```

You then enter:

```
PCE/change_element_definition tan_xm_1 ic=((ch19 2) (ch1 2))
PCE/display_element_definition tan_xm_1
```

Assume the DISPLAY_ELEMENT_DEFINITION (DISED) subcommand displays the following:

```
DEFINE_ELEMENT E = TAN_XM_1 ..
  ELEMENT_IDENTIFICATION = $FA7B5_A STATE = ON ..
  SERIAL_NUMBER = 221 ..
  IOU_CONNECTION = ( ..
  ( CH19 2 $SYSTEM_9303_0102 IOU0 )  ..
  ( CH1 2 $SYSTEM_9303_0102 IOU0 )) ..
  VERIFY_ELEMENT_IDENTIFICATION = TRUE
```

You then enter:

```
PCE/quit
PCU/install_physical_configuration
PCU/quit
```

c. If the faulty element is one of the following, remove the IOU_CONNECTION parameter that includes the failing element from the controller's definition:

- One of the two channels of a 7165-21 or 7165-22 product.
- One of the two CCCs of a 7165-21 or 7165-22 product.
- One of the two storage directors of a 7165-21 or 7165-22 product.
- One of the two *different* channels connected to an 887 disk unit. If the 887 disk unit is connected to ports A and B of the same channel (for example, CCH0A and CCH0B), use step 2b instead.

You need to remember this connection and reinstate it when it has been repaired. If there are multiple 887 disk units connected to the faulty channel, you need to make the change to each disk unit's definition.

In the following example, assume that the failing element is the CCC connected to channel 1. If channel 1 (ch1) were at fault, you then have to change the IOU connection of every controller connected to the channel.

NOTE

The mainframe name in the new IOU_CONNECTION parameter defaults to the name of the mainframe on which the editing is made. The name of the IOU defaults to IOU0 (IOU zero). Therefore, these two fields are normally not required.

However, if you have more than one mainframe, you must qualify the channel with the correct mainframe name. If you have multiple IOUs on the same mainframe, you must qualify the channel and mainframe with the correct IOU name (either IOU0 or IOU1).

At the PCU prompt, enter:

```
PCU/edit_physical_configuration
PCE/display_element_definition violet_7165_2
```

Assume the DISPLAY_ELEMENT_DEFINITION (DISED) subcommand displays the following:

```
DEFINE_ELEMENT E = VIOLET_7165_2 ..
  ELEMENT_IDENTIFICATION = $7165_21 STATE = ON ..
  SERIAL_NUMBER = 588 ..
  IOU_CONNECTION = ( ..
  ( CH1 0 $SYSTEM_0855_0109 IOU0 )  ..
  ( CCH2 0 $SYSTEM_0855_0109 IOU0 )) ..
  VERIFY_ELEMENT_IDENTIFICATION = TRUE
```

You then enter:

```
PCE/change_element_definition violet_7165_2 ic=((cch2 0))
PCE/display_element_definition violet_7165_2
```

Assume the DISPLAY_ELEMENT_DEFINITION (DISED) subcommand displays the following:

```
DEFINE_ELEMENT E = VIOLET_7165_2 ..
  ELEMENT_IDENTIFICATION = $7165_21 STATE = ON ..
  SERIAL_NUMBER = 588 ..
  IOU_CONNECTION = ( ..
  ( CCH2 0 $SYSTEM_0855_0109 IOU0 )) ..
  VERIFY_ELEMENT_IDENTIFICATION = TRUE
```

You then enter:

```
PCE/quit
PCU/install_physical_configuration
PCU/quit
```

d. If the fault is in a disk controller other than the 7165-2x or if both halves of the 7165-2x have failed, the controller may be downed using the following subcommand. After downing an $FA7B5_A or an $FA7B4_D controller, refer to step 2e for further information. In the following example command sequence, assume violet_7155_2 is the name of the controller whose state is to be changed to DOWN.

```
PCU/edit_physical_configuration
PCE/display_element_definition violet_7155_2
```

Assume the DISPLAY_ELEMENT_DEFINITION (DISED) subcommand displays the following:

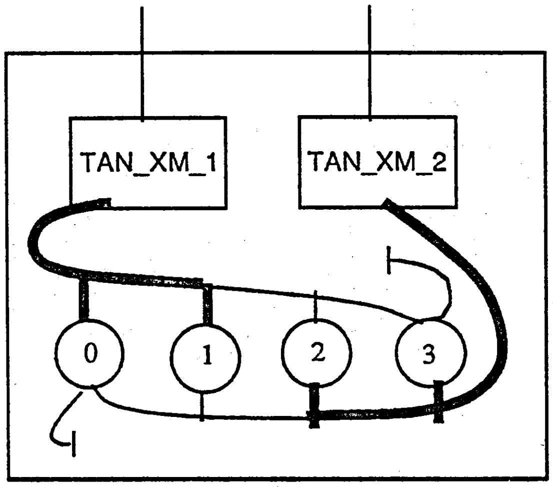
```
DEFINE_ELEMENT E = VIOLET_7155_2 ..
  ELEMENT_IDENTIFICATION = $7155_12 STATE = ON ..
  SERIAL_NUMBER = 3016 ..
  IOU_CONNECTION = ( ..
  ( CH22 0 $SYSTEM_0860_0302 IOU0 )) ..
  VERIFY_ELEMENT_IDENTIFICATION = TRUE
```

You then enter:

```
PCE/change_element_definition violet_7155_2 s=DOWN
PCE/quit
PCU/install_physical_configuration
PCU/quit
```

e. If the fault is in an $FA7B5_A or an $FA7B4_D disk controller connected to 9836 or 9853 disk units, you may want to reconfigure to a redundant controller.

   In the following example, assume that your controllers and disk units are cabled according to this figure. The boldface peripheral connections define the active connections and the other peripheral connections are for redundancy.

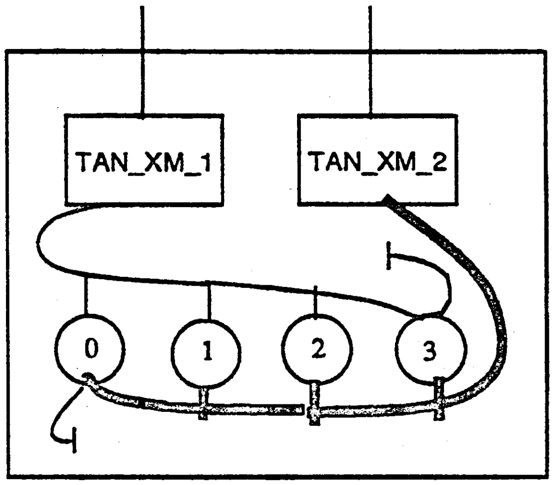


   In this example, assume that the controller TAN_XM_1 has been downed in step 2d and that a second controller, TAN_XM_2, is a redundant controller for disk units TAN_9853_0 and TAN_9853_1.

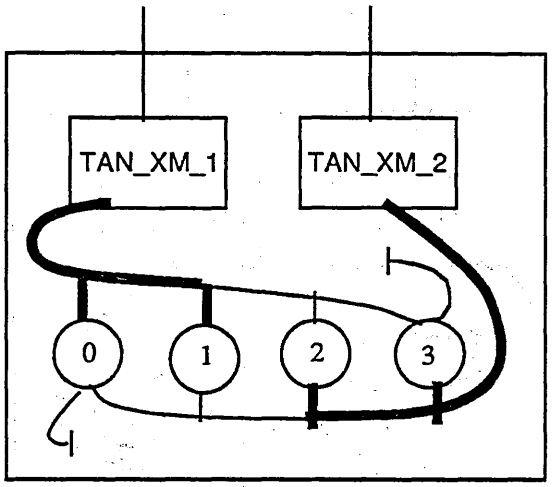
   At version 1.3.1 and previous systems, you were not allowed to describe more than one peripheral connection when defining a 9836 or 9853 unit (that is, only the connection to TAN_XM_1 could be defined for units TAN_9853_0 and TAN_9853_1).

   This restriction is removed in version 1.4.1. Downing the faulty controller in step 2d is sufficient, if you are aware of this change, if you already described all of your 9836 and 9853 units' connections, and if each unit has a redundant controller defined.

   As a result of downing TAN_XM_1, the active configuration is changed such that all four disk units are accessible only from TAN_XM_2 as shown in the following figure:



When TAN_XM_1 is repaired after the configuration deadstart, simply turn it back on and NOS/VE reinstates the active configuration as pictured below:



However, if any unit connected to the faulty controller is physically connected to a redundant controller, but that connection is not yet defined in the physical configuration, proceed with this step.

The DISPLAY_CONNECTED_ELEMENTS (DISCE) subcommand is used to determine the names of the elements connected to each controller. At the PCU prompt, enter:

```
PCU/edit_physical_configuration
PCE/display_connected_elements tan_xm_1 type=pa
```

Assume the DISPLAY_CONNECTED_ELEMENTS (DISCE) subcommand displays the following:

```
Element Name      Product        State       Serial Number      VEI

TAN_XM_1         $FA7B5_A        DOWN        221                TRUE
TAN_9853_0       $9853_1         ON          306                TRUE
TAN_9853_1       $9853_1         ON          307                TRUE
```

You then enter:

```
PCE/display_connected_elements tan_xm_2
```

Assume the DISPLAY_CONNECTED_ELEMENTS (DISCE) subcommand displays the following:

```
Element Name      Product        State       Serial Number      VEI

TAN_XM_2         $FA7B5_A        ON          220                TRUE
TAN_9853_2       $9853_1         ON          308                TRUE
TAN_9853_3       $9853_1         ON          309                TRUE
```

Your goal in the remainder of this step is to change the definition of all four disk units so that each unit is connected to both controllers. Furthermore, when TAN_XM_1 is repaired and turned back on, you want the disk units TAN_9853_0 and TAN_9853_1 to be actively used by TAN_XM_1 again.

You then enter:

```
PCE/change_element_definition TAN_9853_0 pc=((tan_xm_1 0) (tan_xm_2 0))
PCE/change_element_definition TAN_9853_1 pc=((tan_xm_1 1) (tan_xm_2 1))
PCE/change_element_definition TAN_9853_2 pc=((tan_xm_2 2) (tan_xm_1 2))
PCE/change_element_definition TAN_9853_3 pc=((tan_xm_2 3) (tan_xm_1 3))
PCE/disce tan_xm_1
```

Assume the DISPLAY_CONNECTED_ELEMENTS (DISCE) subcommand displays the following:

```
Element Name     Product      State     Serial Number    VEI

TAN_XM_1         $FA7B5_A     DOWN      220              TRUE
TAN_9853_0       $9853_1      ON        306              TRUE
TAN_9853_1       $9853_1      ON        307              TRUE
TAN_9853_2       $9853_1      ON        308              TRUE
TAN_9853_3       $9853_1      ON        309              TRUE
```

You then enter:

```
PCE/disce tan_xm_2
```

Assume the DISPLAY_CONNECTED_ELEMENTS (DISCE) subcommand displays the following:

```
Element Name     Product      State     Serial Number    VEI

TAN_XM_2         $FA7B5_A     ON        220              TRUE
TAN_9853_0       $9853_1      ON        306              TRUE
TAN_9853_1       $9853_1      ON        307              TRUE
TAN_9853_2       $9853_1      ON        308              TRUE
TAN_9853_3       $9853_1      ON        309              TRUE
```

You then enter:

```
PCE/quit
PCU/inspc
PCU/quit
```

3. Enter GO when the NOS/VE Reconfiguration Menu reappears.

4. Answer NO to the prompt about deleting the missing volume(s) from the set. This prompt appears if any volumes have been omitted from the configuration in step 2.

   NOTE

   If the NOS/VE system advises you that the $SYSTEM master catalog was recreated and the following menu appears, refer to repair solution 3b to conduct the remainder of the deadstart; otherwise, go to step 5.

   ```
   NOS/VE SYSTEM SET RECOVERY MENU - $SYSTEM FILE RESTORATION -

   Choose one of the following selections:
    1 - Display the number of unreconciled files and catalogs in the system set.
    2 - Restore files using the RESTORE_UNRECONCILED_FILES command.
    3 - Display the active volumes for all sets in the system.
    4 - Display command information for RESTORE_UNRECONCILED_FILES.
    5 - QUIT (The system will terminate.)
   Enter selection or ? for HELP.
   ```

5. The system may not be ready for production.

   If you downed a disk unit whose volume was a member of class J, K, L, or M, you may want to restore files or catalogs.

   If you downed a controller or removed an IOU_CONNECTION parameter, you may have made one or more disk units inaccessible. Again, you may want to restore files or catalogs at this time.

   Refer to the Restoring Files and Restoring Catalogs sections in this chapter for more information on file recovery. It is not required that you restore files at this time. However, if you are missing a volume belonging to classes J, K, or L, your users may not be able to use the system productively unless you do so. If the missing volume did not belong to classes J, K, or L, you may defer file restoration until it is known whether the data on the missing volume has been lost.

6. If you decide to restore catalogs or files, select the option to activate the system for system console usage. Otherwise, select the option to activate the system for production and go to step 8.

7. Perform the restoration.

8. Refer to the NOS/VE Operations manual for more information on activating a production environment. At the system console, enter:

   ```
   activate_production_environment
   ```

9. If you downed a disk unit, as defined in step 2a, proceed to repair solution 7 for more information about reinstating a disk unit to the configuration.

   If you downed a controller, as defined in step 2d or 2e, proceed to repair solution 7 for more information about reinstating a disk controller to the configuration.

If you edited the IOU_CONNECTION parameter of an element to reconfigure it, as described in steps 2b and 2c, and you want to reverse the configuration change, you must perform a continuation deadstart. However, you may schedule this deadstart for a convenient time.

a. To reconfigure, enter the following command from within the System Operator Utility (SOU):

```
SOU/terminate_system
```

b. Perform a NOS/VE continuation deadstart requesting deadstart pause for operator input.

c. Enter GO (rather than AUTO) to terminate system core command processing.

d. Reverse the changes to the physical configuration made previously in step 2.

e. If you want to initialize any of the volumes that are to be reinstated, go to step 2 of repair solution 2. Otherwise, complete the continuation deadstart.

Solution 2: Initialize Nonsystem Volume

Use this repair solution to initialize a volume other than the system device's volume. Refer to Analyze Repetitive Media Failure and Analyze Noncritical Unit Failure in chapter 12, Failure Analysis.

You may have been referred to this solution by a manual section other than chapter 12. Use this solution if a new version of CIP requires more capacity on a nonsystem volume than had been previously reserved, or if you want to initially install CIP to a nonsystem volume. In these cases, the volume may already be initialized and you may ignore step 5 in repair solutions 2a, 2b, and 2c.

In this repair solution, you will initialize a volume and add it to a set. Select a repair solution as follows:

2a. Initialize a member of the system set.

2b. Initialize the master volume of a nonsystem set.

2c. Initialize a member of a nonsystem set.

2d. Initialize a new volume and add it to a set.

Solutions 2a, 2b, and 2c are each slightly different from each other because of the following:

- An active volume cannot be initialized or added to a set.
- The LCU subcommands ADD_VOLUME_TO_SET and CREATE_SET activate a volume. CREATE_SET activates the master volume of a nonsystem set.
- The master volume of a set must be activated prior to adding a member to the set.
- The system device is automatically activated prior to the time the reconfiguration menu is initially displayed.
- Volumes other than the system device are automatically activated after the option INTERVENE BEFORE ACTIVATING EXISTING MASS STORAGE SET MEMBERS has been processed.

Repair solution 2d is used to add another volume to a set. Use this solution when upgrading the configuration with additional disk units.

Repair Steps for Solution 2a: Initialize System Set Member

1. Initiate a continuation deadstart requesting deadstart pause for operator input. For information on how to do this, refer to the NOS/VE Operations manual.

2. Select menu option 2 in response to the following display:

```
You have requested a pause for operator intervention.


NOS/VE RECONFIGURATION MENU - CONTINUATION DEADSTART -

You have the following choices for reconfiguration:

1 - Intervene before installing the physical configuration.

2 - Intervene before activating existing mass storage set members.

3 - Intervene after activating existing mass storage set members.

Enter selection, GO, or ? for HELP.

? 2
```

3. The member volume's disk unit must be in the ON state. If it is not turned on, use the CHANGE_ELEMENT_STATE subcommand to turn it on.

4. If a media defect was recurring, use the following subcommand to flaw the media defect. Use the location of the media defect that you recorded in Analyze Critical Media Failure.

```
define_ms_flaw recorded_vsn=<member_vsn> c=ccc t=tt s=ss
```

5. Use the following LCU subcommands to initialize the volume, to assign mass storage class membership to the volume, and to add the volume back to its set. You may give the volume the same recorded VSN as it had previously, or you may give the volume a new identity.

```
initialize_ms_volume element=<disk_unit_name> recorded_vsn=<member_vsn>
change_ms_class recorded_vsn=<member_vsn> delete_class=<list>
  add_class=<list>
add_volume_to_set recorded_vsn=<member_vsn>
quit
```

NOTE

If the volume had previously been initialized by NOS/VE, you will see the following message displayed at the system console when you initialize the volume again:

```
INITIALIZE_MS_VOLUME detected a recorded volume serial number: xxxxxx for
element DISK0
Files on this volume will be lost if you initialize it.
Enter YES to allow initialize to continue
enter NO to stop initialization on the current device.
```

Enter YES.

NOTE

If the volume had previously been a member of a set, you will see the following message displayed at the system console when you initialize the volume again.

```
Volume xxxxxx is currently a member of set <set name>
Enter YES to remove the volume from the set and allow initialize to continue,
enter NO to stop initialization on the current device.
```

Enter YES.

---

6. Enter GO when the NOS/VE reconfiguration menu reappears (refer to step 2).

   If the NOS/VE system advises you that the $SYSTEM master catalog was recreated and the following menu appears, refer to repair solution 3b to conduct the remainder of the deadstart; otherwise, go to step 7.

```
NOS/VE SYSTEM SET RECOVERY MENU - $SYSTEM FILE RESTORATION -

Choose one of the following selections:
 1-Display the number of unreconciled files and catalogs in the system set.
 2-Restore files using the RESTORE_UNRECONCILED_FILES command.
 3-Display the active volumes for all sets in the system.
 4-Display command information for RESTORE_UNRECONCILED_FILES.
 5-QUIT (The system will terminate.)
Enter selection or ? for HELP.
```

7. The system may not be ready for production

   If you initialized a volume that was a member of class J, K, L, or M, you may want to restore files or catalogs. Refer to Restoring Files and Restoring Catalogs later in this chapter for more information. It is not required that you restore files at this time. However, if you initialized a volume belonging to classes J, K, or L, your users may not be able to use the system productively unless you do so.

8. If you decide to restore catalogs or files, select the option to activate the system for system console usage; otherwise, proceed to step 10.

9. Perform the restoration.

10. Enter the following command at the system console:

```
activate_production_environment
```

Repair Steps for Solution 2b: Initialize Nonsystem Set Master Volume

When the master volume of a set is initialized, knowledge of the previous set membership is lost. The previous set membership must be reestablished using the LCU subcommand ADD_VOLUME_TO_SET for each member volume. Initialization of the master volume of a set does not affect the files and catalogs of other members of the set. However, to recover the set's files and catalogs, it is necessary to rebuild the root catalog that resides on the set's master volume. The process of rebuilding the root catalog begins with the specification of RECOVER_SET=TRUE when recreating the set using the LCU subcommand CREATE_SET. Once the set membership is reestablished, you must restore missing catalogs to reinstate the family catalogs contained in the root catalog.

1. Initiate a continuation deadstart requesting deadstart pause for operator input. For information on how to do this, refer to the NOS/VE Operations manual.

2. Select menu option 2 in response to the following display:

```
You have requested a pause for operator intervention.


NOS/VE RECONFIGURATION MENU - CONTINUATION DEADSTART -

You have the following choices for reconfiguration:

1 - Intervene before installing the physical configuration.

2 - Intervene before activating existing mass storage set members.

3 - Intervene after activating existing mass storage set members.

Enter selection, GO, or ? for HELP.

? 2
```

3. The master volume's disk unit must be in the ON state. If it is not on, use the CHANGE_ELEMENT_STATE subcommand to turn it ON.

4. If a media defect was recurring, use the following subcommand to flaw the media defect. Use the location of the media defect that you recorded in Analyze Critical Media Failure.

```
define_ms_flaw recorded_vsn=<master_vsn> c=cccc t=tt s=ss
```

5. Use the following LCU subcommands to initialize the master volume, to assign mass storage class membership to the volume, and to recreate the set. You may give the volume the same recorded VSN as it had previously or you may give the volume a new identity.

WARNING

It is required that you retain the original set name. If you change the name of the set, you must use the CREATE_FAMILY command to assign the set's original families to the set. Normally, the families would be automatically created on the proper set during a subsequent catalog restoration; however, changing the name of the set causes the set name in the backup file to become undefined.

```
LCU/initialize_ms_volume element=<disk_unit_name> recorded_vsn=<master_vsn>
LCU/change_ms_class recorded_vsn=<master_vsn> delete_class=<list> ..
LCU../  add_class=<list>
LCU/create_set set_name=<name> master_vsn=<master_vsn> recover_set=true
```

NOTE

If the volume had previously been initialized by NOS/VE, you will see the following message displayed at the system console when you initialize the volume again:

```
INITIALIZE_MS_VOLUME detected a recorded volume serial number: xxxxxx for
element DISKO
Files on this volume will be lost if you initialize it.
Enter YES to allow initialize to continue
enter NO to stop initialization on the current device.
```

Enter YES.

If the volume had previously been a member of a set, you will see the following message displayed at the system console when you initialize the volume again:

```
Volume xxxxxx is currently a member of set <set name>
Enter YES to remove the volume from the set and allow initialize to continue,
enter NO to stop initialization on the current device.
```

Enter YES.

For each original member of the set, use the LCU subcommand ADD_VOLUME_TO_SET to add the member back to the set. In the example below, assume there are only three members that must be reinstated to the set:

```
LCU/var
VAR/set: list of name 6..6 =(vsn001, vsn002, vsn003)
VAR/varend
LCU/for each member in set do
FOR/add_volume_to_set set_name=<name> recorded_vsn=member
FOR/forend
LCU/quit
```

6. Enter GO when the NOS/VE reconfiguraton menu reappears (see step 2).

7. The system is not ready for production.

   If you changed the set's name in step 5, you must now enter a CREATE_FAMILY command to assign the original families back to the set. In the following example, assume there are two families to be reinstated. You must also define the family administrator and the account, project, and password of the family administrator for each family at this time. For brevity, these parameters of CREATE_FAMILY are not shown in the following example:

```
SOU/create_family family_name=family1 permanent_file_set=<name>
SOU/create_family family_name=family2 permanent_file_set=<name>
```

8. You may defer recovering the set until a more convenient time. However, access to catalogs and files on the set is denied until the set is recovered. To proceed with the set recovery, select the option to activate the system for system console usage; otherwise, go to step 10.

9. Proceed to restore missing catalogs from your most recent catalog backup of the set. Restore files if the master volume belonged to classes K or M. Refer to Restoring Catalogs later in this chapter for more information.

10. Enter the following command at the system console:

```
activate_production_environment
```

Repair Steps for Solution 2c: Initialize Nonsystem Set Member

1. Initiate a continuation deadstart requesting deadstart pause for operator input. For information on how to do this, refer to the NOS/VE Operations manual.

2. In response to the following display, select menu option 2:

```
You have requested a pause for operator intervention.


NOS/VE RECONFIGURATION MENU - CONTINUATION DEADSTART -

You have the following choices for reconfiguration:

1 - Intervene before installing the physical configuration.

2 - Intervene before activating existing mass storage set members.

3 - Intervene after activating existing mass storage set members.

Enter selection, GO, or ? for HELP.

? 2
```

3. The member volume's disk unit must be in the ON state. If it is not on, use the CHANGE_ELEMENT_STATE subcommand to turn it on.

4. If a media defect was recurring, use the following subcommand to flaw the media defect. Use the location of the media defect that you recorded in Analyze Critical Media Failure.

```
LCU/define_ms_flaw recorded_vsn=<member_vsn> c=cccc s=ss
```

5. Use the following LCU subcommand to initialize the member volume:

```
LCU/initialize_ms_volume element=<disk_unit_name> recorded_vsn=<member_vsn>
LCU/quit
```

NOTE

If the volume had previously been initialized by NOS/VE, you will see the following message displayed at the system console when you initialize the volume again:

```
INITIALIZE_MS_VOLUME detected a recorded volume serial number: xxxxxx for
element DISK0
Files on this volume will be lost if you initialize it.
Enter YES to allow initialize to continue
enter NO to stop initialization on the current device.
```

Enter YES.

NOTE

If the volume had previously been a member of a set, you will see the following message displayed at the system console when you initialize the volume again.

```
Volume xxxxxx is currently a member of set <set name>
Enter YES to remove the volume from the set and allow initialize to continue,
enter NO to stop initialization on the current device.
```

Enter YES.

---

6. When the reconfiguration menu reappears, select the option: INTERVENE AFTER ACTIVATING EXISTING MASS STORAGE SET MEMBERS. Use the following LCU subcommands to assign mass storage class membership to the volume, and to add it back to the set. Assign the volume to appropriate mass storage classes to reflect your configuration; refer to the section called Configuration Recommendations in chapter 11. You may give the volume the same recorded VSN as it had previously or you may give the volume a new identity.

```
change_ms_class recorded_vsn=<member_vsn> delete_class=<list> ..
  add_class=<list>
add_volume_to_set set_name=<name> recorded_vsn=<member_vsn>
```

7. The system may not be ready for production.

   If you initialized a volume that was a member of class J, K, L, or M, you may want to restore files or catalogs. Refer to Restoring Files and Restoring Catalogs later in this chapter for more information on file and catalog restorations. It is not required that you restore files at this time. However, if you initialized a volume belonging to classes J, K, or L, your users may not be able to use the system productively unless you do.

8. If you decide to restore catalogs or files, select the option to activate the system for system console usage; otherwise, proceed to step 10.

9. Perform the restoration.

10. Enter the following command at the system console:

```
activate_production_environment
```

Repair Steps for Solution 2d: Add New Set Member

This solution assumes that the volume about to be added to a set has not previously been initialized by NOS/VE. If this is not the case, use repair solution 2a, 2b, or 2c, as appropriate.

This solution assumes that the volume's disk unit is not currently defined in the NOS/VE active configuration. If you planned ahead and configured the new disk unit in the OFF state in your active configuration, you may enter the LCU and execute the subcommands shown in step 4 without performing a continuation deadstart.

1. Initiate a continuation deadstart requesting deadstart pause for operator input. For information on how to do this, refer to the NOS/VE Operations manual.

2. Select menu option 1 in response to the following display:

```
You have requested a pause for operator intervention.


NOS/VE RECONFIGURATION MENU - CONTINUATION DEADSTART -

You have the following choices for reconfiguration:

1 - Intervene before installing the physical configuration.

2 - Intervene before activating existing mass storage set members.

3 - Intervene after activating existing mass storage set members.

Enter selection, GO, or ? for HELP.

? 1
```

3. Edit the physical configuration to define the disk unit and its connections. Refer to Repair Solution 1 for examples of editing the physical configuration. Use the following command sequence:

```
PCU/edipc
PCU/add_element_definition e=<new unit> ei=$9853_1 state=off
PCU/quit
PCU/inspc
PCU/quit
```

4. When the reconfiguration menu reappears, select the last option (to intervene after activating existing mass storage set members).

   Use the following LCU subcommands to initialize the volume, to assign mass storage class membership to the volume, and to add the volume to its set. Assign the volume to appropriate mass storage classes to reflect your configuration. For information on how to configure mass storage classes, see the Configuration Recommendations section of chapter 11.

```
LCU/change_element_state element=<new unit> state=on
LCU/initialize_ms_volume element=<new unit> recorded_vsn=<new vsn>
LCU/change_ms_class recorded_vsn=<new vsn> delete_class=<list> ..
LCU../add_class=<list>
LCU/add_volume_to_set set_name=<name> recorded_vsn=<new vsn>
LCU/quit
```

5. Enter the following command at the system console:

```
activate_production_environment
```

Solution 3: Recover the System Set

Use this repair solution to restore unreconciled $SYSTEM catalogs and recover the remaining volumes in the system set. If you came to this repair solution for any of the following reasons, use repair solution 3a to initialize the system device and recover the system set:

- A software fault that prevents a continuation deadstart. Refer to Analyze Critical Software Failure in chapter 12, Failure Analysis. This could be caused by the destruction of the NOS/VE system code or data located on the system device.
- A hardware fault that prevents further use of the system device. Refer to Analyze Critical Unit Failure, Analyze Use of Another Controller, and Analyze Critical Channel Failure in chapter 12, Failure Analysis. In these situations, you need to make physical configuration changes during your installation deadstart.
- An uncorrectable media defect on the system device that could not be flawed during a continuation deadstart. Refer to Analyze Critical Media Failure and Analyze Repetitive Media Failure in chapter 12, Failure Analysis.
- If a new version of CIP requires more capacity on the system device than had been previously reserved or if you want to initially install CIP to your system device.
- If an analyst has indicated that the system device must be initialized for reasons other than those listed above; for example, to change the name of the system set.

You may have begun a deadstart for one of the following reasons. However, NOS/VE detected that the $SYSTEM master catalog was missing and began the process of recovering the system set. Proceed to repair solution 3b to recover the system set without initialization of the system device.

- The failure of a disk unit whose volume belongs to class J. Refer to repair solution 1 in this chapter for references to the analysis that led to this solution.
- The need to initialize a disk volume that belongs to class J. Refer to repair solution 2 in this chapter for references to the analysis that led to this solution.
- The need to recable causes a volume belonging to class J to no longer be accessible. Refer to repair solution 5 for references to the analysis that led to this solution.

Repair Solution 3a: Initialize System Device

NOTE

When you initialize the system device, the following items are lost:

- Queued input files.
- Executing jobs.
- Global system logs.
- Files created on the old system device since the last backup.

The following repair steps initialize a system device:

1. If you have determined in the preceding analysis that the system device is no longer physically accessible, you will need to pick an alternate unit to be the new system device. If you have a spare unit that you set aside for this purpose, your decision is an easy one; otherwise, pick a unit whose volume does not belong to classes J, K, L, and M, if possible. Because the process of reinitializing the system device will not recover active jobs, the ideal choice for the new system device is one that has only job private files assigned to it. Job private files are swap files (class C), temporary files (class N), and critical files (class Q).

2. Install CIP on the new system device if your system device was where CIP was previously installed.

3. The NOS/VE installation deadstart processes for standalone and dual-state systems are different:

   - Standalone

     Mount the NOS/VE deadstart tape and initiate CIP execution by performing the steps described in the NOS/VE Operations manual. At the CIP Initial Options display, enter the following to initiate deadstart:

O	To switch to the Operator Intervention Display.
S	To switch to the Select Deadstart Device Display.
T	To select Tape Deadstart.

     For a CYBER 930 series machine, refer to the NOS/VE Installation Handbook for instructions on how to do an installation deadstart.

   - NOS dual-state

     Mount the NOS/VE deadstart tape on a tape unit reserved for NOS/VE. Initiate deadstart using the NVEffff procedure file.

     Enter the following DSD command at the NOS system console:

     ```
     NVEffff.
     ```

     where ffff is the name of the procedure associated with a NOS/VE tape deadstart. Your site analyst creates and names this deadstart procedure using the SETVE procedure and uses the T=TAPE parameter to specify tape deadstart. The SETVE procedure is documented in the NOS/VE Installation Handbook.

- NOS/BE dual-state

  Mount the NOS/VE deadstart tape on a tape unit reserved for NOS/VE. Initiate deadstart using the NVEffff procedure file. At the NOS/BE system console, enter the following commands to execute the NOS/VE deadstart procedure at an empty control point:

  ```
  n.CLEAR
  N.X NVE(ffff,id)
  ```

  where n is a control point number, ffff is the name of the procedure associated with a NOS/VE deadstart from tape, and id is an optional permanent file identifier (ID) of the deadstart procedure file. Your site analyst creates and names the deadstart procedure using the SETVE procedure and uses the T=TAPE parameter to specify tape deadstart. The SETVE procedure is documented in the NOS/VE Installation Handbook.

4. NOS/VE deadstart pauses at the Deadstart and System Device Configuration Selections menu. If option 2 (Deadstart pause for operator input) is not TRUE, select option 2 and set it to TRUE.

5. Check that the values for IOU, channel, controller type, equipment number and unit number describe the path to the system device and the deadstart tape device. If they are not correct, enter the number of the value that is incorrect and supply the correct value.

6. When you are satisfied with the values on the display, press the NEXT key to continue the deadstart process.

7. Specify the RECOVER_SYSTEM_SET parameter on the INITIALIZE_SYSTEM_DEVICE system core command. This parameter causes the new system device to be initialized without affecting the other volumes in the system set. In response to the following message:

   ```
   Enter system core commands
   ```

   enter the following command to initialize the system device and recover the remaining volumes of the system set:

   ```
   initialize_system_device <rvsn>,<boolean>,recover_system_set,<set_name>
   ```

   where:

rvsn	Specifies the recorded volume serial number (RECORDED_VSN) to assign to the volume on the system device.
boolean	Specifies whether or not the software flaws recorded on this volume are to be retained. Specify FALSE if the volume has not been previously initialized by NOS/VE or has been reformatted; otherwise, specify TRUE. The default value is TRUE.

recover_system_set	A value that specifies that only the system device is to be initialized during this installation deadstart.
set_name	An optional value that specifies the name of the system set. The default value is NVESET.

For example:

```
insd vsn001,true,recover_system_set
```

WARNING

It is strongly recommended that you retain the original name of the system set. However, if it is necessary to change the name of the system set:

1. You will need to manually add each original member of the system set back to the set using the LCU subcommand ADD_VOLUME_TO_SET. Refer to step 11 in this section for more information.
2. You may need to use the CREATE_FAMILY command in step 16 before restoring files belonging to any family other than $SYSTEM. Refer to step 15, for more information.

8. Enter any other system core commands, as directed by your site analyst.
9. Continue the deadstart process by entering:

   ```
   GO
   ```

   Refer to the section called Changing the Configuration at Deadstart for more information concerning the process of changing the physical or logical configuration.
10. Perform this step if any of the following conditions exist; otherwise, go on to the next step:

    - A different system device is to be configured.
    - It is necessary to edit the physical configuration to ensure that a different channel or controller is configured to the existing system device. For examples of editing the configuration, see step 2 of Repair Solution 1 or to step 3 of Repair Solution 5.
    - You are configuring a new system device and the original system device is to remain defined in the physical configuration. You should change the state of the original device to DOWN. Ensure that the new system device is defined in the physical configuration and that it is in the ON state.
    - You recabled and must describe the reconfiguration in the physical configuration.

Select menu option 1 in the following display:

```
You have requested a pause for operator intervention.


NOS/VE RECONFIGURATION MENU - CONTINUATION DEADSTART -

You have the following choices for reconfiguration:

1 - Intervene before installing the physical configuration.

2 - Intervene before activating existing mass storage set members.

3 - Intervene after activating existing mass storage set members.

Enter selection, GO, or ? for HELP.

? 1
```

Edit the physical configuration, then enter QUIT.

11. If you changed the system set name in step 7, select the option to intervene before activating existing mass storage set members. Enter the LCU subcommand ADD_VOLUME_TO_SET for each disk unit whose volume was originally a member of the system set; then enter QUIT:

    ```
    add_volume_to_set recorded_vsn=xxxxxx
    ```

12. When the NOS/VE reconfiguration menu reappears, select the last option (INTERVENE AFTER ACTIVATING EXISTING MASS STORAGE SET MEMBERS) to establish the correct mass storage classes for the system device.

    Using the LCU subcommand CHANGE_MS_CLASS, alter the mass storage class membership of the system device. It is recommended that you delete all classes except classes A and Q. However, if you have decided not to follow the configuration recommendations in chapter 11, Fault Tolerance, reinstate the class membership of your choice for the system device. In the following example, VSN001 was the recorded VSN used to initalize the system device in step 7:

    ```
    change_ms_class vsn001 dc=all ac=(a, q)
    quit
    ```

13. NOS/VE prompts you through each of the remaining steps of the process. You will be presented with two menus. One menu controls restoration of $SYSTEM catalogs, the other menu controls optional restoration of $SYSTEM (and perhaps other) files. At each menu, it is strongly recommended that you read the help information for the menu and for each individual menu selection before proceeding to choose a particular menu selection.

    To obtain help for a menu, enter a question mark (?) at the following prompt:

    ```
    Enter selection or ? for help: ?
    ```

To obtain help for a specific menu selection, enter the number of the selection followed by a question mark. For example, to obtain help for menu item 2, respond to the prompt as follows:

```
Enter selection or ? for help:  2?
```

Information is scrolled to the system console throughout the process. When you have read each message, you must press RETURN or NEXT to continue the process.

NOTE

The menu option to display unreconciled files and catalogs does not identify the name of a missing catalog. It can only identify that a catalog is not online.

The exception message identifying the nonexistence of a catalog defines which mass storage volume the catalog resided on. If you used the same recorded VSN for the system device as it had previously, and there were $SYSTEM catalogs on the previous system device, do not be alarmed that the recorded VSN in the exception message is the system device's recorded VSN. NOS/VE has detected that the original volume is no longer in the configuration and is reporting that fact.

After catalog and file restoration are completed, the process concludes with an automatically executed TERMINATE_SYSTEM command. It is possible that files referenced by the SYSTEM_TERMINATION_PROLOG commands are missing; therefore, some of the commands may fail. This is not a source of concern.

14. Either perform a continuation deadstart without requesting a pause for operator input or enter AUTO when you see the prompt:

    ```
    Enter system core commands
    ```

15. NOS/VE stores the job name counter on the system device. The job name counter is used to generate the unique system job name and system file name for queued input (jobs) and queued output files respectively. The value of the job name counter has been lost due to the intialization of the system device. To ensure that future queued file names do not conflict with those previously created, you must specify a new value for the job name counter. For information on how to specify a new value for the job name counter, refer to the section called Changing the Job Name Counter at the end of this chapter.

16. If there were missing files identified in step 13 and you chose not to restore them at that time, refer to the section called Restoring Files for further information.

WARNING

If all of the following are true:

- You changed the name of the system set in step 7
- You have a multiple set configuration
- There were multiple families on the system set

then you must use the CREATE_FAMILY command to define each family on the correct set prior to restoring families (other than $SYSTEM) that were previously defined on the system set. Refer to the Logical Configuration Utility chapter for more information.

17. Step 13 deleted damage conditions resulting from the restoration of unreconciled catalogs and files. However, damage conditions were only deleted for the files in catalog :$SYSTEM.$SYSTEM. You may want to delete damage conditions for other users who were validated in the $SYSTEM family or for other families. Refer to the sections called Restoring Files and Restoring Catalogs for further information.

18. If you came to this repair solution because of a fault in a disk channel, controller, or unit, go to step 9 of Repair Solution 1 for more information about reinstating the repaired element in the configuration.

19. It is strongly recommended that you schedule a continuation deadstart in the near future. Because you had to restore missing catalogs and files, you may be able to reclaim the capacity assigned to unreconciled files and catalogs that may result from this repair solution. For more information about deleting unreconciled files, refer to the sections in this chapter called Restoring Files and Restoring Catalogs and to the description of the DELETE_UNRECONCILED_FILES system attribute in the NOS/VE System Performance and Maintenance manual, Volume 1.

20. At completion of the system device initialization process a report is generated on file $SYSTEM.RECOVER_SYSTEM_SET_REPORT_yyddd, where yyddd is a Julian date. This file contains:
    - The job log of the system job
    - The listing of the BACPF operation from which the existence of unreconciled files and catalogs was determined, if selected
    - The listings from RESPF operations to restore catalogs and files
    - An echo of the commands executed by the system set recovery process

Repair Solution 3b: Restore Unreconciled $SYSTEM Catalogs

NOS/VE detects that the $SYSTEM master catalog is inaccessible. Although the system device does not require initialization, you must perform the following steps to recover the system set:

1. You are presented with two menus. One menu controls restoration of $SYSTEM catalogs and the other controls optional restoration of $SYSTEM and perhaps other files. At each menu, we strongly recommend that you read the help information for the menu and for each individual menu selection before choosing.

   To obtain help for a menu, enter a question mark (?) at the following prompt.

   ```
   Enter selection or ? for help:
   ?
   ```

   To obtain help for a specific menu selection, enter the number of the selection followed by a question mark. For example, to obtain help for selection 2, enter the following at the prompt:

   ```
   Enter selection or ? for help:
   2?
   ```

   Information is scrolled to the system console throughout the process. When you have read each message, press RETURN or NEXT to continue.

NOTE

The menu option to display unreconciled files and catalogs does not identify the name of a missing catalog. It can only identify that a catalog is not online.

The exception message identifying the nonexistence of a catalog defines which mass storage volume the catalog resided on. If you used the same RECORDED_VSN for the system device as it had previously and there were $SYSTEM catalogs on the previous system device, do not be alarmed that the RECORDED_VSN in the exception message identifies the system device's RECORDED_VSN. NOS/VE has detected that the original volume is no longer in the configuration and is reporting that fact.

After catalog and file restoration are complete, the process concludes with an automatically executed TERMINATE_SYSTEM command. Refer to the NOS/VE Operations manual for more information on automatic system termination. It is possible that files referenced by the SYSTEM_TERMINATION_PROLOG commands are missing. Therefore, some of these commands may fail. This should not be a source of concern.

2. Perform a continuation deadstart requesting pause for operator input. At the following prompt:

   ```
   Enter system core commands
   ```

   Enter the following command to proceed with deadstart:

   ```
   auto
   ```

3. NOS/VE stores the job name counter on the system device. The job name counter is used to generate the unique system job name and system file name for queued input (jobs) and queued output files, respectively. The value of the job name counter has been lost due to the initialization of the system device. To ensure that future queued file names do not conflict with those previously created, you must specify a new value for the job name counter. Refer to Changing the Job Name Counter later in this chapter for an example of computing a new value for the job name counter.

4. If there were missing files identified in step 1 and you chose not to restore them at that time, refer to Restoring Files in this chapter for more information.

5. Step 1 deleted damage conditions resulting from the restoration of unreconciled catalogs and files. However, damage conditions were only deleted for the files in catalog :$SYSTEM.$SYSTEM. You may want to delete damage conditions for other users who were validated in the $SYSTEM family or for other families. Refer to the Restoring Files and Restoring Catalogs sections in this chapter for more information.

6. If you came to this repair solution because of a fault in a disk channel, controller, or disk unit, go to step 9 of repair solution 1 for more information when you are ready to reinstate the repaired element in the configuration.

7. We strongly recommend that you schedule a continuation deadstart in the near future for the following reason:

   Because you had to restore missing catalogs and files, you may be able to reclaim the capacity assigned to unreconciled files and catalogs that may result from this repair solution. Refer to Restoring Files and Restoring Catalogs in this chapter, and the DELETE_UNRECONCILED_FILES system attribute the System Performance and Maintenance manual, Volume 1, for more information on deleting unreconciled files.

8. A report is generated on file $SYSTEM.RECOVER_SYSTEM_SET_REPORT_yydd at the completion of the system device initialization process that contains the following:

   - The job log of the system job.
   - The listing of the BACKUP_PERMANENT_FILES utility operation from which the existence of unreconciled files and catalogs was determined, if selected.
   - The listings from RESTORE_PERMANENT_FILES utility operations to restore catalogs and files.
   - An echo of the commands executed by the system set recovery process.

Solution 4: Installation Deadstart

Use this repair solution when the system is no longer operational as a result of a software fault that could not be corrected by performing a continuation deadstart and initializing the system device (repair solution 3a). Refer to Analyze Critical Software Failure in chapter 12, Failure Analysis. This repair solution should also be used when you want to initialize all volumes in the system set.

Repair Steps

The following steps accomplish an installation deadstart:

1. If CIP is not currently installed, or if you have replaced the volume on which CIP resides, reinstall CIP.

2. The NOS/VE installation deadstart processes for standalone and dual-state systems are different.

   - Standalone

     Mount the NOS/VE deadstart tape and perform the sequence of steps to initiate CIP execution described in the NOS/VE Operations manual. At the CIP Initial Options Display, enter the following to initiate the deadstart:

O	To switch to the Operator Intervention Display.
S	To switch to the Select Deadstart Device Display.
T	To select Tape Deadstart.

   - NOS dual-state

     Mount the NOS/VE deadstart tape on a tape unit reserved for NOS/VE. Initiate deadstart using the NVEffff procedure file.

     At the NOS system console, enter the following DSD command:

     ```
     NVEffff
     ```

     where ffff is the name specified for the procedure associated with deadstarting NOS/VE from tape. This deadstart procedure is created and named using the SETVE procedure. It uses the T=TAPE parameter to specify tape deadstart. The SETVE procedure is documented in the NOS/VE Software Release Bulletin.

   - NOS/BE dual-state

     Mount the NOS/VE deadstart tape on a tape unit reserved for NOS/VE. Initiate deadstart using the NVEffff procedure file. At the NOS/BE system console, enter the following commands to execute the NOS/VE deadstart procedure at an empty control point:

     ```
     n.clear.
     n.x NVE(ffff,id)
     ```

     where n is a control point number, ffff is the name specified for the procedure associated with deadstarting NOS/VE from tape, and id is an optional permanent file identifier (ID) of the deadstart procedure file.

     This deadstart procedure is created and named using the SETVE procedure. It uses the T=TAPE parameter to specify tape deadstart. The SETVE procedure is documented in the NOS/VE Software Release Bulletin.

3. NOS/VE deadstart pauses at the Deadstart and System Device Configuration Selections menu. If option 2 (Deadstart pause for operator input) is not TRUE, select option 2 and set it to TRUE.

4. Check that the values for IOU, channel, controller type, equipment number, and disk unit number describe the path to the system device and the deadstart tape device.

   If they are not correct, enter the number of the value that is incorrect and supply the correct value.

5. When you are satisfied with the values on the display, press the NEXT key to continue the deadstart process.

6. At the following prompt:

   ```
   Enter system core commands
   ```

   enter the following command to initialize the system set:

   ```
   initialize_system_device <name>,<boolean>,,<set_name>
   ```

   This command contains the following elements:

name	Specifies the recorded volume serial number (VSN) to assign to the volume on the system device.
boolean	Specifies whether or not the software flaws recorded on this volume are to be retained. Specify FALSE if the volume has not been previously initialized by NOS/VE or if it has been reformatted; otherwise, specify TRUE. The default is true.
set_name	An optional value that specifies the name of the system set. The default name is NVESET. It is recommended that you retain the original name of the system set. However, if it is necessary to change the name of the system set, you may need to use the CREATE_FAMILY command in step 15 before restoring files.

   For example, enter:

   ```
   initialize_system_device vsn001,true
   ```

7. Enter any other system core commands.

8. Continue the deadstart process by entering:

   ```
   GO
   ```

9. Refer to Changing the Configuration at Deadstart later in this chapter for information concerning the process of changing the physical or logical configuration.

10. If you came to this repair solution because of a fault in a disk channel, controller or disk unit, select menu option 1 from the display below and perform the necessary physical reconfiguration; otherwise, go to step 11. Refer to step 2 of repair solution 1 for examples of editing the configuration.

    ```
    You have requested a pause for operator intervention.

    NOS/VE RECONFIGURATION MENU - SYSTEM SET INITIALIZATION -

    You have the following choices for reconfiguration:

     1 - Intervene to change the physical configuration.

     2 - Intervene to change the logical configuration.

    Enter selection, GO, or ? for HELP
    ? 1
    ```

11. You may have a need to change the logical configuration. Choose the menu item to intervene to change the logical configuration (this is either menu option 1 or 2, depending on your choice in step 10), if any of the following apply; otherwise, enter GO to continue the deadstart:

    - You have no prolog. You need to enter the LCU subcommands to prepare your disk volumes for system use. Refer to chapter 3, Logical Configuration Utility, for more information on the LCU subcommands INITIALIZE_MS_VOLUME, CHANGE_MS_CLASS, and ADD_VOLUME_TO_SET.

    - Your prolog is incomplete. This may occur when your deadstart tape does not correctly reflect your latest mass storage class membership. Refer to the Configuration Recommendations section in chapter 11, Fault Tolerance, for more information on mass storage class.

    - You need to define a flaw on a volume that your prolog has initialized. Normally, this is not necessary because retention of flaws is the default. Refer to the LCU subcommand DEFINE_MS_FLAW in chapter 3, Logical Configuration Utility, for more information on flaw retention.

    Configuration prologs on your deadstart tape automatically initialize your volumes, change mass storage class assignments, and add volumes to a set. If you have a configuration prolog on your deadstart tape, the prolog has already been executed by the time the following LCU prompt is given:

    ```
    LCU/
    ```

    If the prolog is in error, each subcommand in the LCU_MAINFRAME_SUBCOMMANDS prolog file is executed and each abnormal status is displayed. You must enter NEXT or RETURN to acknowledge each reported error. Your prolog may be in error because you have chosen a new system device using the INITIALIZE_SYSTEM_DEVICE system core command and the prolog tried to initialize it.

    Make your changes to the logical configuration and enter QUIT to terminate the LCU interaction.

12. Select the option to activate the system for system console usage. Ignore any errors displayed at this time. The failures are likely due to missing files.

13. NOS/VE deadstart then completes and displays the message:

```
--- SYSTEM ACTIVATION COMPLETE ---
```

14. Determine the most recent catalog backup for the system set.

    Your most recent catalog backup may exist in any of the following forms. Refer to your records to determine which of the most recent of your backups may contain the following catalogs:

    - A catalog-only backup created by the CREATE_CATALOG_BACKUP command.
    - A partial backup created by the CREATE_PARTIAL_BACKUP command.
    - A full backup created by the CREATE_FULL_BACKUP command.

    However, if you exclude catalog information from your partial backups and full backups by specifying BACKUP_CATALOGS=FALSE on the CREATE_FULL_BACKUP and CREATE_PARTIAL_BACKUP commands, then the most recent catalog-only backup should be restored.

    NOTE

    You must restore catalogs from the most recent backup that contains catalogs. If you do not restore the latest catalog backup, you may lose recent catalog modifications, subcatalogs, and files.

    Use the RESTORE_CATALOGED_FILES subcommand to restore catalogs (and files, if applicable) from this most recent backup. Remember to set the RESTORE_CATALOGS parameter of the RESTORE_CATALOGED_FILES subcommand to TRUE for the catalog restoration and to FALSE for all subsequent class K, partial, and full backups restored in the next step.

15. Restore catalogs from the most recent catalog backup. Use the RESTORE_CATALOGED_FILES subcommand to restore catalogs and files from the backup. For example, enter the following command from the System Operator Utility (SOU) to restore catalogs and files:

```
SOU/restore_cataloged_files restore_catalogs=true ..
SOU../vsn_prefix=FULL vsn_count=40 ..
SOU../increment_scheme=decimal ..
SOU../file_label_type=labelled ..
SOU../type=mt9$6250
```

    WARNING

    If all of the following are true:

    - You changed the name of the system set in step 6
    - You have a multiple set configuration
    - There were multiple families on the system set

    then you must use the CREATE_FAMILY command to define each family on the correct set prior to restoring families (other than $SYSTEM) that were previously defined on the system set. Refer to the Logical Configuration Utility chapter for more information.

16. Restore files from any remaining partial or full backups in the order of most recent to least recent. Specify false for RESTORE_CATALOGS on each restoration to avoid restoring catalogs deleted prior to the most recent catalog backup. From the System Operator Utility (SOU), enter the following command for each backup to be restored:

```
SOU/restore_cataloged_files restore_catalogs=false ..
SOU../vsn_prefix=PART ..
SOU../vsn_count=20 increment_scheme=decimal ..
SOU../file_label_type=labelled ..
SOU../type=mt9$6250
```

17. Refer to the NOS/VE Operations manual for more information on activating a production environment. Make the system operational by entering the command:

```
activate_production_environment
```

18. Install the NOS/VE deadstart file on the system device, if your site normally makes this installation. Perform the installation using the ESTABLISH_DISK_BASED_SYSTEM and COMMIT_NEW_SYSTEM subcommands of the MAINTAIN_DEADSTART_SOFTWARE utility. The MAINTAIN_DEADSTART_SOFTWARE utility is documented in the Deadstart File Software chapter of this manual.
19. If you came to this repair solution because of a fault in a disk channel, controller, or unit, go to step 9 of repair solution 1 for more information about reinstating the repaired element in the configuration.

Solution 5: Recable at Continuation Deadstart

Use this repair solution when the system is no longer operational as a result of a hardware failure in the only controller to which the system device is cabled. Refer to Analyze Use of Another Controller in chapter 12, Failure Analysis. You have determined that there is another controller to which the system device may be cabled and the CE has been called to recable the system device.

You may also use this repair solution as a guide if you are moving one mainframe's system set to another mainframe. You may need to perform more extensive editing of the physical configuration than described in the following section.

NOTE

This repair solution does not apply to an 836 or an 887 system device because the controller is an integral part of the disk unit. If your system device is an 887 disk unit, refer to Analyze Critical Drive Failure in chapter 12, Failure Analysis.

This repair solution may not apply to an 836 system device because the $FA7B4_D controller can only be connected to one disk unit. However, if you do have another $FA7B4_D controller, you could disconnect a disk unit from this controller and connect the system device to it. Alternatively, you could choose a new system device from your remaining 836 or other disk units and proceed to repair solution 3.

Repair Steps

The following steps provide recabling at a continuation deadstart:

1. Have the CE perform the recabling. In a standalone system, if the CIP device is also the system device, you must change the deadstart panel to identify the new channel and/or controller address.

   Refer to the CYBER Initialization Package (CIP) Reference manual for more information on steps to implement CIP.

   If you are moving a system set from one mainframe to another, you must initialize the contents of real memory using CIP before proceeding with this repair solution. Refer to the CYBER Initialization Package (CIP) Reference manual for more information.

2. Initiate a continuation deadstart requesting deadstart pause for operator input. You will need to describe the reconfiguration of the system device to NOS/VE on the Deadstart and System Device Configurations Menu.

   Refer to Changing the Configuration at Deadstart later in this chapter for more information about making physical configuration changes at deadstart.

3. To replace a failed controller with another where both controllers are connected to disk units, begin by selecting menu option 1 in response to the following display:

```
You have requested a pause for operator intervention. .

NOS/VE RECONFIGURATION MENU - CONTINUATION DEADSTART -

You have the following choices for reconfiguration:

 1 - Intervene before installing the physical configuration.

 2 - Intervene before activating existing mass storage set members.

 3 - Intervene after activating existing mass storage set members.

Enter selection, GO, or ? for HELP
? 1
```

In this example, assume that the failed controller is TAN_XM_1 and that a second controller, TAN_XM_2, has been recabled to both the system device (TAN_9853_0) and another disk unit (TAN_9853_1). Assume further that TAN_XM_2 already has two other disk units connected to it and that the unit numbers of all four disk units are different. Enter at the PCU prompt:

```
PCU/edit_physical_configuration
```

Change the state of the faulty controller to DOWN:

```
PCE/change_element_definition tax_xm_1 state=down
```

Use the DISPLAY_CONNECTED_ELEMENTS (DISCE) subcommand to determine the names of the elements connected to TAN_XM_1.

```
display_connected_elements tan_xm_1 type=pa
```

Assume the DISPLAY_CONNECTED_ELEMENTS (DISCE) subcommand displays the following:

```
Element Name       Product         State        Serial Number        VEI

TAN_XM_1           $FA7B5_A        DOWN         221                  TRUE
TAN_9853_0         $9853_1         ON           306                  TRUE
TAN_9853_1         $9853_1         ON           307                  TRUE
```

You then enter:

```
PCE/display_element_definition tan_9853_0
```

Assume the DISPLAY_ELEMENT_DEFINITION (DISED) subcommand displays the following:

```
DEFINE_ELEMENT E = TAN_9853_0 ..
  ELEMENT_IDENTIFICATION = $9853_1 STATE = ON ..
  SERIAL_NUMBER = 306 ..
  PERIPHERAL_CONNECTION = ( ..
  ( TAN_XM_1 0 )) ..
  VERIFY_ELEMENT_IDENTIFICATION = TRUE
```

You then enter:

```
PCE/change_element_definition TAN_9853_0 pc=((tan_xm_2 0))
PCE/display_element_definition TAN_9853_1
```

Assume the DISPLAY_ELEMENT_DEFINITION (DISED) subcommand displays the following:

```
DEFINE_ELEMENT E = TAN_9853_1 ..
  ELEMENT_IDENTIFICATION = $9853_1 STATE = ON ..
  SERIAL_NUMBER = 307 ..
  PERIPHERAL_CONNECTION = ( ..
  ( TAN_XM_1 1 )) ..
  VERIFY_ELEMENT_IDENTIFICATION = TRUE
```

You then enter:

```
PCE/change_element_definition TAN_9853_1 pc=((tan_xm_2 1))
PCE/display_connected_elements tan_xm_2
```

Assume the DISPLAY_CONNECTED_ELEMENTS (DISCE) subcommand displays the following:

```
Element Name       Product        State       Serial Number       VEI

TAN_XM_2           $FA7B5_A       ON          220                 TRUE
TAN_9853_0         $9853_1        ON          306                 TRUE
TAN_9853_1         $9853_1        ON          307                 TRUE
TAN_9853_2         $9853_1        ON          308                 TRUE
TAN_9853_3         $9853_1        ON          309                 TRUE
```

You then enter:

```
PCE/quit
PCU/install_physical_configuration
PCU/quit
```

4. Enter GO when the NOS/VE Reconfiguration Menu reappears.

NOTE

If the NOS/VE system advises you that the $SYSTEM master catalog was recreated and the following menu appears, refer to repair solution 3b to conduct the remainder of the deadstart; otherwise, go to step 5.

```
NOS/VE SYSTEM SET RECOVERY MENU - $SYSTEM FILE RESTORATION -

Choose one of the following selections:
 1 - Display the number of unreconciled files and catalogs in the system set.
 2 - Restore files using the RESTORE_UNRECONCILED_FILES command.
 3 - Display the active volumes for all sets in the system.
 4 - Display command information for RESTORE_UNRECONCILED_FILES.
 5 - QUIT (The system will terminate.)
Enter selection or ? for HELP.
```

5. The system may not be ready for production. As a result of the recabling, you may no longer have access to one or more disk volumes.

   If you are missing a volume that was a member of class K, L, or M, you may want to restore files or catalogs. For more information, refer to the section in this chapter called Restoring Files and Restoring Catalogs. It is not required that you restore files at this time. However, if you are missing a volume belonging to classes K or L, your users may not be able to use the system productively unless you do so. If the missing volume did not belong to classes K or L, you may defer file restoration and wait for access to the volume to be regained by a future recabling.

6. Select the option to activate the system for system console usage.

7. Perform file or catalog restoration as necessary.

8. If you have recabled the system device to the same mainframe as before, proceed to step 10.

   NOS/VE stores the job name counter on the system device. The job name counter is used to generate the unique system job name and system file name for queued input (jobs) and queued output files, respectively. Because you have moved the system device from one mainframe to another, you must change the value of the job name counter. This ensures that future queued file names do not conflict with those previously created by the mainframe to which the system device is now connected. Refer to Changing the Job Name Counter later in this chapter for an example of computing a new value for the job name counter.

9. Refer to the NOS/VE Operations manual for more information on activating a production environment. At the system console, enter:

   ```
   activate_production_environment
   ```

10. Request a CE to diagnose and repair the controller while you are using the system.

11. If you need to recable after the repairs are completed, when it is convenient for you to do so, you must enter the TERMINATE_SYSTEM command at the system console from within the System Operator Utility (SOU) and then repeat this repair solution again starting at step 1.

    If no further recabling is required, you may be able to reinstate any elements that you downed or turned off during this repair solution without performing another continuation deadstart. Refer to repair solution 7 in this chapter for more information.

Solution 6: Change Mass Storage Class

Changing the mass storage class of a volume does not affect the residence of existing files; however, it does affect the residence of newly created files and catalogs and may affect the residence of existing catalogs on a subsequent continuation deadstart.

NOTE

If you remove class Q or J from a volume, the volume does not become less critical until the next continuation deadstart. We strongly recommend that you remove class Q or J from a volume only prior to executing the TERMINATE_SYSTEM command or during a continuation deadstart.

NOS/VE assumes that it can tolerate the loss of a volume that does not belong to class Q or J. If you remove class Q or class J from a volume and the volume becomes unavailable, a system interruption may follow. If you want to remove class J, refer to Moving Catalogs Among Volumes later in this chapter.

You may add class Q or J (or any other class) to the volume of your choice at any time.

This repair solution is recommended for any of the following reasons:

- The VED MS display identified one or more classes that were out of space.
- The VED MS display indicated that one or more volumes had been automatically added to class Q.
- The site analyst decided to move catalogs from one volume to another.

The following solutions each require you to know how to display mass storage class assignments and change the mass storage class of a volume. Both of these capabilities are provided by the LCU and both may be executed during or after deadstart. Refer to chapter 3, Logical Configuration Utility, for a description of the CHANGE_MS_CLASS and DISPLAY_MS_CLASSES subcommands. Refer to Changing the Configuration at Deadstart later in this chapter for more information about changing the configuration at deadstart. To display and change classes, enter the following commands from within the System Operator Utility (SOU):

```
SOU/lcu
LCU/display_ms_classes
LCU/change_ms_class recorded_vsn=<vsn_name> add_classes=<list>
LCU/change_ms_class recorded_vsn=<vsn_name> delete_classes=<list>
LCU/quit
```

Classes Other Than Q Out of Space

The VED MS display identifies classes are out of space with the following message:

```
Classes out of space: N.
<setname> Classes out of space: M
```

You need to pick one or more additional volumes to become members of the classes listed or you need to make additional space available on the volumes that are already members of the classes listed.

Look at the mass storage class display to determine whether there are any volumes that are not already members of the classes that are out of space. If there are volumes that could be added to the class, refer to the Configuration Recommendations section in chapter 11, Fault Tolerance, to ensure that the change you are about to make will not violate these recommendations. Then assign the additional volume(s) to the class(es) identified.

If you have decided to try to make space available on existing members of the class(es) identified, follow the recommendations for managing disk space in the NOS/VE Operations manual.

For permanent file classes J, K, L, and M, if multiple sets are configured, make sure you change the class of a volume on the correct mass storage set. The setname is given on the VED MS display.

Class Q Out of Space

If the following message is displayed in the VED MS display, NOS/VE has automatically added one or more volumes to class Q in an effort to maintain service:

```
   Class Q devices automatically added: n.
```

In the preceding message, n is the number of volumes that were added to class Q since the last deadstart. However, the identities of the volumes that have been added to class Q are not shown in the VED MS display.

If you followed the recommendations under Site Preparation in chapter 11, Fault Tolerance, you recorded the mass storage class assignments in effect at your last deadstart on file $SYSTEM.MAINFRAME.CLASSES_AT_DEADSTART. Compare the current mass storage class assignments with those in effect at the last deadstart. If NOS/VE added class Q to a volume that you think is inappropriate, you may want to make a change in this assignment at your next continuation deadstart.

If any of the original members of class Q are also members of class K or M, make space available on those volumes. Follow the recommendations for managing disk space in the NOS/VE Operations manual.

If none of the original members of class Q are also members of class K or M, the system job created a large temporary file or the global logs are too large and should be terminated.

NOTE

You must not remove class Q from a volume until just prior to executing a TERMINATE_SYSTEM command or during your next continuation deadstart.

Moving Catalogs Among Volumes

You may want to move catalogs from one volume to another for several reasons:

- If you had two or more catalog volumes of a particular class (J or L), you may want to consolidate your catalogs on a single volume of that class. For fault tolerance, we recommend this to reduce the probability that a disk unit fault will make your system unavailable or unusable.

- You may have noticed correctable disk unit or disk volume faults occurring on a catalog volume and you may want to relocate that volume's contents to another volume to avoid losing information if the fault worsens.

NOTE

Moving catalogs among volumes requires a continuation deadstart. If you attempt to move catalogs to a volume that has insufficient space, the continuation deadstart will not complete (hangs).

Repair Steps

The following steps move catalogs among volumes:

1. We recommend that you only move catalogs from one volume at a time. This simplifies the task of finding a volume with sufficient space.

2. Pick a new catalog volume that has as much space available as the old catalog volume. Use one of the following methods to select a volume (the methods are listed in order of preference):

   a. Find a volume that contains temporary files (class N) and/or swap files (class C) and does not belong to classes J, K, L, and M.

   b. Find a volume that belongs to class M (permanent files) but does not belong to classes J, K, and L. If there are multiple candidate volumes, choose the one that has the most available space. You are asked to initialize this volume in step 6. Backup the files on this volume to magnetic tape. In the example below, assume DISK8 belongs to class M and has the most available space.

```
SOU/job job_class=system
JOB/request_magnetic_tape file=$local.backup type=mt9$6250 ring=yes ..
JOB/recorded_vsn=('vol1' 'vol2' 'vol3')
JOB/task ring=3
JOB/backup_permanent_files backup_file=$local.backup
JOB/include_volumes recorded_vsn=disk8 ..
JOB/cycle_selection=multiple_volumes
JOB/sort_users
JOB/backup_all_files
JOB/quit
JOB/taskend
JOB/quit
JOB/jobend
```

   c. Look at the VED MS display and pick a volume belonging to classes C, K, N, or Q (but not J or L) that has the most available space.

3. Make a note of the disk unit names on which the old and future catalog volumes are mounted. The easiest way to determine the disk unit names, if you know the names of the volumes, is to enter the following command at the system console:

```
display_active_volumes
```

   Assume in the steps that follow that the old catalog volume belongs to class L. The old catalog volume is on DISK7 mounted on disk unit green_895_7. The new catalog volume is on DISK8 mounted on disk unit green_895_8.

4. If you picked a volume that belongs either to class C or N, make additional space available on this volume by deselecting active job recovery. To deselect active job recovery, set the system attribute JOB_RECOVERY_OPTION to 3. If you select this approach, inform users that jobs will not be recovered and allow them time to logout.

5. Enter the TERMINATE_SYSTEM command at the system console and perform a continuation deadstart. Select active set validation and optionally deselect active job recovery by entering the following system core commands at the system console:

   ```
   set_system_attribute validate_active_sets 1
   set_system_attribute job_recovery_option 3
   go
   ```

   You may want to refer at this time to Changing the Configuration at Deadstart in this chapter to familiarize yourself with the next two steps.

6. If you used approach 2b, choose menu selection 2 in response to the following display; otherwise, proceed to step 7.

   ```
   You have requested a pause for operator intervention.

   NOS/VE RECONFIGURATION MENU - CONTINUATION DEADSTART -

   You have the following choices for reconfiguration:

    1 - Intervene before installing the physical configuration.

    2 - Intervene before activating existing mass storage set members.

    3 - Intervene after activating existing mass storage set members.

   Enter selection, GO, or ? for HELP
   ? 2
   ```

   In the following command entries, answer YES to both prompts that may follow the INITIALIZE_MS_VOLUME command. Refer to repair solution 2 in this chapter for more information. Enter the following subcommands to initialize the volume that you chose and to add that volume back to a set:

   ```
   lcu
   initialize_ms_volume element_name=<disk_unit_name> ..
     recorded_vsn=disk8 retain_device_flaws=true
   add_volume_to_set recorded_vsn=disk8
   quit
   ```

   NOTE

   The command ADD_VOLUME_TO_SET uses the parameter SET_NAME, which defaults to the system set, to add a volume to a set other than the system set.

7. From the NOS/VE Reconfiguration Menu, select the menu option to intervene after activating existing mass storage set members. This is either menu selection 1 or 3, depending on what you entered in step 6.

   Now enter the following subcommands to change the mass storage class memberships of the two volumes:

   ```
   change_ms_class recorded_vsn=disk7 delete_class=L add_class=M
   change_ms_class recorded_vsn=disk8 delete_classes=all add_class=L
   quit
   ```

   NOTE

   In this example, class M was added to the old catalog volume because it was assumed that you used approach 2b. In general, you want to exchange the class memberships of the two volumes.

8. Enter GO at the NOS/VE Reconfiguration Menu prompt and complete the continuation deadstart. If the continuation deadstart hangs, it is probably due to insufficient space on the new catalog volume. We recommend that you do the following in the event of a hang:

   a. Initiate another continuation deadstart and reenter the system core commands identified in step 5.

   b. Choose menu option 3 on the NOS/VE Reconfiguration Menu. Enter the LCU subcommand DISPLAY_MS_CLASSES and use the display to identify another volume that meets the criteria identified in step 2a. If there are no candidates, add class L (in this example) to a volume that belongs to class J, Q, N, K, or M, in order of preference.

   c. Repeat this step (8) until successful.

Solution 7: Perform Online Repair

Use this repair solution when the system has tolerated the failure of a disk or a magnetic tape drive, controller or channel. In chapter 12, Failure Analysis, refer to the following sections:

- Analyze Noncritical Unit Failure
- Analyze Noncritical Controller Failure
- Analyze Noncritical Channel Failure
- Analyze Magnetic Tape Failure
- Analyze Magnetic Tape Channel Failure

Repair Steps

The following repair steps provide online repair:

1. Use the LCU subcommand CHANGE_ELEMENT_STATE to down the failing channel, controller, or disk unit if it is not already down as a result of the failure analysis or repair solutions you have already performed.

   If you are downing a disk channel; controller, or disk unit and abnormal status is returned, it is probably because a class J or Q volume is no longer accessible as a result of the change. It could also be true that you arrived at this repair solution by a mistake in the failure analysis. Please repeat your analysis or write a documentation PSR against this manual.

   If the manual is in error and you again arrived at this repair solution, you will not be able to perform online repair. However, the CE may be able to use HPA/VE and MALET/VE to diagnose or isolate the failure while the elements are still in use. If the CE can repair the failure, enter the STEP_SYSTEM command in the input line of the critical display window and follow the recommendations of Perform Repair in Step in chapter 12, Failure Analysis.

   If you are downing a magnetic tape channel or controller and there are jobs trying to access the magnetic tape units that are only connected to this channel or controller, NOS/VE rejects the state change. In order to complete the state change, you must terminate the batch jobs that are using the affected units. If there are interactive jobs using the affected units, ask the user to terminate the tape task and detach the tape file.

   Use the VEDISPLAY TAPE_STATUS (VED TS) and VEDISPLAY JOB_STATUS (VED JS) Displays to determine which jobs and users are affected. Once the tape units are no longer assigned to any jobs, repeat the state change of the magnetic tape channel or controller.

If you are downing a magnetic tape unit that is currently assigned to a job, NOS/VE rejects the state change. Refer to the previous discussion to terminate that assignment and then repeat the state change attempt by entering the following commands at the system console:

```
SOU/lcu
LCU/change_element_state <failing element name> state=down
LCU/quit
```

2. If you have downed a disk channel or a controller that provides access to volumes belonging to class J, K, L, M, or Q and there is an alternate access to those volumes, consider backing up the files and catalogs on those volumes as a precaution.

   The CE may request concurrent access to one of these volumes to analyze the channel or controller failure. This should not cause any damage to the data on the volume, but whenever you use faulty hardware there is an element of risk involved.

   Refer to Alternate and Redundant Path Configurations in appendix E, Supported Hardware Products, for more information about alternate and redundant access. The following commands may be used to backup files on only those volumes affected by the repair. In the example below, assume that volumes VSN007 and VSN008 are the volumes that may be affected by the repair. To execute the commands in this example, you must have the SYSTEM_ADMINISTRATION SOU capability and you must execute from within the System Operator Utility.

```
SOU/job job_class=system
JOB/task ring=3
JOB/request_magnetic_tape file=$local.dump ..
JOB/recorded_vsn=('reel1', 'reel2') type=mt9$6250 ring=true
JOB/backup_permanent_files backup_file=$local.dump
JOB/include_volumes recorded_vsns=(VSN007,VSN008)
JOB/backup_all_files
JOB/quit
JOB/taskend
JOB/jobend
```

3. Contact a CE to diagnose and repair the problem while you are using the system.

4. Use this step to reinstate the repaired disk unit and its volume.

NOTE

If a disk unit is to be reinstated, the disk unit's volume has been replaced or damaged, and you have not performed a continuation deadstart since the disk unit failed, you must first perform a continuation deadstart. NOS/VE does not allow the initialization of a volume that has been active since the last deadstart.

If a disk unit is to be reinstated and the VED MS display shows the message `Volume is not available - logging process damaged`, a deadstart is required to reinstate the volume after the repair of its disk unit. This message indicates a potential problem with the NOS/VE system files on the volume. If the volume cannot be activated during the next deadstart, redeadstart specifying the following system attribute:

```
set_system_attribute recover_at_all_costs 1
```

Enter the following at the system console:

```
lcu
```

To reinstate the repaired element, enter the following at the system console:

```
change_element_state <name> state=on
```

where name is the name of the tape channel, controller, or disk unit that was repaired.

If the reinstated element is a disk unit whose volume requires initialization, use additional LCU subcommands to initialize the volume, establish its mass storage class membership, and add the volume to a set at this time. Refer to repair solution 2 in this chapter for an example of the required commands.

NOTE

If the disk volume requires initialization and you enter QUIT before initializing the volume, a continuation deadstart is required before you have another opportunity to initialize the volume.

Enter the following command at the system console:

```
quit
```

Changing the Configuration at Deadstart

Modify the configuration at initial installation, subsequent installations, and at continuation deadstart.

Reconfiguration Menu Contents and Operation

If you request a pause for operator input and enter the GO command to terminate system core command processing, you see a menu similar to the following example. Menu content may vary slightly, depending on the type of deadstart performed.

```
You have requested a pause for operator intervention.

USE_CONFIGURATION_PROLOG selection:  NONE
Configuration prolog to be executed: NONE

NOS/VE RECONFIGURATION MENU - SYSTEM SET INITIALIZATION -

You have the following choices for reconfiguration:

1 - Intervene to change the physical configuration.

2 - Intervene to change the logical configuration.

Enter selection, GO, or ? for HELP.
```

NOTE

All the text the operator sees in the output window during the part of the deadstart where reconfiguration takes place is recorded in the system job's log. Included is text identifying the menu selections made by the operator.

The first line of the display indicates the reason for the appearance of the menu. The reason is either the operator explicitly requesting a pause for operator input or the detection of a problem that requires the operator to solve.

The second line of the display identifies the name of the configuration prolog selected by the USE_CONFIGURATION_PROLOG (USECP) system core command. The value NONE indicates no selection was made. This line is not provided during a continuation deadstart because configuration prologs are not automatically executed during a configuration deadstart.

The third line of the display (`Configuration prolog to be executed: NONE`) identifies the name of the configuration prolog to be executed. The value NONE may occur for several reasons:

- The deadstart file does not contain configuration prologs. That is, the file $LOCAL.PROLOG_LIBRARY is not defined.
- No USECP selection was made and there is no prolog defined that matches the name of the mainframe (the default USECP selection).
- The configuration prolog identified in the USECP selection does not exist in the $LOCAL.PROLOG_LIBRARY file.

The second and third lines of this dislpay are not provided on a continuation deadstart because configuration prologs are not automatically executed during a continuation deadstart.

The fourth line identifies the kind of deadstart that is occurring. One of the following phrases is displayed:

```
SYSTEM SET INITIALIZATION
CONTINUATION DEADSTART
SYSTEM SET RECOVERY
```

A system set initialization menu offers up to two selections and the other kinds of deadstart offer up to three selections. Each selection provides a reconfiguration opportunity at a different step in the NOS/VE configuration process. The opportunities offered by a particular selection for a particular kind of deadstart are discussed elsewhere, either below or in a repair solution.

Menu Operation Example

In the preceding menu, if you choose selection 2, NOS/VE installs the physical configuration without your involvement and then pauses to allow logical reconfiguration as you requested. However, if an error in the physical configuration prevents its automatic installation, NOS/VE pauses to allow physical reconfiguration. After the physical configuration is installed, a second menu is displayed:

```
You have just completed an operator intervention to correct a problem
with the configuration. Because of the original problem, you may have to
perform additional reconfiguration.

USE_CONFIGURATION_PROLOG selection:  NONE
Configuration prolog to be executed: NONE

NOS/VE RECONFIGURATION MENU - SYSTEM SET INITIALIZATION -

You have the following choices for reconfiguration:

 1 - Intervene to change the logical configuration.

Enter selection, GO, or ? for HELP.
```

Reconfiguration at Initial Installation

Reconfiguration at installation deadstart has the following steps:

1. Initiate an installation deadstart. Refer to the NOS/VE Operations manual for more information.

2. Enter the AUTO system core command to terminate system core command processing. You then see the following menu displayed because the deadstart file provided by Control Data is unconfigured:

```
The deadstart file is unconfigured.

You have the following choices:

 1 - Install the default physical configuration.

 2 - Enter the PHYSICAL_CONFIGURATION_UTILITY (PCU) to define
and install your physical configuration.

Enter selection, ? for HELP.
```

Selection 1 causes the installation of the mass storage and tape subsystems used to initialize the system device. Selection 2 allows you to manually define the physical configuration. If you choose selection 2, you see the following information displayed:

```
ENTERING THE PHYSICAL CONFIGURATION UTILITY ...

The file $local.physical_configuration has been created for you.
This file contains the DEFINE_ELEMENT subcommands for the system
disk subsystem and the tape subsystem used for system installation.
The element names and serial numbers of these peripherals have been
invented by NOS/VE; please correct this information now.  The
following commands are used to correct and install the physical
configuration:

EDIT_PHYSICAL_CONFIGURATION
ADD_ELEMENT_DEFINITION        (for each new peripheral)
CHANGE_ELEMENT_DEFINITION     (to correct serial number)
CHANGE_ELEMENT_NAME           (to correct element name)
QUIT
INSTALL_PHYSICAL_CONFIGURATION
QUIT
PCU/
```

While in the PCU, use the command sequence shown above to define your configuration. When the physical configuration is successfully installed, the following message is displayed:

```
The physical configuration has been installed.
```

Because an intervention was necessary to install the physical configuration, the following menu is displayed:

```
You have requested a pause for operator intervention.

USE_CONFIGURATION_PROLOG selection:  NONE
Configuration prolog to be executed: NONE

NOS/VE RECONFIGURATION MENU - SYSTEM SET INITIALIZATION -

You have the following choices for reconfiguration:

 1 - Intervene to change the logical configuration.

Enter selection, GO, or ? for HELP.
```

If you need to change the logical configuration, enter selection 1; otherwise, enter GO to complete the deadstart. If you enter selection 1, you see the following message:

```
ENTERING THE LOGICAL CONFIGURATION UTILITY
LCU/
```

Selection 1 enables you to initialize volumes, change mass storage class, and add volumes to the set. You may need the additional disk capacity to install all the products at this deadstart. Enter the following subcommands once for each volume:

```
initialize_ms_volume element=<unit name> recorded_vsn=<new vsn>
change_ms_class <new vsn> delete_class=<list> add_class=<list>
add_volume_to_set member_vsn=<new vsn> set_name=<mass storage set>
```

After changing the logical configuration, enter QUIT to complete the deadstart.

Reconfiguration at Subsequent Installations

The following steps change the physical or logical configuration at an installation deadstart. It is assumed that your deadstart file contains the definition of your configuration that is to be modified.

1. Initiate an installation deadstart. Refer to the NOS/VE Operations manual for more information.

2. Enter the GO (rather than AUTO) system core command to terminate system core command processing. You then see the following menu displayed.

```
You have requested a pause for operator intervention.

USE_CONFIGURATION_PROLOG selection:  EXAMPLE
Configuration prolog to be executed: EXAMPLE

NOS/VE RECONFIGURATION MENU - SYSTEM SET INITIALIZATION -

You have the following choices for reconfiguration:

1 - Intervene to change the physical configuration.

2 - Intervene to change the logical configuration.

Enter selection, GO, or ? for HELP.
```

3. Choosing menu selection 1 allows you to change the physical configuration before it is installed. If there is a PCU_SUBCOMMANDS file created by the configuration prolog (EXAMPLE) you selected, the subcommands contained on this file are executed prior to your intervention. You see the following displayed, for example:

```
ENTERING THE PHYSICAL CONFIGURATION UTILITY

Executing PCU_SUBCOMMANDS from prolog EXAMPLE.

PCU/
```

Use the following command sequence to change the physical configuration:

```
PCU/edit_physical_configuration
PCE/change_element_definition  (or any other EDIPC subcommand)
PCE/quit
PCU/install_physical_configuration
PCU/quit
```

When the physical configuration is successfully installed, the following message is displayed:

```
The physical configuration has been installed.
```

Because you selected choice 1, the following menu is displayed:

```
You have requested a pause for operator intervention.

USE_CONFIGURATION_PROLOG selection:  EXAMPLE
Configuration prolog to be executed: EXAMPLE

NOS/VE RECONFIGURATION MENU - SYSTEM SET INITIALIZATION -

You have the following choices for reconfiguration:

1 - Intervene to change the logical configuration.

Enter selection, GO, or ? for HELP.
```

4. Selecting choice 2 to change the logical configuration causes the following to be displayed:

```
ENTERING THE LOGICAL CONFIGURATION UTILITY
LCU/
```

The following command sequence is used to prepare each volume for system use.

```
initialize_ms_volume element=<unit name> recorded_vsn=<new vsn>
change_ms_class <new vsn> delete_class=(list) add_class=(list)
add_volume_to_set member_vsn=<new vsn>
```

Enter QUIT to terminate the LCU.

Reconfiguration at Continuation Deadstart

The following steps change the physical or logical configuration at a continuation deadstart:

1. Initiate a continuation deadstart requesting pause for operator intervention. Refer to the NOS/VE Operations manual for more information.

2. Enter the GO (rather than AUTO) system core command to terminate system core command processing. The following menu is displayed:

```
You have requested a pause for operator intervention.

NOS/VE RECONFIGURATION MENU - CONTINUATION DEADSTART -

You have the following choices for reconfiguration:

1 - Intervene before installing the physical configuration.

2 - Intervene before activating existing mass storage set members.

3 - Intervene after activating existing mass storage set members.

Enter selection, GO, or ? for HELP.
```

Selection 1 allows you to modify the physical configuration prior to its installation. Selection 2 allows you to initialize an existing set member or to define a flaw for a volume before the volume is activated.

NOS/VE automatically performs the following after processing selection 2:

- Activates all volumes that are mounted on units that are in the ON state, that are physically accessible, and that are a member of a set.

- Identifies to the operator any candidate volume that cannot be activated. You can make the unit ready and reattempt the activation of the volume.

  If a volume cannot be activated, the operator may choose to continue without the volume or to stop and wait for a repair action. If the operator chooses to continue, the unit on which the volume is mounted is automatically downed by NOS/VE.

  If the volume that could not be activated is the only member of a particular mass storage class, the operator is notified and some other volume must be assigned to this class to complete the deadstart. If the system device fails during deadstart, you must either fix the problem or create a new system device; refer to repair solution 3.

Selection 3 allows you to reconfigure an activated set member using LCU subcommands such as CHANGE_MS_CLASS and DEFINE_MS_FLAW. Although an active volume cannot be initialized, a unit that was previously DOWN or OFF may be turned ON, its volume initialized, and reconfigured at this time.

Restoring Files

You have been referred to this section from this chapter or other chapters because you omitted from the configuration at a continuation deadstart a volume containing permanent files, or a volume has become unavailable since the last deadstart. In either case, you may want to restore files.

Use the following steps to restore files:

- If necessary, restore missing catalogs.
- Determine the most recent file backup.
- Determine the effect of file restoration.
- Decide whether or not to perform restoration.
- Select a method and perform restoration.
- Optionally inform users of the restoration.
- Optionally reinstate volume(s).
- Delete unreconciled files.

1. If necessary, restore missing catalogs.

   If the missing or unavailable volume contained catalogs belonging either to class J or class L, you should first perform the process identified in the Restoring Catalogs section in this chapter. Return to this section when you have completed the catalog restoration.

2. Determine the most recent file backup.

   Determine whether your last partial backup occurred before or after your last full backup. Do not consider a catalog-only backup or a backup of class K permanent files in this determination. Determine the date and time of the completion of the most recent backup for use in the next step. For example, in the next step assume that the last backup completed at 23:30:14.011 on August 31, 1988.

   NOTE

   If your backups are done by set, be sure to find the backup for the correct mass storage set.

   If your system prolog uses files that are on the missing volume, you will not be able to submit batch jobs as shown in the following examples. The batch jobs will hang during login until the required files are made available. If you encounter this difficulty, replace the JOB and TASK commands (in the following examples) with:

   ```
   TASK NAME=<name> RING=3
   ```

3. Determine the effect of file restoration.

   If the missing volume was not a catalog volume or catalogs have been restored as directed in step 1, the catalogs can be interrogated to provide information about the magnitude and the consequences of restoring files at this time. The information in the catalogs is obtained using steps 3a and 3b, as follows:

   a. To determine the extent of lost work should you restore all files, enter the following commands from the System Operator Utility (SOU). Specify the date and time of the last file backup (determined in step 2) on the INCLUDE_CYCLES subcommand.

      ```
      SOU/job job_name=display job_class=system
      JOB/sou capability=system_administration
      JOB/task ring=3
      JOB/backup_permanent_files backup_file=$null
      JOB/include_cycles sc=modified after=1988-08-31.23:30:14.011
      JOB/backup_all_files
      JOB/quit
      JOB/taskend
      JOB/quit
      JOB/jobend
      ```

      The job output contains a list of all files modified since the last backup and a media missing or unavailable volume message for each file cycle affected by the failure. If you should decide to restore files rather than reinstate the missing or unavailable volume, these files represent lost work. The BACKUP_PERMANENT_FILES utility subcommand INCLUDE_CYCLES is discussed in chapter 8, Permanent File Maintenance.

b. To determine how many files may have to be restored, enter the following commands from the System Operator Utility (SOU):

```
SOU/job job_class=system
JOB/sou capability=system_adminstration
JOB/create_command_list_entry entry=$system.osf$sou_library
JOB/display_unreconciled_files catalog=all display_option=all
JOB/quit
JOB/jobend
```

These commands submit a batch job whose job output identifies catalogs and file cycles that are missing or unavailable. If a catalog is missing or unavailable, the files or catalogs it contains are not identified in the output.

NOTE

If you have multiple sets, use the SET_NAME parameter of the DISPLAY_UNRECONCILED_FILES command to display only the unreconciled files that belong to the set being restored.

4. Decide whether to perform restoration.

Based on the information you obtained from running the preceding jobs, you now must decide whether to restore files.

If a volume became unavailable due to a fault in a disk channel, controller, or unit, you need not restore any files at this time unless you cannot wait for repair or the volume has been damaged.

NOTE

If you restore files that were being modified at the time of the failure, these modifications are lost. If you do decide to restore files in this situation, you need not restore all files. Any file that you do not restore is recovered once the undamaged volume is reinstated. Consider restoring only those files required to sustain production. In this category are the class K files that you may have backed up after your installation or upgrade of NOS/VE.

5. Select a method and perform restoration.

If you decide to restore files, you may choose to restore:

- All files that have not been modified since the last backup.
- Individual files on a demand basis.
- All missing or unavailable files.

Regardless of which of the methods you choose, if any of the missing or unavailable volumes belonged to class K, first restore your most recent class K permanent file backup, if you perform this backup. Use the procedure described in step 5a that follows. You may find that this restoration returns a sufficient number of important files to use and that method 5b may be sufficient for the remaining class K files.

WARNING

If there are volumes missing in more than one mass storage set and your backups include files from both sets, files from *both* sets are reloaded.

---

a. Restore all file cycles that have not been modified since the last backup.

   Choosing this option allows you to ultimately recover those files that have been modified since the last backup and to provide access now to all the other missing or unavailable files.

   Use the RESTORE_CATALOGED_FILES command to restore files from your partial and full backup. The backups must be restored in the order of most recent to least recent. This procedure uses the RESTORE_PERMANENT_FILES subcommand RESTORE_EXCLUDED_FILE_CYCLES but does not restore cycles whose last modification date in the catalog disagrees with the backup file. Refer to chapter 8, Permanent File Maintenance, for more information on the RESTORE_EXCLUDED_FILE_CYCLES subcommand.

   In the following example, it is assumed that the restoration occurs on a Friday and the last partial backup occurred on the preceding day. It is further assumed that the partial backup includes all cycles modified since the last full backup taken the previous weekend. The last partial backup required six reels of magnetic tape and the last full backup required 20. The commands are entered from the System Operator Utility.

```
restore_cataloged_files restore_catalogs=false vsn_prefix=thurs ..
  vsn_count=6 file_label_type=labelled type=mt9$6250

restore_cataloged_files restore_catalogs=false vsn_prefix=full ..
  vsn_count=20 file_label_type=labelled type=mt9$6250
```

b. Restore files on a demand basis.

   Restoring files on a demand basis may take less time initially but more time later, depending upon how many files need to be restored. It depends on the probability that a user will reference the missing or unavailable files and whether the user wants the files restored rather than waiting for the repair.

   If you want to restore a file cycle that has not been modified since the last backup of the file, use the following command sequence as an example.

```
SOU/job job_class=system
JOB/sou capability=system_administration
JOB/request_magnetic_tape file=$local.restore type=mt9$6250 ..
JOB../recorded_vsn=('wed001') ring=false
JOB/restore_permanent_files
JOB/set_restore_options require_matching_modification=true
JOB/restore_excluded_file_cycle ..
JOB../file=:nve.ajl.labor_data.1 ..
JOB../backup_file=$local.restore ..
JOB../restore_options=(media_missing volume_unavailable ..
JOB../no_data_defined)
JOB/quit
JOB/quit
JOB/jobend
```

If you want to restore all files for a particular user except those that have been modified since the last backup of the user's files, use the following command sequence as an example.

```
SOU/job job_class=system
JOB/sou capability=system_administration
JOB/request_magnetic_tape file=$local.restore type=mt9$6250 ..
JOB../recorded_vsn=('wed001') ring=false
JOB/restore_permanent_files
JOB/set_restore_options require_matching_modification=true
JOB/restore_excluded_file_cycle catalog=:nve.ajl ..
JOB../backup_file=$local.restore ..
JOB../restore_options=(media_missing volume_unavailable ..
JOB../no_data_defined)
JOB/quit
JOB/jobend
JOB/quit
```

Refer to chapter 8, Permanent File Maintenance, for more information on the SET_RESTORE_OPTIONS subcommand. If you want to restore a file that has been modified since the last backup of the user's files, replace the SET_RESTORE_OPTIONS subcommand in the preceding examples with the following:

```
set_restore_options require_matching_modification=false
```

c. Restore all missing or unavailable files.

Choosing this option means that you cannot recover any of the files that were modified since the last backup. This is the logical choice if the volume or volumes have been damaged and cannot be recovered anyway.

The catalogs may describe files modified since the last backup but before the failure. Therefore, you may have to restore a cycle that has a different modification date than the catalog.

If you have both a volume that is missing (that is, it was omitted from the configuration at a continuation deadstart) and one that is unavailable (that is, it has failed since the last deadstart), you can restore files only on the missing volume, only on the unavailable volume, or on both volumes.

If you have two volumes in the same condition (that is, both missing or both unavailable) and you want to restore files for one volume but not the other, use the RECORDED_VSNS parameter of the RESTORE_UNRECONCILED_FILES command to select the volume to be restored.

NOTE

In the following examples, only the restoration of the most recent backup is shown. You must repeat the RESTORE_UNRECONCILED_FILES command for each partial backup and full backup. The backups must be restored in the order of most recent to least recent. To execute these examples, you must have the SYSTEM_ADMINISTRATION capability and you must execute from the System Operator Utility.

To restore all the missing files without restoring those that are unavailable:

```
SOU/restore_unreconciled_files vsn_prefix=wed vsn_count=3 ..
SOU../file_label_type=labelled type=mt9$6250 ..
SOU../restore_options=(media_missing no_data_defined)
```

To restore all unavailable files without restoring those that are missing:

```
SOU/restore_unreconciled_files vsn_prefix=wed vsn_count=3 ..
SOU../file_label_type=labelled type=mt9$6250 ..
SOU../restore_options=(volume_unavailable no_data_defined)
```

To restore all unavailable files on DISK1 without restoring those that are missing or unavailable on other volumes:

```
SOU/restore_unreconciled_files vsn_prefix=wed vsn_count=3 ..
SOU../recorded_vsns=disk1 file_label_type=labelled type=mt9$6250 ..
SOU../restore_options=(volume_unavailable no_data_defined)
```

To restore all missing files and all unavailable files:

```
SOU/restore_unreconciled_files vsn_prefix=wed vsn_count=3 ..
SOU../file_label_type=labelled type=mt9$6250 ..
SOU../restore_options=(media_missing volume_unavailable ..
SOU../no_data_defined)
```

6. Optionally inform users of the restoration.

   Each file restored with a modification date other than the one described in the catalog is marked as damaged by NOS/VE. An abnormal status is returned to the user who tries to attach this file after it is restored. This alerts the user to the fact that the file data may be other than what the user expects.

   Once notified, the user may again attach the file after using the CHANGE_CATALOG_ENTRY (CHACE) command to delete the RESPF_MODIFICATION_MISMATCH damage condition. Refer to the NOS/VE Commands and Functions manual for more information on the CHANGE_CATALOG_ENTRY command.

   If you do not want users to know that a restoration has occurred with a different modification date, use the CHANGE_CATALOG_CONTENTS command to delete the damage condition from all cycles in this damaged state.

   The time required for the CHANGE_CATALOG_CONTENTS command is proportional to the number of catalogs and files in your system. Refer to the discussion of this command in step 8a because you may also want to delete unreconciled files at this time if the volume must be replaced or initialized.

   To delete the RESPF_MODIFICATION_MISMATCH damage condition from all files in all families, enter the following commands from the System Operator Utility (SOU):

   ```
   SOU/change_catalog_contents catalog=all ..
   SOU../delete_damage_condition=respf_modification_mismatch
   ```

To delete the RESPF_MODIFICATION_MISMATCH damage condition from all files in a particular family (for example, family NVE), enter the following commands from the System Operator Utility (SOU):

```
SOU/change_catalog_contents catalog=:nve ..
SOU../delete_damage_condition=respf_modification_mismatch
```

7. Optionally reinstate volume(s).

   Refer to repair solution 7 in this chapter to reinstate a missing or unavailable volume.

8. Optionally delete unreconciled files. You may possibly regain space by selecting one of the following steps:

   NOTE

   If you delete unreconciled files before reinstating a volume that contains catalogs, you lose all files and subcatalogs described in the missing catalogs. Refer to the Restoring Catalogs section in this chapter.

   If you delete unreconciled files before reinstating a volume that contains permanent files, you lose files that have been modified since your last backup.

   a. You may regain space on your catalog volumes by deleting the catalog entries for files that were not restored and have been lost if:

      - You have initialized a volume or had to replace a volume.
      - All other volumes in the configuration are active (accessible).

      From within the System Operator Utility (SOU), enter:

      ```
      SOU/change_catalog_contents catalog=all delete_unreconciled_files=true
      ```

   b. At the next continuation deadstart you may be able to regain permanent file space on the reinstated volume by deleting unreconciled files if you have:

      - Restored some or all the volume's files.
      - Reinstated a volume (without initializing it).
      - Reinstated to the configuration all the missing or unavailable volumes (or will at the next deadstart).

      Enter the following system core command during the continuation deadstart:

      ```
      set_system_attribute delete_unreconciled_files 1
      auto
      ```

Restoring Catalogs

You have been referred to this section from this chapter or other chapters because you omitted a volume containing catalogs (and possibly permanent files) from the configuration during a continuation deadstart.

Perform the following steps to restore missing catalogs after a continuation deadstart:

- Determine the most recent catalog backup.
- Restore missing catalogs (without restoring files) from most recent backup.
- Terminate catalog restoration.
- Optionally perform a continuation deadstart.
- Optionally restore the remainder of your files.
- Optionally inform users of restoration.
- Perform a catalog-only backup when convenient.

1. Determine the most recent catalog backup.

   Your most recent catalog backup may exist in any of the following forms. Refer to your records to determine which of the most recent of your backups may contain catalogs.

   - A catalog-only backup created by the CREATE_CATALOG_BACKUP command.
   - A partial backup created by the CREATE_PARTIAL_BACKUP command.
   - A full backup created by the CREATE_FULL_BACKUP command.

   However, if you exclude catalog information from your partial backups and full backups by specifying BACKUP_CATALOGS=FALSE on the CREATE_FULL_BACKUP and CREATE_PARTIAL_BACKUP commands, then the most recent catalog-only backup should be restored.

   WARNING

   Ensure that there are no users logged in or capable of logging in before you proceed to the next step. You must restore catalogs from the most recent backup that contains catalogs. If you do not restore the latest catalog backup, you may lose recent catalog modifications, subcatalogs, and files.

   If your backups are done by set, be sure to find the backup for the correct mass storage set.

2. Restore missing catalogs (without restoring files) from the most recent backup.

   The following example restores only catalogs from the most recent backup. From the System Operator Utility (SOU), enter:

   ```
   SOU/restore_unreconciled_catalogs restore_excluded_file_cycles=none ..
   SOU../vsn_prefix=catb vsn_count=2 increment_scheme=decimal ..
   SOU../file_label_type=labelled
   ```

   WARNING

   If there are volumes missing in more than one mass storage set and your backups include catalogs from both sets, catalogs from *both* sets are reloaded.

3. Terminate catalog restoration.

   Once all tapes in the set comprising the most recent catalog backup have been restored, terminate the catalog restoration process by entering the following commands from the System Operator Utility (SOU):

   ```
   SOU/restore_permanent_files
   PUR/set_restore_missing_catalogs operation=end
   PUR/quit
   ```

4. Optionally perform a continuation deadstart.

   There are three reasons why you may want to perform a continuation deadstart at this time:

   a. If you set the system attribute IGNORE_IMAGE to 1 during the last deadstart, we recommend that you perform a continuation deadstart now to ensure that there are no damaged catalogs in the system. Specify the following system core commands at the continuation deadstart:

      ```
      set_system_attribute validate_system_set 1
      auto
      ```

   b. The catalog restoration process marks each restored and recovered catalog. A deadstart is required to clear this mark. If the mark is not cleared and you should for any reason execute another RESTORE_UNRECONCILED_ CATALOGS command at any future time, you may restore catalogs and files that had been previously deleted.

      As a precaution, we recommend that you perform a continuation deadstart at this time to clear the marks from the catalogs before you forget to do so. However, you may defer this to any future deadstart as long as you do not restore catalogs prior to clearing the catalog marks. Enter the following system core command during a continuation deadstart to clear the catalog marks:

      ```
      set_system_attribute validate_system_set 1
      ```

c. If there are no files on the missing catalog volume or you plan to restore all the missing files, you may be able to regain some space on your class K or class M file volumes by deleting files that are no longer described by a catalog but are occupying disk space. It is necessary to perform a continuation deadstart to delete files that can no longer be attached. You may perform the continuation deadstart now or at any future time to regain this capacity. Enter the following system core command during the continuation deadstart:

```
set_system_attribute delete_unreconciled_files 1
```

NOTE

If you delete unreconciled files before restoring missing catalogs, you will lose all files and subcatalogs described by the missing catalogs. If you have files on the missing catalog volume and you delete unreconciled files before reinstating the missing volume, you will lose files that have been modified since your last backup. Refer to the Restoring Files section in this chapter for more information on file recovery.

5. Optionally restore files.

   You have recovered all files that reside on available volumes and that were described by the restored catalogs. If the catalog volume that is missing also contained permanent files, you may restore the missing files now, restore the files later, or wait for the fault to be repaired. Refer to the Restoring Files section in this chapter to analyze the options available to you. When you finish with the Restoring Files section in this chapter, return and complete the following steps.

6. Optionally inform users of restoration.

   All files contained in a restored catalog are placed in the PARENT_CATALOG_RESTORED and RESPF_MODIFICATION_MISMATCH damage conditions. This alerts a user who later attaches the file that the file's attributes, permits, access control logs, creation date/time, last access date/time, or last modification date/time may be inaccurate.

   A file whose parental catalog was restored but whose data was not recovered by catalog restoration remains in the damaged condition (PARENT_CATALOG_RESTORED and RESPF_MODIFICATION_MISMATCH) even though the data is later restored from tape. NOS/VE does not assume that the catalog restoration returned the catalog to the same state it was in before restoration.

   Whenever a user attaches a file that has been placed in a damaged condition, an abnormal status is returned. The abnormal status indicates which of the damage conditions exist.

   If you want your users to be aware of the damage conditions, you may want to inform them of how to delete the damage conditions for all the user's files in the system. For example, enter from the System Operator Utility (SOU):

```
SOUchange_catalog_contents catalog=<user_name> ..
SOU../delete_damage_conditions= ..
SOU../(respf_modification_mismatch parent_catalog_restored)
```

To delete the damage conditions for a specific cycle, enter the following from the System Operator Utility:

```
SOU/change_catalog_entry file=$user.file.5 delete_damage_conditions= ..
SOU../(respf_modification_mismatch parent_catalog_restored)
```

If you do not want your users to be aware of the damage conditions, use the CHANGE_CATALOG_CONTENTS command to delete the damage conditions from all of the files in the system. The time required is proportional to the number of catalogs and files in your system. For example, enter the following commands from the System Operator Utility:

```
SOU/change_catalog_contents catalog=all ..
SOU../delete_damage_conditions= ..
SOU../(respf_modification_mismatch parent_catalog_restored)
```

Refer to additional examples in the Restoring Files section in this chapter.

7. Allow users to log in.

8. Perform a catalog-only backup when convenient.

   If your site has a policy of backing up catalogs without data, perform a catalog backup as soon as convenient. You restored catalogs from tape and may have recovered some number of catalogs from disk (assuming you had multiple catalog volumes). A catalog backup consolidates the restored and recovered catalogs on one set of tapes.

   If your site does not do a catalog-only backup but performs partial backups that include catalog information, catalogs are backed up at the next partial backup. Consider performing a partial backup as soon as possible.

   WARNING

   To be able to recover files and catalogs if another catalog volume fails, back up catalogs as soon as possible after a catalog restoration.

Finding Inconsistent Files

NOS/VE normally recovers modified pages of a file that were in memory when a system interruption occurs and writes them to the file during a continuation deadstart. However, a number of situations might prevent this recovery:

- A power or environmental failure that destroys central memory.
- A mainframe hardware failure that damages central memory.
- A recovery without an image. Refer to the system attribute IGNORE_IMAGE in the NOS/VE System Performance and Maintenance manual, Volume 1.
- A system interruption or TERMINATE_SYSTEM command that occurs at the time that a disk volume is unavailable.
- Omitting a volume from the configuration during a continuation deadstart by downing a disk unit or downing a controller or channel that provides the only access to a volume.

If the recovery of modified data is not performed for a subset of files, NOS/VE does not currently identify which files are inconsistent. However, any file whose last modification date/time is after your last file backup could be inconsistent. To obtain a list of files that have been modified since a particular date and time, you may submit a job similar to the following example.

In this example, the date and time of the last partial backup was chosen as the criteria for the listing. The job output will at least give you a starting point in your investigation; however, it is possible for a file to be attached for write access for a long period of time. Such a file might not appear in the list because it was attached for write access prior to the date and time you chose, but the file might be inconsistent just the same.

```
SOU/job job_name=display job_class=system
JOB/sou capability=system_administration
JOB/task ring=3
JOB/backup_permanent_files backup_file=$null
JOB/include_cycles sc=modified after=1987-08-31.23.30:14.011
JOB/backup_all_files
JOB/quit
JOB/taskend
JOB/quit
JOB/jobend
```

Changing the Job Name Counter

NOS/VE stores the job name counter on the system device. The job name counter is used to generate the unique values for the SYSTEM_JOB_NAME and SYSTEM_FILE_NAME attributes for queued input (jobs) and queued output files, respectively. The circumstances under which the job name counter must be changed are as follows:

- If you initialized the system device, the value of the job name counter has been lost.
- If you moved the system set from one mainframe to another, the job name counter on the system device is no longer appropriate for the mainframe to which it is now connected.
- If you replaced the mainframe's central processor zero (CP0), the job name counter for the mainframe is no longer valid.

In any case, you must change the value of the job name counter to ensure that future queued file names do not conflict with those previously created by the mainframe. To determine a new value for the job name counter, we recommend that you find the most recently generated queued file name that you can for the mainframe in question. The following formula is provided to aid in the generation of the new job name counter:

```
increment=((jobs_per_day+output_files_per_day)*number_of_days*2) +
  (deadstarts*1000)/1000
```

This formula uses the following values:

jobs_per_day	An estimate of the average number of jobs that are processed by the mainframe each day.
output_files_per_day	An estimate of the number of queued output files that are processed by the mainframe each day.
number_of_days	The number of days that have elapsed since the last known queued file in your records was created.
2	A constant that allows for error in the preceding estimates of jobs and output files.
deadstarts	An estimate of the number of deadstarts performed on this mainframe since the last known queued file in your records was created.

For example, suppose you have two mainframes and you move the system set from a CYBER 860 to a CYBER 990. If you do not change the job name counter, queued files created on the CYBER 990 would be created using the job name counter of the CYBER 860 instead. To ensure that queued files previously created on the CYBER 990 do not conflict with those created in the future on the CYBER 990, you must change the job name counter. In this example, assume the following:

- The last known queue file created on the CYBER 990 was $0990_0102_ABC_9081.
- The CYBER 990 processes about 1,500 batch and interactive jobs a day.
- The CYBER 990 processes about 2,000 queued output files a day.
- Four days have elapsed since $0990_0102_ABC_9081 was created.
- Five deadstarts of the CYBER 990 occurred prior to the movement of the system set.

For more information on the Site Analyst Examples online manual, refer to chapter 1, Overview. To access the Site Analyst Examples online manual, enter:

```
/help calculate_job_name_counter manual=site_analyst_examples
```

Select the menu option that copies the example to a file and specify the following for the file name:

```
/$local.calculate_job_name_counter
```

Execute the example procedure providing the information necessary to determine a new job name counter value:

```
/$local.calculate_job_name_counter lkc=abc jpd=1500 ofpd=2000 d=5 nod=4
```

The result is similar to the following display:

```
Current Counter ->>ABC  New counter ->>ACJ
```

To change the mainframe's job name counter to the new value, enter the following command at the system console:

```
change_job_name_counter $ACJ_0000
```

Glossary A

Appendixes

Glossary A

A

Active Configuration

Physical configuration that was installed at the preceding deadstart and that may have since been changed using the CHANGE_ELEMENT_STATE subcommand of the Logical Configuration Utility. Changes that occur in the active configuration after deadstart are copied to a file on the system device and are retained across a continuation deadstart. See also Installed Configuration.

B

Binary Log

Sequential file maintained by NOS/VE that contains data about the system. Entries in the file are time-stamped, and numerical data items are stored in binary format to reduce the size of the file.

C

CIO Channel

Channel that allows the PP that controls the data transfer to execute independently of the data transfer. A CIO channel is only supported in the I4 class of IOUs.

Cluster

Grouping of PPs and channels that is physically distinct from another grouping such that PPs in one cluster cannot access channels in another cluster of the same IOU. The I4 class of IOUs can have one or two CIO clusters.

Command Utility

NOS/VE processor that adds its command table (referred to as its subcommands) to the beginning of the SCL command list. The subcommands are removed from the command list when the processor terminates.

Continuation Deadstart

NOS/VE deadstart that recovers the NOS/VE environment from a mass storage file called the image file.

D

Dedicated Fault Tolerance

System features intended to minimize the negative impact of system errors, as well as to minimize the extent of the repair effort required to recover from errors.

Device Allocation Unit

Unit of space allocation on a spindle of a disk unit.

DMA Channel

Channel that allows data to be transferred directly between central memory and the channel. DMA stands for direct memory access.

E

Element

Addressable hardware entity within a mainframe's physical configuration. See also Physical Configuration Utility and Physical Configuration File.

Express Deadstart Dump (EDD)

Dump that writes the contents of central memory to tape. An EDD is performed from the CYBER Test and Initialization (CTI) utilities option following a system interruption.

F

File Server

NOS/VE product that transfers requests and data between mainframes connected by STORNET or ESM-II extended memory devices.

File Transfer Utility

NOS or NOS/BE utility that routes NOS/VE batch input jobs from NOS or NOS/BE to NOS/VE and routes NOS/VE print files to NOS or NOS/BE for printing. This utility is started automatically during NOS/VE deadstart. When not executing at a control point, this utility is in a rolled-out state.

Flaw

Mass storage space surrounding a volume defect. NOS/VE treats this space as a device allocation unit (DAU). Flawing prevents the DAU from being assigned to another file. Flaws are recorded either automatically or manually.

I

I/O Streaming

See Page Streaming.

Image File

File containing a copy of the real memory assigned to NOS/VE. The image file is written to disk during the recovery phase and is used to verify permanent file device integrity.

Installation Deadstart

NOS/VE deadstart that installs NOS/VE and its products. During an installation deadstart, the deadstart device is initialized.

Installed Configuration

Physical configuration that has been manually created or read from the deadstart file during a deadstart and that has been put into effect by the INSTALL_PHYSICAL_CONFIGURATION subcommand of PCU. See also Active Configuration.

L

Logical Configuration Utility (LCU)

Utility used to define the logical configuration and to initialize disks and make them members of a set.

N

Network Activation

That part of system activation that activates NAM/VE. See also System Activation.

NIO Channel

Channel that does not allow the PP that controls the data transfer to execute other instructions during the transfer.

P

Page Streaming

Form of page fault processing in which multiple pages are read from disk in response to a page fault when the page fault occurs in a file that is being processed sequentially.

Partition

Unit of data in a sequential or byte-addressable file delimited by end-of-partition separators or the beginning- or end-of-information.

PASSON

NOS or NOS/BE job that acts as the communications link between a NOS/VE interactive task and a terminal connected to the NOS or NOS/BE network.

Physical Configuration File

File containing a series of Physical Configuration Utility subcommands that define the mainframe and elements of a physical configuration. This file is created or edited either by the EDIT_PHYSICAL_CONFIGURATION subutility of the PCU, or by the EDIT_FILE utility. See also Physical Configuration Utility.

Physical Configuration Utility (PCU)

Utility used to define a mainframe's attached storage devices and device connections. This utility can be used to call a subutility (EDIT_PHYSICAL_CONFIGURATION) that creates or edits a physical configuration file. See also Physical Configuration File.

Production Activation

That part of system activation that makes the system available to users. See also System Activation.

R

Recovery Phase

Part of the processing of the TERMINATE_SYSTEM command (and some continuation deadstarts) that ensures the integrity of the image file.

S

System Activation

Entire deadstart process consisting of the system initiation, production activation, and network activation phases.

System Administrator

A user assigned a system administration capability.

See also System Job.

System Disk Volume

Mass storage volume that contains the installed NOS/VE system.

System Initiation

That part of system activation that consists of the production activation and network activation phases. See also System Activation, Production Activation, and Network Activation.

System Job

Job running at the system console. This job always executes with a system administration capability. See also System Administrator.

System-Supplied Job Name

Unique, 19-character name the system gives each job submitted.

Related Manuals B

Related Manuals B

Table B-1 lists the titles of all manuals referenced in this manual. The table also includes the titles of any other system, product, or hardware manuals that are directly related to this manual. The Related Manuals appendix of the NOS/VE System Usage manual contains a complete list of NOS/VE manuals.

If your site has installed the online manuals, you can find an abstract of 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 an online NOS/VE manual, log in to NOS/VE and enter the online title on the HELP command. The online titles are listed in table B-1. For example, to see the Site Analyst Examples manual, enter:

```
/help manual=site_analyst_examples
```

Table B-1. Related Manuals

Manual Title	Publication Number	Online Title
CYBER 930 Computer System Guide to Operations Usage	60469560	
CYBER Initialization Package (CIP) CYBER 810A, 830A Reference Manual	60000417	
CYBER Initialization Package (CIP) CYBER 180 Models 840, 850, 860 with IOU AB115A Reference Manual	60000418	
CYBER Initialization Package (CIP) CYBER 840, 850, 860 with IOU AT478/AT481A; CYBER 840A, 850A, 860A, 870A, 990, 990E, 995E Reference Manual	60000419	
CYBER Initialization Package (CIP) Reference Manual	60000420	
CYBER Initialization Package (CIP) Reference Manual	60000421	
CYBER Initialization Package (CIP) Non-Model 8XX/9XX Series Reference Manual	60000422	
Desktop/VE Host Utilities Usage	60463918	
NOS/VE Accounting Analysis System Usage	60463923	
NOS/VE Accounting and Validation Utilities for Dual State Usage	60458910	
NOS/VE LCN Configuration and Network Management Usage	60463917	
NOS/VE Network Management Usage	60463916	
NOS/VE Operations Usage	60463914	

(Continued)

Table B-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Title
NOS/VE System Performance and Maintenance Volume 1: Performance Usage	60463915	
NOS/VE User Validation Usage	60464513	
NOS/VE File Archiving Usage	60463944	
NOS/VE Security Administration Usage	60463945	
NOS/VE File Server For STORNET and ESM-II Usage	60000190	
NOS/VE Site Analyst Examples Online manual		SITE_ ANALYST_ EXAMPLES
NOS/VE Commands and Functions Quick Reference	60464018	SCL
NOS/VE Object Code Management Usage	60464413	
NOS/VE System Usage	60464014	EXAMPLES
NOS/VE Terminal Definition Usage	60464016	
VX/VE Administrator Guide and Reference Tutorial/Usage	60469770	
CDCNET Batch Device User Guide	60463863	CDCNET_ BATCH
CDCNET Configuration Guide	60461550	
CDCNET Network Operations and Analysis	60461520	

(Continued)

Table B-1. Related Manuals *(Continued)*

Manual Title	Publication Number	Online Title
NOS 2 Installation Handbook	60459320	
NOS/VE Diagnostic Messages Usage	60464613	MESSAGES
CYBER 170 Computer Systems Models 815, 825, 835, 845, and 855 CYBER 180 Models 810, 830, 835, 840, 845, 850, 855, 860, and 990 CYBER 845S, 855S, 840A, 850A, 860A, 990E, and 995E General Description Hardware Maintenance	60459960	
CYBER 170 State Virtual State Codes Booklet	60458100	
CPU, CM, IOU Maintenance Registers Codes Booklet	60458110	
HPA/VE Reference	60461930	
Virtual State Volume II Hardware Reference	60458890	

Character Set C

Character Set C

ASCII Character Set

This appendix lists the ASCII character set (refer to table C-1).

NOS/VE supports the American National Standards Institute (ANSI) standard ASCII character set (ANSI X3.4-1977). NOS/VE represents each 7-bit ASCII code in an 8-bit byte. These 7 bits are right justified in each byte. For ASCII characters, the eighth or leftmost bit is always zero. However, in NOS/VE the leftmost bit can also be used to define an additional 128 characters.

If you want to define additional non-ASCII characters, be certain that the leftmost bit is available in your current working environment. The full screen applications (such as the EDIT_FILE utility, the EDIT_CATALOG utility, and the programming language environments) already use this bit for special purposes. Therefore, these applications accept only the standard ASCII characters. In applications in which the leftmost bit is not used, however, you are free to use it to define the interpretation of each character as you wish.

Table C-1. ASCII Character Set

Decimal Code	Hexadecimal Code	Octal Code	Graphic or Mnemonic	Name or Meaning
000	00	000	NUL	Null
001	01	001	SOH	Start of heading
002	02	002	STX	Start of text
003	03	003	ETX	End of text
004	04	004	EOT	End of transmission
005	05	005	ENQ	Enquiry
006	06	006	ACK	Acknowledge
007	07	007	BEL	Bell
008	08	010	BS	Backspace
009	09	011	HT	Horizontal tabulation
010	0A	012	LF	Line feed
011	0B	013	VT	Vertical tabulation
012	0C	014	FF	Form feed
013	0D	015	CR	Carriage return
014	0E	016	SO	Shift out
015	0F	017	SI	Shift in
016	10	020	DLE	Data link escape
017	11	021	DC1	Device control 1
018	12	022	DC2	Device control 2
019	13	023	DC3	Device control 3
020	14	024	DC4	Device control 4
021	15	025	NAK	Negative acknowledge
022	16	026	SYN	Synchronous idle
023	17	027	ETB	End of transmission block
024	18	030	CAN	Cancel
025	19	031	EM	End of medium
026	1A	032	SUB	Substitute
027	1B	033	ESC	Escape
028	1C	034	FS	File separator
029	1D	035	GS	Group separator
030	1E	036	RS	Record separator
031	1F	037	US	Unit separator
032	20	040	SP	Space
033	21	041	!	Exclamation point
034	22	042	"	Quotation marks
035	23	043	#	Number sign
036	24	044	$	Dollar sign
037	25	045	%	Percent sign
038	26	046	&	Ampersand
039	27	047	'	Apostrophe

(Continued)

Table C-1. **ASCII Character Set** *(Continued)*

Decimal Code	Hexadecimal Code	Octal Code	Graphic or Mnemonic	Name or Meaning
040	28	050	(	Opening parenthesis
041	29	051	)	Closing parenthesis
042	2A	052	*	Asterisk
043	2B	053	+	Plus
044	2C	054	,	Comma
045	2D	055	-	Hyphen
046	2E	056	.	Period
047	2F	057	/	Slant
048	30	060	0	Zero
049	31	061	1	One
050	32	062	2	Two
051	33	063	3	Three
052	34	064	4	Four
053	35	065	5	Five
054	36	066	6	Six
055	37	067	7	Seven
056	38	070	8	Eight
057	39	071	9	Nine
058	3A	072	:	Colon
059	3B	073	;	Semicolon
060	3C	074	<	Less than
061	3D	075	=	Equals
062	3E	076	>	Greater than
063	3F	077	?	Question mark
064	40	100	@	Commercial at
065	41	101	A	Uppercase A
066	42	102	B	Uppercase B
067	43	103	C	Uppercase C
068	44	104	D	Uppercase D
069	45	105	E	Uppercase E
070	46	106	F	Uppercase F
071	47	107	G	Uppercase G
072	48	110	H	Uppercase H
073	49	111	I	Uppercase I
074	4A	112	J	Uppercase J
075	4B	113	K	Uppercase K
076	4C	114	L	Uppercase L
077	4D	115	M	Uppercase M
078	4E	116	N	Uppercase N
079	4F	117	O	Uppercase O

(Continued)

Table C-1. ASCII Character Set *(Continued)*

Decimal Code	Hexadecimal Code	Octal Code	Graphic or Mnemonic	Name or Meaning
080	50	120	P	Uppercase P
081	51	121	Q	Uppercase Q
082	52	122	R	Uppercase R
083	53	123	S	Uppercase S
084	54	124	T	Uppercase T
085	55	125	U	Uppercase U
086	56	126	V	Uppercase V
087	57	127	W	Uppercase W
088	58	130	X	Uppercase X
089	59	131	Y	Uppercase Y
090	5A	132	Z	Uppercase Z
091	5B	133	[	Opening bracket
092	5C	134	\	Reverse slant
093	5D	135	]	Closing bracket
094	5E	136	^	Circumflex
095	5F	137	_	Underline
096	60	140	`	Grave accent
097	61	141	a	Lowercase a
098	62	142	b	Lowercase b
099	63	143	c	Lowercase c
100	64	144	d	Lowercase d
101	65	145	e	Lowercase e
102	66	146	f	Lowercase f
103	67	147	g	Lowercase g
104	68	150	h	Lowercase h
105	69	151	i	Lowercase i
106	6A	152	j	Lowercase j
107	6B	153	k	Lowercase k
108	6C	154	l	Lowercase l
109	6D	155	m	Lowercase m
110	6E	156	n	Lowercase n
111	6F	157	o	Lowercase o
112	70	160	p	Lowercase p
113	71	161	q	Lowercase q
114	72	162	r	Lowercase r
115	73	163	s	Lowercase s
116	74	164	t	Lowercase t
117	75	165	u	Lowercase u
118	76	166	v	Lowercase v
119	77	167	w	Lowercase w

(Continued)

Table C-1. ASCII Character Set *(Continued)*

Decimal Code	Hexadecimal Code	Octal Code	Graphic or Mnemonic	Name or Meaning
120	78	170	x	Lowercase x
121	79	171	y	Lowercase y
122	7A	172	z	Lowercase z
123	7B	173	{	Opening brace
124	7C	174	|	Vertical line
125	7D	175	}	Closing brace
126	7E	176	~	Tilde
127	7F	177	DEL	Delete

Assigning Files to a Specific Device D

Assigning Files to a Specific Device D

Normally, NOS/VE automatically assigns files and catalogs to specific mass storage (disk) device classes. Table D-1 summarizes the default assignments that NOS/VE makes to each device class. For more information about mass storage classes, refer to chapter 3, Logical Configuration Utility.

Table D-1. Default Mass Storage Class Assignments

Class	Default Assignment
C	System swap files.
J	Catalogs for the $SYSTEM user in all families.
K	Permanent files for the $SYSTEM user in all families; NOS/VE products that are critical to providing system service.
L	Catalogs for all users except $SYSTEM users.
M	Nonservice-critical permanent files belonging to the user :$SYSTEM.$SYSTEM and permanent files belonging to families other than $SYSTEM.
N	User temporary files and system segments private to every job except the system job.
Q	All temporary files created by the system job.

By using the REQUEST_MASS_STORAGE command, you can place a file on a volume other than the one to which it may be assigned by default. This command is provided primarily to allow you to place files on specific volumes for fault tolerance reasons. With this command, you can improve file availability by placing important files on a small number of volumes. This minimizes the chance that a disk failure will affect those files. For more information about fault tolerance features and how to use them, see Part IV, Fault Tolerance Configuration and Failure Analysis.

In addition, you can use this command to improve performance either by controlling file residence or by specifying a larger allocation size for a file. You can improve performance by separating files that are concurrently accessed. This action takes advantage of NOS/VE's ability to perform seek optimization. Seek optimization refers to the process of overlapping data transfer on one device with positioning on another device; it is possible only when multiple disk devices are configured per channel. If your system has multiple disk channels, separating files allows for more uniform use of channels. To separate or colocate files, use the INITIAL_VOLUME parameter.

You can also improve the elapsed time of some applications by selecting a larger allocation size for a large file that is accessed sequentially. For more information about how to do this, see the description of the ALLOCATION_SIZE parameter of the REQUEST_MASS_STORAGE command.

REQUEST_MASS_STORAGE Command

Purpose Assigns a file to a specific mass storage device.

Format **REQUEST_MASS_STORAGE** or
REQMS
  FILE=file
  ALLOCATION_SIZE=integer
  FILE_CLASS=character or keyword
  INITIAL_VOLUME=name or keyword
  TRANSFER_SIZE=integer
  VOLUME_OVERFLOW_ALLOWED=boolean
  STATUS=status variable

Parameters FILE or F

Specifies the name of the file to be registered in a catalog and assigned to a mass storage device class. The cycle reference must not refer to $HIGH or $LOW. The default cycle reference is 1 for a temporary file and for a permanent file when there are no existing cycles. Otherwise, a value one higher than the current highest cycle is used. This parameter is required.

ALLOCATION_SIZE or *AS*

Specifies an integer indicating the amount of contiguous mass storage space, in bytes, to be allocated to the file each time additional space is needed. The system uses the value of this parameter as a guide in selecting the amount of space to allocate for this file. Mass storage space actually allocated may vary slightly from the value specified. Values for this parameter range from 16,384 to 16,777,215. By default, the system chooses the allocation size associated with the initial volume. Refer to the Remarks section for a description of the restrictions on the use of this parameter.

If the value specified for this parameter exceeds the value of the MAXIMUM_ALLOCATION_SIZE system attribute, the latter value is used. For a description of this system attribute, see the NOS/VE System Performance and Maintenance manual, Volume 1.

FILE_CLASS or *FC*

Specifies the class of the file to be assigned. The value specified can be any alphabetic character (either uppercase or lowercase). Certain alphabetic characters are associated with specific mass storage classes as described under the CHANGE_MS_CLASS subcommand in chapter 3, Logical Configuration Utility.

NOS/VE selects a volume belonging to the class specified by this parameter. An abnormal status is returned if no candidate volume belongs to the specified class.

You can specify one of the following keywords; by default, NOS/VE places the file on the volume identified by the INITIAL_VOLUME parameter:

PRODUCT or M

Refers to a file whose unavailability affects only users of that product. This keyword is associated with class M mass storage devices.

SERVICE_CRITICAL_PRODUCT or K

Refers to a file whose unavailability may deny service to all users even though the system itself is not affected. Examples are files that are attached by the system prolog, the terminal definition libraries, and command libraries. This keyword is associated with class K mass storage devices.

SYSTEM_CRITICAL_FILE or Q

Refers to a file whose absence will cause a system interrupt. This keyword is associated with mass storage class Q.

SYSTEM_PERMANENT_FILE or K

Refers to a $SYSTEM permanent file. This type of file is associated with mass storage class K.

TEMPORARY_FILE or N

Refers to a user temporary file. This type of file is associated with mass storage class N.

USER_PERMANENT_FILE or M

Refers to a user permanent file. This type of file is associated with mass storage class M.

INITIAL_VOLUME or *IV*

Specifies the volume serial number (VSN) of the mass storage volume or volumes to which the file is to be assigned. The VSN is the same name defined for the volume on the INITIALIZE_MS_VOLUME subcommand. If this volume is not affiliated with the file class specified in the FILE_CLASS parameter, the volume is full, or the volume does not exist in the active configuration, the system rejects this command.

You can specify this volume directly by supplying its recorded VSN, or you can specify the SYSTEM_DEVICE (or SD) keyword. Specifying the latter ensures that the file specified by the FILE parameter is available whenever the system is available. However, if the file is accessed frequently, consider putting it elsewhere for performance reasons.

If you specify the FILE_CLASS parameter, the volume specified by this parameter must belong to the file class specified or the request will be rejected.

By default, one of the volumes is chosen that belongs to the class specified by the FILE_CLASS parameter.

TRANSFER_SIZE or *TS*

Specifies the amount of data, in bytes, that is read from the file's mass storage device whenever the system detects that the file can be read at the device rate (the streaming rate). The specified value is automatically rounded up to the nearest power of 2. You can specify an integer from 16,384 to 2,147,483,648. By default, the system chooses the transfer size associated with the initial volume.

VOLUME_OVERFLOW_ALLOWED or *VOA*

Specifies whether the file can be assigned to more than one volume. The default is TRUE.

TRUE

The file is permitted to span any volume subject to validation and file class restraints.

FALSE

The file is confined to the initial volume to which it is assigned. FALSE may be specified only by the system administrator (that is, a user validated with a system administration capability) or by a job having maintenance privileges.

STATUS

Returns the completion status for this command.

Remarks

- This command provides the following ways to select the initial volume to which a file is assigned; these ways are based on the presence or absence of the INITIAL_VOLUME and FILE_CLASS parameters:
  - If you specify only the INITIAL_VOLUME parameter, the volume it identifies is selected.
  - If you specify only the FILE_CLASS parameter, the file is assigned to a member of the designated class.
  - If you specify both the INITIAL_VOLUME and FILE_CLASS parameters, the specific volume is selected only if the mass storage volume is a member of the class specified.
  - If you specify neither the INITIAL_VOLUME nor the FILE_CLASS parameter, the system selects a candidate volume associated with the default file class. The default file class of a permanent file is class K for user $SYSTEM and class M for other users. The default file class of a temporary file is class Q for the system job and class N for other jobs..
- The ALLOCATION_SIZE and TRANSFER_SIZE parameters are not used to select the initial volume. Once the initial volume is selected, the allocation size and transfer size specifications are adjusted to reflect the size supported by the initial volume.
- Only the system administrator may specify classes C, J, K, L and Q either directly or indirectly using the keyword values for the parameter.
- A task executing in rings 6 to 4 may specify classes M, N, U, V, W, X, Y, and Z for a permanent or temporary file.
- A task executing in rings 13 to 7 may only specify class M for a permanent file and class N for a temporary file.

- If you specify neither the FILE_CLASS nor the INITIAL_VOLUME parameter, NOS/VE places the file on a volume that belongs to the class that is appropriate for the file and the job in which the file is created. For a description of the default NOS/VE file assignments, refer to table D-1, Default Mass Storage Class Assignments.

- If you specify the INITIAL_VOLUME parameter, the following conditions apply:

  - Only the system administrator may place a file (temporary or permanent) on any volume in the configuration, regardless of the execution ring and the volume's mass storage class membership.

  - If you are executing with an ENGINEERING_OPERATION validation capability, you may place a temporary file on any volume in the configuration, regardless of the execution ring and the volume's mass storage class membership.

  - A task executing in rings 6 to 4 may specify any volume that belongs to one of the following classes: M, N, U, V, W, X, Y, and Z. If the volume specified does not belong to the file's default file class, you must also include the FILE_CLASS parameter, specifying one of the above classes to which the volume belongs.

    For example, suppose you want to place permanent file MY_FILE on volume VSN002, which belongs to file classes N and U but does not belong to file class M (the default file class for a permanent file). To do so, enter:

    ```
    request_mass_storage file=my_file initial_volume=vsn002 ..
      file_class=n
    ```

  - A task executing in rings 13 to 7 may only place a permanent file on a volume that belongs to class M and only place a temporary file on a volume that belongs to class N.

- Cautionary notes on the use of the REQUEST_MASS_STORAGE command:

  - The command creates the file specified by the FILE parameter and leaves it attached for all modes of access and no sharing. Therefore, you need to use the DETACH_FILE command to terminate your access to the file.

  - A site that uses the INCLUDE_VOLUME subcommand of the BACKUP_PERMANENT_FILES utility to back up files by volume should also be sure to include those volumes containing files assigned by the REQUEST_MASS_STORAGE command. That is, sites like this may need to include volumes other than those that are members of the permanent file classes K and M.

- Using allocation sizes larger than the default value (16,384 bytes) has the following disadvantages:
  - It may take longer for the system to allocate space when a file is created. If a device contains files with different allocation sizes, it becomes harder to locate enough contiguous space to satisfy the file's requirement as the devices fill up.
  - It wastes capacity because the last allocation unit is only partially used (unless the file size is an integral number of allocation units).
  - For security reasons, NOS/VE initializes each allocation unit of a file when at least one page is first written to that allocation unit. This prevents users from reading information that they did not write. The larger the allocation size you choose, the more disk revolutions are required to write the last page of a sequential file.
- To use the allocation size capability effectively, follow these guidelines:
  - Reserve some disks exclusively for large files. For example, remove the devices from membership in classes C, J, K, L, M, N, and Q. Then, assign large files to these volumes using the INITIAL_VOLUME and ALLOCATION_SIZE parameters. If you back up permanent files by volume, remember to include these volumes in your backup.
  - Use this capability for long files that are read sequentially and frequently. NOS/VE reads the entire allocation unit of a sequential file when the first page fault occurs within the allocation unit.
- The INITIAL_VOLUME parameter can be used to improve reliability or performance by separating files on different volumes. For example, you may want to separate a database from its journals for both reliability and performance reasons.
- Specifying no volume overflow is typically used in conjunction with a request for a specific mass storage volume. This is generally done for fault tolerance reasons. For example, specify no volume overflow if you want a file required for a particular application to reside exclusively on a single volume (so that if one volume fails, you know immediately whether the application is usable). However, assigning files to the same volume may have a negative impact on the application's performance.
- The parameters for this subcommand are functionally identical to corresponding parameters for the SET_RESTORE_OPTIONS subcommand of the RESTORE_PERMANENT_FILES utility described in chapter 8, Permanent File Maintenance.

Program Description

The REQUEST_MASS_STORAGE command is packaged in the operator command library. You may authorize other users to use this command by creating an object library that contains a program description for the command, specifying the ring attributes of the library to select the desired privilege, and creating a file permit to allow specified groups of users to use the library. For example, use the following commands to create a command library on file :NVE.JIM.REQMS_COMMAND:

```
create_object_library
  create_program_description names=(request_mass_storage,reqms) ..
    starting_procedure=rmp$request_mass_storage_cmd ..
    library=osf$task_services_library scope=gate
  generate_library library=:nve.jim.reqms_command
quit
```

If you want to prevent users from using classes other than M or N while allowing them to use the INITIAL_VOLUME, ALLOCATION_SIZE, and TRANSFER_SIZE parameters, change the ring attributes of the file as follows:

```
change_file_attributes file=:nve.jim.reqms_command ring_attributes=(11 11 13)
```

If you want users to be able to specify file classes other than M or N, or if you want to assign a file to a volume that does not belong to class M or N, you must change the ring attributes of the file as follows:

```
change_file_attributes file=:nve.jim.reqms_command ring_attributes=(6 6 13)
```

To create a file permit to authorize a group of users to execute the REQUEST_MASS_STORAGE command from this library, enter the following:

```
create_file_permit file=:nve.jim.reqms_command group=user user=bill ..
  access_mode=execute share_modes=(read execute)
```

User BILL, who runs at ring 11, can now have access to the ring 6 capabilities of the REQUEST_MASS_STORAGE command. After adding :NVE.JIM.REQMS_COMMAND to his command library, he can assign permanent files to class U, as in the following example:

```
create_command_list_entry entry=:nve.jim.reqms
request_mass_storage file=$user.file file_class=u
```

In addition, user BILL can use the INITIAL_VOLUME parameter to assign permanent files to volumes belonging to a class other than A, C, J, K, L, and Q. The following examples illustrate this capability (refer also to the sample configuration in the next subsection):

```
request_mass_storage file=$user.file initial_volume=vsn005 file_class=u

request_mass_storage file=$user.file1 initial_volume=vsn002 file_class=n
```

Sample Configuration

The following example of a fault tolerant mass storage configuration illustrates which users may place files on specific volumes. In this example, class U is defined for a privileged application.

VSN	A	C	J	K	L	M	N	Q	U
VSN001	x							x	
VSN002	x		x	x			x		
VSN003	x	x			x				
VSN004	x	x				x			
VSN005	x						x		x
VSN006	x					x	x		
VSN007	x	x				x	x		

The following considerations apply to the example:

- The system administrator may place a file (temporary or permanent) on any volume in the configuration.
- A user with an ENGINEERING_OPERATION validation capability can place a temporary file on any volume in the configuration.
- Only the system administrator may place a permanent file on VSN001 and VSN003, because these volumes belong to no classes other than A, C, J, K, L and Q.
- Only the system administrator and users executing in rings 6 to 4 may place a permanent file on VSN002 and VSN005. The reason a user executing in rings 6 to 4 may do so is that these volumes belong to a class other than A, C, J, K, L, and Q.
- Only the system administrator, a user with an ENGINEERING_OPERATION validation capability, and a user executing in rings 6 to 4 may place a temporary file on VSN004. The reason a user executing in rings 6 to 4 may do so is that this volume belongs to a class other than A, C, J, K, L, and Q. For example:

  ```
  request_mass_storage file=$local.file_3 file_class=m initial_volume=vsn004
  ```

- Any user may assign a permanent file to volumes VSN004, VSN006, and VSN007 because these volumes belong to class M.
- Any user may assign a temporary file to volumes VSN002, VSN005, VSN006, and VSN007 because these volumes belong to class N.

RMP$REQUEST_MASS_STORAGE Program Interface

Purpose Registers a file in a specified catalog, assigns the file to a mass storage device with particular attributes, and attaches the file to the requesting job. Refer to the REQUEST_MASS_STORAGE Command section in this chapter for restrictions on how the request may be used.

Format **RMP$REQUEST_MASS_STORAGE (file, allocation_size, estimated_file_size, file_class, initial_volume, volume_overflow_allowed, status)**

Parameters **file**: fst$file_reference

See the description of the FILE parameter of the REQUEST_MASS_STORAGE command.

allocation_size: rmt$allocation_size

See the description of the ALLOCATION_SIZE parameter of the REQUEST_MASS_STORAGE command.

estimated_file_size: amt$file_byte_address

Currently unimplemented. Specify rmc$unspecified_file_size for this parameter.

file_class: rmt$mass_storage_class

Specifies the class of the file which is to be assigned. NOS/VE supports up to 26 classes of files. Each class is identified by an alphabetical character (uppercase and lowercase characters are equivalent). NOS/VE selects a volume which belongs to the class specified by this parameter. An abnormal status is returned if no candidate volume belongs to the specified class.

Only the system administrator may specify the following values:

rmc$msc_system_swap_files	(C)
rmc$msc_system_catalogs	(J)
rmc$msc_system_permanent_files	(K)
rmc$msc_user_catalogs	(L)
rmc$msc_system_critical_files	(Q)

A task executing in rings 6 to 4 may specify file classes M, N, U, V, W, X, Y, and Z.

A task executing in rings 13 to 7 may specify rmc$user_permanent_files (M) for a permanent file and rmc$user_temporary_files (N) for a temporary file.

The value rmc$unspecified_file_class causes NOS/VE to place the file on the volume specified by the INITIAL_VOLUME parameter.

If you specify the value rmc$unspecified_vsn for the INITIAL_VOLUME parameter, and rmc$unspecified_file_class for the FILE_CLASS parameter, NOS/VE assigns the file to a volume that belongs to the class appropriate for the file and the job in which the file is created. Refer to table D-1 for information on NOS/VE default file assignments.

initial_volume: rmt$recorded_vsn

Identifies a specific mass storage volume to which this file is assigned.

If volume overflow is not allowed, the entire file resides on this volume. Otherwise, this volume is the initial volume assigned to the file. Refer to the VOLUME_OVERFLOW_ALLOWED parameter.

This request is rejected if the requested volume has no space available or the volume does not exist in the active configuration.

If you specify a value other than rmc$unspecified_file_class for the FILE_CLASS parameter, the request is rejected unless the volume specified by this parameter belongs to the file class specified.

As the system administrator, you can use this parameter to place a file on any volume in the configuration. However, you need to specify rmc$unspecified_file_class for the FILE_CLASS parameter.

If you are executing with an ENGINEERING_OPERATION validation capability, you can use this parameter to place a temporary file on any volume in the configuration whether or not the volume belongs to the temporary file class (N). However, you need to specify rmc$unspecified_file_class for the FILE_CLASS parameter.

A task executing in rings 6 to 4 may specify any volume that belongs to one of the following classes: M, N, U, V, W, X, Y, and Z. If the volume specified does not belong to the file's default file class, you must also include the FILE_CLASS parameter, specifying one of the above classes to which the volume belongs.

For example, suppose you want to place permanent file MY_FILE on volume VSN002, which belongs to file classes N and U but does not belong to file class M (the default file class for a permanent file). To do so, enter:

```
request_mass_storage file=my_file initial_volume=vsn002 file_class=n
```

A task executing in rings 13 to 7 may specify any volume that belongs to class M for a permanent file or class N for a temporary file.

The value rmc$unspecified_vsn causes all volumes belonging to a file class (specified or defaulted) to be considered as candidates.

volume_overflow_allowed: boolean

See the description of the VOLUME_OVERFLOW_ALLOWED parameter of the REQUEST_MASS_STORAGE command.

status: VAR of ost$status

See the description of the STATUS parameter of the REQUEST_MASS_STORAGE command.

Conditions

pfe$bad_cycle_number
pfe$bad_cycle_option
pfe$bad_family_name
pfe$bad_local_file_name
pfe$bad_log_option
pfe$bad_master_catalog_name
pfe$bad_nth_subcatalog_name
pfe$bad_password
pfe$bad_permanent_file_name
pfe$bad_retention_period
pfe$catalog_full

pfe$cycle_overflow
pfe$cycle_underflow
pfe$duplicate_cycle
pfe$incorrect_password
pfe$lfn_in_use
pfe$name_already_subcatalog
pfe$path_too_short
pfe$pf_system_error
pfe$unknown_family
pfe$unknown_master_catalog
pfe$unknown_nth_subcatalog
pfe$usage_not_permitted
rme$file_class_not_valid
rme$improper_alloc_size_value
rme$improper_cycle_reference
rme$improper_est_file_size
rme$improper_file_class
rme$improper_recorded_vsn
rme$improper_vol_overflow
rme$job_not_privileged
rme$redundant_alloc_size_spec
rme$vsn_not_part_of_set

Remarks

- The RMP$REQUEST_MASS_STORAGE interface is used only in CYBIL programs.
- This request is optional because, by default, NOS/VE assigns files to the appropriate mass storage device classes.
- The file is attached with all modes of access and no sharing.
- This request is rejected if the file cycle is already registered in the specified catalog.
- You must issue a DETACH_FILE command or AMP$RETURN request to explicitly terminate the attachment of this file to the job. An FSP$CLOSE_FILE request does not detach a file whose attachment originated with this request.
- Users are each validated to have their files on a specific mass storage set. Each set consists of one or mass storage volumes that are candidates for assignment to this file. This request provides the following ways to select, from the list of candidate volumes, the initial volume to which the file is assigned.
  - If you assign the INITIAL_VOLUME parameter a value other than rmc$unspecified_vsn, the volume identified by the recorded VSN is selected.
  - If you assign the FILE_CLASS parameter a value other than rmc$unspecified_file_class, the file is assigned to a member of the designated class.
  - If you specify both the INITIAL_VOLUME and FILE_CLASS parameters, the specific volume is selected only if the mass storage volume is a member of the class specified.

  - If you select both rmc$unspecified_vsn and rmc$unspecified_file_class, the system selects a candidate volume that has the most available space.

- The ALLOCATION_SIZE and TRANSFER_SIZE parameters are not used to select the initial volume. Once the initial volume is selected, the allocation size and transfer size specifications are adjusted to reflect the size supported by the initial volume.

- If your site backs up files by volume, it may have decided to back up only volumes that belong to classes K and M; by default, these are the only volumes that contain permanent files. Bear this in mind when choosing to place a permanent file on a volume not belonging to either class K or class M.

- FSP$COPY_FILE does not copy the mass storage attributes of its input file when it creates the output file. Therefore, you must precede your call to FSP$COPY_FILE with a call to RMP$REQUEST_MASS_STORAGE.

- The identifier for this request is rmc$resource_management_id.

Examples

The RMP$REQUEST_MASS_STORAGE request is contained in the following procedure:

```
PROCEDURE [XREF] rmp$request_mass_storage (file: fst$file_reference;
      allocation_size: rmt$allocation_size;
      estimated_file_size: amt$file_byte_address;
      file_class: rmt$mass_storage_class;
      initial_volume: rmt$recorded_vsn;
      volume_overflow_allowed: boolean;
      VAR status: ost$status);

?? PUSH (LISTEXT := ON) ??
*copyc amt$file_byte_address
*copyc fst$file_reference
*copyc ost$status
*copyc rmc$condition_code_limits
*copyc rmc$unspecified_allocation_size
*copyc rmc$unspecified_file_class
*copyc rmc$unspecified_file_size
*copyc rmc$unspecified_vsn
*copyc rme$request_mass_storage
*copyc rmt$allocation_size
*copyc rmt$mass_storage_class
*copyc rmt$recorded_vsn
?? POP ??
```

Type Declarations

The following RMP$REQUEST_MASS_STORAGE declarations reside in library $SYSTEM.CYBIL.OSF$PROGRAM_INTERFACE. They reflect XDCL and #GATE attributes. These declarations can be used by any caller up to ring 13.

AMT$FILE_BYTE_ADDRESS

This declaration is documented in the CYBIL File Management manual.

FST$FILE_REFERENCE

This declaration is documented in the CYBIL File Management manual.

OST$STATUS

This declaration is documented in the CYBIL File Management manual.

RMC$CONDITION_CODE_LIMITS

```
CONST
  rmc$min_ecc = (($INTEGER ('R') * 100(16)) + $INTEGER ('M')) * 1000000(16),
  rmc$min_ecc_resource_management = rmc$min_ecc,
  rmc$max_ecc_resource_management = rmc$min_ecc_resource_management + 9999,

  rmc$resource_management_id = 'RM';
```

RMC$MASS_STORAGE_CLASS

The following are NOS/VE system conventions for classifying files and catalogs. A particular mass storage device may be associated with none, one, or more than one of these classes.

```
?? FMT (FORMAT) := OFF) ??
 CONST
    rmc$msc_system_swap_files       = 'C',
    rmc$msc_system_catalogs         = 'J',
    rmc$msc_system_permanent_files  = 'K',
    rmc$msc_user_catalogs           = 'L',
    rmc$msc_user_permanent_files    = 'M',
    rmc$msc_user_temporary_files    = 'N',
    rmc$msc_system_critical_files   = 'Q';
?? FMT (FORMAT) := ON) ??
```

RMC$MAX_ALLOCATION_SIZE

```
CONST
   rmc$max_allocation_size = 0ffffff(16);
```

RMC$UNSPECIFIED_ALLOCATION_SIZE

```
CONST
   rmc$unspecified_allocation_size = 0;
```

RMC$UNSPECIFIED_FILE_CLASS

```
CONST
   rmc$unspecified_file_class = 'A';
```

RMC$UNSPECIFIED_FILE_SIZE

```
CONST
   rmc$unspecified_file_size = 0;
```

RMC$UNSPECIFIED_FILE_VSN

```
CONST
   rmc$unspecified_vsn = '      ';
```

RMT$ALLOCATION_SIZE

```
  TYPE
     rmt$allocation_size = 0 .. rmc$max_allocation_size;

*copyc rmc$max_allocation_size
*copyc rmc$unspecified_allocation_size
```

RMT$MASS_STORAGE_CLASS

```
  TYPE
     rmt$mass_storage_class = char;

*copyc rmc$mass_storage_class
*copyc rmc$unspecified_file_class
```

RMT$RECORDED_VSN

This declaration is documented in the CYBIL File Management manual.

Supported Hardware Products E

Supported Hardware Products E

The Control Data products supported by NOS/VE are listed in table E-1. The individual products are described in the section called Device Descriptions. The PCU subcommand DEFINE_ELEMENT requires a serial number to identify an element. A product does not have a serial number. To identify a product by serial number, take the serial number from the equipment identification label of the equipment specified in table E-2.

Table E-1. Hardware Product Serial Numbers

Element Identification	Equipment Identification (Use Serial Number from This Equipment)	Description of Element
Attached Processors:		
$65354_10	AE128 A	MAP V without CM connection
$65354_11	AE128 A	MAP V for CYBER 835
$65354_12	AE128 A	MAP V for CYBER 855
Storage Units:		
$5682_12	$5682_12	18-track, 79 ips, 50/60 Hz
$5682_14	$5682_14	18-track, 79 ips, 50/60 Hz
$639_1	BY309 A	9-track, 25/75 ips, 60 Hz
$639_1	BY309 B	9-track, 25/75 ips, 50 Hz
$679_2	BW416 A	9-track, 100 ips, dual-mode kit
$679_3	BW415 A	9-track, 150 ips, dual-mode kit
$679_4	BW414 A	9-track, 200 ips, dual-mode kit
$679_5	BW413 A	9-track, 100 ips, GCR kit
$679_6	BW412 A	9-track, 150 ips, GCR kit
$679_7	BW411 A	9-track, 200 ips, GCR kit
$698_30	308VC OHI	9-track, 200 ips, 1600/6250 cpi, 60 Hz
$698_31	308WC OHI	9-track, 200 ips, 1600/6250 cpi, 50 Hz
$834_12	PA501 A	Disk storage unit
$836_110	PA506 A	Disk storage unit upgrade
$836_221	PA506 A	2x2 with cabinet
$836_441	PA506 A	4x4 with cabinet
$844_41	BR308 y	Disk storage unit (y=A,B,C or D)
$844_44	BR312 y	Disk storage unit (y=A,B,C or D)
$885_11	BZ703 y	Disk storage unit (y=A or B)
$885_12	BZ703 y	Disk storage unit (y=E or F)
$887_1	PA601 A	Disk storage unit, 50 Hz
$887_1	PA601 B	Disk storage unit, 60 Hz
$895_2	BZ640 C	Disk storage unit, 60 Hz
$895_2	BZ640 D	Disk storage unit, 50 Hz
$9639_1	BY3G7 y	Tape storage unit, 25/75 ips (y=A or B)
$9836_1	PA5R2 A	Disk storage subsystem
$9853_x	PA2A7 F	Disk storage subsystem, 60 Hz
$9853_x	PA2A7 G	Disk storage subsystem, 50 Hz

(Continued)

Table E-1. Hardware Product Serial Numbers *(Continued)*

Element Identification	Equipment Identification (Use Serial Number from This Equipment)	Description of Element
Channel Adapters:		
$2629_2	GK431 A	Integrated communications adapter (ICA)
Communication Elements:		
$380_170	FW201 y	Network access device (y=B or C)
$2620_210	GH120 B	MTI, 60 Hz
$2620_211	GH121 A	MTI, 50 Hz
$2621_210	GH120 B	MDI, 60 Hz
$2621_211	GH121 A	MDI, 50 Hz
$5380_100	BS231 A	STORNET
$7040_200	BU246 B	Extended semiconductor memory (ESM-II)
Controllers:		
$10395_11	FA7B4 B	834 disk controller
$FA7B4_D[1]	FA7B4 D	836 disk controller
$FA7B5_A[2]	FA7B5 A	9836 and 9853 disk controllers
$5680_11	$5680_11	50/60 Hz single-access cartridge tape controller (CYBER 170)
$5698_10	311VC OHI	208/230V two-port IPI interface
$5698_11	311VC OHI	380/415V two-port IPI interface
$5698_12	311VC OHI	220/240V two-port IPI interface
$698_10	310VC OHI	208/230V single-channel controller
$698_11	310WC OHI	380/415V single-channel controller
$698_12	310WC 1HI	220/240V single-channel controller
$698_20	310VC OHI	208/230V dual-channel controller
$698_21	310WC OHI	380/415V dual-channel controller
$698_22	310WC 1HI	220/240V dual-channel controller
$7021_31	FA450 y	ATS single-channel (y=A,B or C)
$7021_32	FA450 y	ATS dual-channel (y=A,B or C)
$7155_1	FA211 A	Fixed module drive (FMD) disk controller
$7155_11	FA211 B	Single-channel FMD controller

1. The $836_xxx product consists of both disk controllers and storage units. The product identification ($836_xxx) is used to describe the disk storage unit and the equipment number ($FA7B4_D) is used to describe the disk controller.

2. The $9836_x, $9853_1, $9853_2, and $9853_3 products consist of both disk controllers and storage units. The product identification ($9836_1 and $9853_x) is used to describe the disk storage unit and the equipment number ($FA7B5_A) is used to describe the disk controller.

(Continued)

Table E-1. Hardware Product Serial Numbers *(Continued)*

Element Identification	Equipment Identification (Use Serial Number from This Equipment)	Description of Element
$7155_12	FA211 B	2-channel FMD controller
$7155_13	FA211 B	3-channel FMD controller
$7155_14	FA211 B	Four-channel FMD controller
$7165_21	FA163 A	Single-ported 895 SCU, 60 Hz
$7165_21	FA163 B	Single-ported 895 SCU, 50 Hz
$7165_22	FA163 C	Dual-ported 895 SCU, 60 Hz
$7165_22	FA163 D	Dual-ported 895 SCU, 50 Hz
$7221_1	FA165 A	639-1 tape controller
$7221_11	FA168 A	9639-1 tape controller

NOS EST Entries for Peripherals

Table E-2 shows whether a peripheral element may or must be specified in the equipment status table (EST) of the NOS EQPDECK in order to be used by NOS/VE.

Table E-2. NOS EQPDECK Entries for Peripherals

Storage Unit	May Be in NOS EQPDECK	Must Be in NOS EQPDECK	Ownership May Be Toggled[1]	Concurrent Use Is Permitted
$10395_11	N[2,3]	N	N	N
$262x_xxx	N[2]	N	N	N
$380_170	N[2]	N	N	N
$5380_100	N[2]	N	N	N
$5680_11	N[2]	N	N	N
$5682_1x	N	N	N	N
$5698_1x	N	N	N	N
$639_1	Y	N[4]	Y	N
$679_1	Y	Y[4]	Y	N
$698_1x/2x	Y	Y[4]	Y	N
$7021_31	Y	Y[5]	Y	N
$7021_32	Y	Y[5]	Y	Y[6]
$7040_200	N[2]	N	N	N
$7155_1x	Y	Y[5]	N	Y[6]
$7165_2x	N[2]	N	N	Y[6]
$7221_1	Y[5]	N[2]	Y	N
$7221_11	N[2]	N	N	N
$834_12	N[2]	N	N	N
$836_xxx	N[2]	N	N	N
$844_4x	Y	Y[4]	N	N
$885_1x	Y	Y[4,7]	N	N

1. Use the LCU subcommand CHANGE_ELEMENT_STATE to toggle device ownership.

2. In the NOS configuration, the channel to be used by NOS/VE must be in either the ON or the DOWN state and must not have any elements connected to it.

3. If the element is connected to two channels from the same mainframe, both channels must be either in the ON or DOWN state in the NOS/VE physical configuration (that is, NOS cannot be allowed to use either channel concurrently with NOS/VE use).

4. The NOS/VE channel, controller, and storage unit must be in the DOWN state in the NOS configuration.

5. The NOS/VE channel to the element must be in the DOWN state in the NOS configuration.

6. The controller may be accessed by more than one host data channel. Each state of a dual-state system must have its own channel to the controller. In the case of a $7165_2x subsystem, this implies that each state must have its own CYBER channel coupler (CCC).

7. There must be a REMOVE entry for the storage unit's equipment entry.

(Continued)

Table E-2. NOS EQPDECK Entries for Peripherals *(Continued)*

Storage Unit	May Be in NOS EQPDECK	Must Be in NOS EQPDECK	Ownership May Be Toggled[1]	Concurrent Use Is Permitted
$887_1	N[2]	N	N	N
$895_2	N[2]	N	N	N
$9639_1	N	N	N	N
$9836_1	N	N	N	N
$9853_x	N	N	N	N
$FA7B4_D	N[2,3]	N	N	N
$FA7B5_D	N	N	N	N

1. Use the LCU subcommand CHANGE_ELEMENT_STATE to toggle device ownership.

2. In the NOS configuration, the channel to be used by NOS/VE must be in either the ON or the DOWN state and must not have any elements connected to it.

3. If the element is connected to two channels from the same mainframe, both channels must be either in the ON or DOWN state in the NOS/VE physical configuration (that is, NOS cannot be allowed to use either channel concurrently with NOS/VE use).

NOS/BE EST Entries for Peripherals

Table E-3 shows whether a peripheral element may or must be specified in the equipment status table (EST) of the NOS/BE CMR in order to be used by NOS/VE.

Table E-3. NOS/BE CMR Entries for Peripherals

Storage Unit	May Be in NOS/BE CMR	Must Be in NOS/BE CMR	Ownership May Be Toggled[1]	Concurrent Use Is Permitted
$10395_11	N[2,3]	N	N	N
$262x_xxx	N[2]	N	N	N
$380_170	N[2]	N	N	N
$5380_100	N[2]	N	N	N
$5680_11	N[2]	N	N	N
$5682_1x	N	N	N	N
$5698_1x	N	N	N	N
$639_1	N	N[2]	N	N
$679_1	Y	Y[4]	Y	N
$698_1x/2x	N	N	N	N
$7021_31	Y	Y[5]	Y	N
$7021_32	Y	Y[5]	Y	Y[6]
$7040_200	N[2]	N	N	N
$7155_1x	Y	Y[5]	N	N
$7165_2x	N[2]	N	N	N
$7221_1	Y[2]	N	N	N
$7221_11	N	N	N	N
$834_12	N[2]	N	N	N
$836_xxx	N[2]	N	N	N

1. Use the LCU subcommand CHANGE_ELEMENT_STATE to toggle device ownership.

2. In the NOS/BE configuration, the channel to be used by NOS/VE must be in either the ON or the DOWN state and must not have any elements connected to it.

3. If the element is connected to two channels from the same mainframe, both channels must be either in the ON or DOWN state in the NOS/VE physical configuration (that is, NOS/BE cannot be allowed to use either channel concurrently with NOS/VE use).

4. The NOS/VE channel, controller, and/or storage unit must all be in the DOWN state in the NOS/BE configuration.

5. The NOS/VE channel to the element must be in the DOWN state in the NOS/BE configuration.

6. The controller may be accessed by more than one host data channel. Each state of a dual-state system must have its own channel to the controller. In the case of a $7165_2x subsystem, this implies that each state must have its own CYBER channel coupler (CCC).

(Continued)

Table E-3. NOS/BE CMR Entries for Peripherals *(Continued)*

Storage Unit	May Be in NOS/BE CMR	Must Be in NOS/BE CMR	Ownership May Be Toggled[1]	Concurrent Use Is Permitted
$844_4x	Y	Y[2]	N	N
$885_1x	Y	Y[2]	N	N
$887_1	N[3]	N	N	N
$895_2	N[3]	N	N	N
$9639_1	N	N	N	N
$9836_1	N	N	N	N
$9853_x	N	N	N	N
$FA7B4_D	N[3,4]	N	N	N
$FA7B5_D	N	N	N	N

1. Use the LCU subcommand CHANGE_ELEMENT_STATE to toggle device ownership.

2. The NOS/VE channel, controller, and/or storage unit must all be in the DOWN state in the NOS/BE configuration.

3. In the NOS/BE configuration, the channel to be used by NOS/VE must be in either the UP or the DOWN state and must not have any elements connected to it.

4. If the element is connected to two channels from the same mainframe, both channels must be either in the ON or DOWN state in the NOS/VE physical configuration (that is, NOS/BE cannot be allowed to use either channel concurrently with NOS/VE use).

Characteristics of CYBER 180 IOUs

Tables E-4 and E-5 list the channel characteristics and supported mainframes for CYBER 180 input/output units (IOUs).

Table E-4. Characteristics of CYBER 180 IOUs (Part 1)

	I0	I1	I1/CR	I2
Peripheral Processors:[1]				
PP memory size (NIO)[2]	16K	4K	4K	4K
PP memory size (CIO)[2]	NA	NA	NA	NA
PP speed (relative)	≥4X	2X	2X	4X
Max PPs	10	20	20	20
Min PPs	5	10	10	10
Max PPs (NIO)	10	20	20	20
Min PPs (NIO)	5	10	10	10
Max PPs (CIO)	NA	NA	NA	NA
Min PPs (CIO)	NA	NA	NA	NA
Supports enhanced R register	YES	NO	NO	
Supports HOLD instruction	YES	NO	NO	
Channels:				
Max channels	12	24	24	24
Min channels	6	12	8	12
Max channels (NIO)	12	24	24	24
Min channels (NIO)	6	12	8	12
Max channels (CIO)	NA	NA	NA	NA
Min channels (CIO)	NA	NA	NA	NA
Max 170 channels (NIO)	0/0/2[3,4]	24/22	18/12/10	24
Min 170 channels (NIO)	0/0/0	12	18/6/4	12
Max 170 channels (CIO)	NA	NA	NA	NA
Min 170 channels (CIO)	NA	NA	NA	NA
Max ICI channels (NIO)	6/4/4[4]	0/2	6/4/6	NA
Min ICI channels (NIO)	3	0	6/2/4	NA
Max ISI channels (CIO)/ports per channel	NA	NA	NA	
Min ISI channels (CIO)	NA	NA	NA	NA
Max IPI channels (NIO)	6/8/6[4]	NA	NA	NA
Min IPI channels (NIO)	3	NA	NA	NA
Max IPI channels (CIO)/ports per channel	NA	NA	NA	NA
Min IPI channels (CIO)	NA	NA	NA	NA

1. In a configuration involving CIO channels, both PPs and channels must be from the same cluster. Problems can arise when downing a PP or when adding equipment to a free channel when no PPs are free.

2. 16-bit bytes.

3. A CYBER 170 channel cannot be configured on a CYBER 930 with only five PPs.

4. A total of eight channels can be actively configured by NOS/VE on a CYBER 930 that has ten PPs. An actively configured channel has a PP assigned to it. Channels not actively configured can be physically configured for redundancy.

(Continued)

Table E-4. Characteristics of CYBER 180 IOUs (Part 1) *(Continued)*

	I0	I1	I1/CR	I2
Performance (10^6 bytes/sec)				
PP/CM block transfer rate	8	4	4	8
DMA chan/CM transfer rate[1]	20	NA	NA	NA
Supported Mainframes:				
810/810A/830/830A	NO	NO	YES	NO
815/825	NO	YES	NO	NO
835	NO	NO	NO	YES
845/855	NO	NO	NO	YES
840/850/860	NO	NO	NO	YES
840A/850A/860A/870A	NO	NO	NO	NO
930-11/930-31	YES	NO	NO	NO
932-11/932-31/932-32	YES	NO	NO	NO
960-11/960-31/960-32	NO	NO	NO	NO
962-11/962-31/962-32	NO	NO	NO	NO
992-31/992-32	NO	NO	NO	NO
994-31/994-32	NO	NO	NO	NO
990/990E/995E	NO	NO	NO	NO

1. Theoretical rate; in practice, constrained by controller speed.

Table E-5. Characteristics of CYBER 180 IOUs (Part 2)

	I4/ I4A/ I4AC	Dual[1] I4/ I4A	I4C	Dual[1] I4C/ I4C
Peripheral Processors:[2]				
PP memory size (NIO)[3]	4K/8K	4K/8K	4K/8K	4K/8K
PP memory size (CIO)[3]	8K	8K	8K	8K
PP speed (relative)	4X	4X	4X	4X
Max PPs	30	30	30	30
Min PPs	20	20	20	20
Max PPs (NIO)	20	20	20	20
Min PPs (NIO)	20	20	20	20
Max PPs (CIO)	10	10	10	10
Min PPs (CIO)	5	5	5	5
Supports enhanced R register	NO	NO	NO	NO
Supports HOLD instruction	NO	NO	NO	NO
Channels:				
Max channels	34	34	34	34
Min channels	2	24	24	24
Max channels (NIO)	24	24	24	24
Min channels (NIO)	24	24	24	24
Max channels (CIO)	10	10	10	10
Min channels (CIO)	5	5	5	5
Max 170 channels (NIO)	24	24	24	24
Min 170 channels (NIO)	24	24	24	24
Max 170 channels (CIO)	10	10	10	10
Min 170 channels (CIO)	0	0	0	0
Max ICI channels (NIO)	NA	NA	NA	NA
Min ICI channels (NIO)	NA	NA	NA	NA
Max ISI channels (CIO)/ports per channel	10/2	10/2	18/2	18/2
Min ISI channels (CIO)	0	0	0	0
Max IPI channels (NIO)	NA	NA	NA	NA
Min IPI channels (NIO)	NA	NA	NA	NA
Max IPI channels (CIO)/ports per channel	10/2	10/2	10/2	10/2
Min IPI channels (CIO)	0	0	0	0
Performance (10^6 bytes/sec)				
PP/CM block transfer rate	8	8	8	8
DMA chan/CM transfer rate[4]	16	16	16	16

1. Number per IOU.

2. In a configuration involving CIO channels, both PPs and channels must be from the same cluster. Problems can arise when downing a PP or when adding equipment to a free channel when no PPs are free.

3. 16-bit bytes.

4. Theoretical rate; in practice, constrained by controller speed.

(Continued)

Table E-5. Characteristics of CYBER 180 IOUs (Part 2) *(Continued)*

	I4/ I4A/ I4AC	Dual[1] I4/ I4A	I4C	Dual[1] I4C/ I4C
Supported Mainframes:				
810/810A/830/830A	NO	NO	NO	NO
815/825	NO	NO	NO	NO
835	NO	NO	NO	NO
845/855	YES	NO	NO	NO
840/850/860	YES	YES	NO	NO
840A/850A/860A/870A	YES	YES	NO	NO
930-11/930-31	NO	NO	NO	NO
932-11/932-31/932-32	NO	NO	NO	NO
960-11/960-31/960-32	YES	YES	NO	NO
962-11/962-31/962-32	NO	NO	YES	YES
992-31/992-32	NO	NO	YES	YES
994-31/994-32	YES	YES	NO	NO
990/990E/995E	YES	YES	NO	NO

1. Number per IOU.

Characteristics of NOS/VE Storage Units

This and the next several sections supply information about NOS/VE disk units, tape units, communication devices, and extended memory devices. In addition to describing unit characteristics, logical constructs of disk units, and disk unit initialization, these sections describe alternate and redundant path configurations and give recommendations about how to handle volume defects.

Tables E-6 and E-7 list the characteristics of NOS/VE supported disk units.

Table E-6. Disk Unit Characteristics - 834, 836, 844, and 885

Characteristic	834-12	836-xxx	844-4x	885-1x
Physical characteristics:				
Channel protocol	ISI	ISI	C170	C170
Number of channel accesses	2	2	1 to 4	1 to 4
Number of drive accesses	1	1	2 or 4	1 or 2
Spindles/cabinet	2 to 4	2 to 4	1	2
Cylinders/spindle	817	701	823	843
Tracks/cylinder	10	24	19	40
Sectors/track	32	47	24	8
Physical capacity:[1]				
10^6 bytes/cabinet (max)	535.43	1619.41	181.26	1113.57
10^6 bytes/spindle	133.86	404.85	181.26	556.78
Bytes/cylinder	163840	577536	220248	660480
Bytes/track	16384	24064	11592	16512
Bytes/sector	512	512	483	2064
Wasted capacity:[2]				
10^6 bytes/spindle	0	3	33	4
Gross capacity:[3]				
10^6 bytes/spindle	133.86	401.98	148.32	552.47
10^6 bytes/cylinder	163840	573440	180224	655360

1. Physical capacity refers to the remaining capacity of a volume that has been formatted for the NOS/VE sector size, but has not yet been initialized.

2. Wasted capacity occurs if the cylinder capacity is not a multiple of the default allocation size (16,384 bytes) or the page size is not a multiple of the sector size.

3. Gross capacity may be either the same as, or slightly less than, physical capacity. Where gross capacity is less than physical capacity, the difference may be caused by either or both of the following reasons: 1) the cylinder capacity is not an integral number of DAUs, or 2) the number of bytes per sector is not a power of 2 multiple. Gross capacity is obtained by subtracting the wasted capacity from the physical capacity.

(Continued)

Table E-6. Disk Unit Characteristics - 834, 836, 844, and 885 *(Continued)*

Characteristic	834-12	836-xxx	844-4x	885-1x
Net capacity:[1]				
Net formatted capacity	99.15%	98.46%	81.07%	98.46%
10^6 net bytes/cabinet	530.88	1594.52	146.95	1096.48
10^6 net bytes/spindle	132.72	398.63	146.95	548.24
Reserved capacity:				
Total DAUs overhead/spindle	69	205	335	1033
–DAUs for reserved cylinders	20	70	132	320
–DAUs for device allocation table	5	13	19	67
–DAUs for device file list	29	85	125	462
–DAUs for device log	13	35	57	182
Number of reserved cylinders	2	2	3	2
–Deadstart cylinder number	816	700	822	841
–Maintenance cylinder numbers	815	699	820,821	842
–Confidence test cylinder number	815	699	821	842
File allocation details:				
Total DAUs/spindle (HDA)	8170	24535	36212	134880
–Bytes/DAU	16384	16384	4096	4096
–Sectors/DAU	32	32	10	2
Default AUs/cabinet	32680	98140	18106	67440
Default AUs/spindle (HDA)	8170	24535	9053	33720
–Bytes/default AU	16384	16384	16384	16384
–Default AUs/cylinder	10	35	11	40
–Default AUs/track	1.00	1.47	0.6	1.00
Max files of min. size/HDA[2]	8121	24400	8850	33007
Performance characteristics of disk units:				
Supports dynamic dual access	No	No	Yes	Yes
Average latency (ms)	8.33	8.33	8.33	8.33
Random seek time (ms)	28.5	20	28.4	25
Disk speed (Mbyte/s)	1.22	1.82	0.806	1.1975
Performance characteristics of controller:				
Controller latency/command (ms)	7.2	7.2	0.1	0.1
Controller transfer rate (Mbyte/s)	2	2	3	3
One cylinder burst rate (Mbyte/s)	0.983	1.431	0.569	0.983
Controller buffer memory size	16384	16384	2064	2064
Buffer size/track size	1.00	0.68	0.21	0.13

1. Net capacity refers to the remaining capacity available to users after a disk volume has been initialized. Net capacity is obtained by subtracting the reserved capacity from the gross capacity.

2. These values apply to sites that initialize their volumes at NOS/VE Version 1.3.1 or later. For sites that initialize their volumes prior to NOS/VE Version 1.3.1, the following values apply: 4000 (834-12), 12200 (836-xxx), 4600 (844-4x), and 16800 (885-1x).

(Continued)

Table E-6. Disk Unit Characteristics - 834, 836, 844, and 885 *(Continued)*

Characteristic	834-12	836-xxx	844-4x	885-1x
Performance characteristics of disk subsystem:[1]				
Bytes real memory used/spindle	6163	7534	6350	9513
Bytes/MAU	2048	2048	2048	2048
Sectors/4K page	8	8	10	2
Sectors/65K job swap	520	520	650	130
Revolutions/page fault	0.250	0.170	0.417	0.250
Revolutions/job swap	16.250	11.064	27.083	16.250
Page faults/second[2]	20.7	26.1	22.8	26.6
Job swaps/second[2]	2.8	3.9	1.9	3.0
10^6 bytes/second (paging)	0.08	0.11	0.09	0.11
10^6 bytes/second (swapping)	0.74	1.04	0.51	0.79
Page fault ratio to 885	0.78	0.98	0.86	1.00
Job swap ratio to 885	0.94	1.32	0.64	1.00
Effect of allocation size upon sequential file transfer rate:[3]				
10^6 bytes/second with allocation size:				
16,384 bytes	0.27	0.35	0.25	0.33
32,768 bytes	0.42	0.56	0.35	0.49
65,536 bytes	0.59	0.81	0.44	0.65
131,072 bytes	0.74	1.04	0.51	0.79
262,144 bytes	NA	1.21	NA	0.87
524,288 bytes	NA	1.32	NA	0.93
1 cylinder sustained rate	0.78	1.32	0.51	0.94

1. The information on disk subsystem performance is based on the following assumptions: random I/O activity, one channel, one controller, and one spindle; no NOS/VE overhead; no NOS/VE optimizations.

2. The formulae for calculating the page fault and job swap rate are as follows:

```
If:
        1000=number of milliseconds per second
        C=controller overhead per command
        L=average drive latency (1/2 revolution)
        N=number of 128K (32 page) i/o requests per swap
        P=number of sectors per page
        R=number of sectors per track (1 revolution)
        S=random seek time of drive
        W=number of sectors per 65 page job swap

Then:

        page faults/sec= 1000/(C+L+S+((P/R)*2*L))
        job swaps/sec = 1000/((N*(C+L+S))+((W/R)*2*L))
```

3. These values assume a file size of 1,048,576 bytes.

(Continued)

Table E-6. Disk Unit Characteristics - 834, 836, 844, and 885 *(Continued)*

Characteristic	834-12	836-xxx	844-4x	885-1x
Percentage of physical cylinder capacity wasted for each supported allocation size:				
16,384 bytes	0.00%	0.71%	3.29%	0.78%
32,768 bytes	0.00%	3.55%	10.73%	0.78%
65,536 bytes	20.00%	9.22%	10.73%	0.78%
13,1072 bytes	20.00%	9.22%	40.49%	0.78%
26,2144 bytes	NA	9.22%	NA	20.62%
52,4288 bytes	NA	9.22%	NA	20.62%
Cylinder	0.00%	0.00%	0.00%	0.00%
Percent of 1 cylinder sustained rate with allocation size:				
16,384 bytes	34.71%	26.54%	49.73%	34.93%
32,768 bytes	54.46%	42.73%	69.56%	52.42%
65,536 bytes	76.13%	61.50%	86.68%	69.93%
131,072 bytes	95.03%	78.81%	99.24%	83.96%
262,144 bytes	NA	91.71%	NA	93.31%
524,288 bytes	NA	99.89%	NA	98.82%
Effect of page size upon random access transfer rate:[1]				
10^6 bytes/second with page size:				
2,048 bytes	0.04	0.06	0.05	0.06
4,096 bytes	0.08	0.11	0.09	0.11
8,192 bytes	0.16	0.20	0.16	0.20
16,384 bytes	0.27	0.35	0.25	0.33
32,768 bytes	0.42	0.56	0.35	0.49
65,536 bytes	0.59	0.81	0.44	0.65
Effect of page size upon a fixed 4096-byte transfer requirement:				
10^6 bytes/second with page size:				
2,048 bytes	0.04	0.06	0.05	0.06
4,096 bytes	0.08	0.11	0.09	0.11
8,192 bytes	0.08	0.10	0.08	0.10
16,384 bytes	0.07	0.09	0.06	0.08
32,768 bytes	0.05	0.07	0.04	0.06
65,536 bytes	0.04	0.05	0.03	0.04

1. This analyzes the additional overhead required to transfer more or less data than the 4,096 bytes needed.

(Continued)

Table E-6. Disk Unit Characteristics - 834, 836, 844, and 885 *(Continued)*

Characteristic	834-12	836-xxx	844-4x	885-1x
Connectivity:				
Controller	10395-11	FA7B4D	7155-1x	7155-1x
Channel protocol	ISI	ISI	170	170
Requires DMA	No	No	No	No
PPs/channel	1	1	1	1
Adapters/channel	1	1	NA	NA
Ports/channel	1	1	1	1
Controllers/channel	8	8	1	1
Controllers/channel port	NA	NA	NA	NA
Spindles/controller	4	1	8	16
Spindles/channel	8	8	8	16
IOU connections:[1]				
I0 - NIO/IPI/DMA	NA	NA	NA	NA
I1 - NIO/170	NA	NA	1.2.1	1.2.1
I1 - NIO/ICI	1.2.1	1.2.1	NA	NA
I2 - NIO/170	NA	NA	1.2.1	1.2.1
I4 - CIO/170	NA	NA	1.4.1	1.4.1
I4/I4A/I4AC - CIO/170/DMA	NA	NA	NA	NA
I4/I4A/I4AC - CIO/IPI/DMA	NA	NA	NA	NA
I4/I4A/I4AC - CIO/ISI/DMA	NA	NA	NA	NA
I4/I4A/I4AC - NIO/170	NA	NA	1.2.1	1.2.1
I4C - CIO/170	NA	NA	1.4.1	1.4.1
I4C - CIO/170/DMA	NA	NA	NA	NA
I4C - CIO/IPI/DMA	NA	NA	NA	NA
I4C - CIO/ISI/DMA	NA	NA	NA	NA

1. This category indicates the NOS/VE version under which the specified IOU connection first became available.

Table E-7. Disk Unit Characteristics - 887, 895, 9836, and 9853

Characteristic	887-1	895-2	9836-1	9853-1
Physical characteristics:				
Channel protocol	ISI	C170	IPI	IPI
Number of channel accesses	2	2 or 4	2	2
Number of drive accesses	NA	2	2	2
Spindles/cabinet	2	4	2 to 4	2 to 4
Cylinders/spindle	884	886	703	1412
Tracks/cylinder	4	15	24	19
Sectors/track	38	10	12	21
Physical capacity:[1]				
10^6 bytes/cabinet (max)	1100.74	2194.44	1658.59	4615.27
10^6 bytes/spindle	550.37	548.61	414.65	1153.82
Bytes/cylinder	622592	619200	589824	817152
Bytes/track	155648	41280	24576	43008
Bytes/sector	4096	4128	2048	2048
Wasted capacity:[2]				
10^6 bytes/spindle	0	12	0	20
Gross capacity:[3]				
10^6 bytes/spindle	550.37	537.10	414.65	1133.58
10^6 bytes/cylinder	622592	606208	589824	802816
Net capacity:[4]				
Net formatted capacity	99.12%	97.04%	99.03%	97.53%
10^6 bytes/cabinet	1091.08	2129.49	1642.47	4501.39
10^6 bytes/spindle	545.54	532.37	410.62	1125.35

1. Physical capacity refers to the remaining capacity of a volume that has been formatted for the NOS/VE sector size, but has not yet been initialized.

2. Wasted capacity occurs if the cylinder capacity is not a multiple of the default allocation size (16,384 bytes) or the page size is not a multiple of the sector size.

3. Gross capacity may be either the same as, or slightly less than, physical capacity. Where gross capacity is less than physical capacity, the difference may be caused by either or both of the following reasons: 1) the cylinder capacity is not an integral number of DAUs, or 2) the number of bytes per sector is not a power of 2 multiple. Gross capacity is obtained by subtracting the wasted capacity from the physical capacity.

4. Net capacity refers to the remaining capacity available to users after a disk volume has been initialized. Net capacity is obtained by subtracting the reserved capacity from the gross capacity.

(Continued)

Table E-7. Disk Unit Characteristics - 887, 895, 9836, and 9853 *(Continued)*

Characteristic	887-1	895-2	9836-1	9853-1
Reserved capacity:				
Total DAUs overhead/spindle	295	289	246	502
–DAUs for reserved cylinders	114	111	108	147
–DAUs for device allocation table	17	17	13	35
–DAUs for device file list	116	113	88	225
–DAUs for device log	46	46	35	93
Number of reserved cylinders	3	3	3	3
–Deadstart cylinder number	NA	885	702	1411
–Maintenance cylinder numbers	882,883	883,885	700	1409
–Confidence test cylinder number	881	884	701	1410
File allocation details:				
Total DAUs/spindle(HDA)	33592	32782	25308	69188
–Bytes/DAU	16384	16384	16384	16384
–Sectors/DAU	4	4	8	8
Default AUs/cabinet	67184	131128	101232	262140
Default AUs/spindle (HDA)	33592	32782	25308	69188
–Bytes/default AU	16384	16384	16384	16384
–Default AUs/cylinder	38	37	36	49
–Default AUs/track	9.5	2.50	1.5	2.63
Max files of min. size/HDA[1]	33411	32604	25170	65180
Performance characteristics of disk units:				
Supports dynamic dual access	Yes	Yes	No	No
Average latency (ms)	8.33	8.33	8.33	8.33
Random seek time (ms)	16	16	20	16
Disk speed (Mbyte/s)	12	3	1.82	3
Performance characteristics of controller:				
Controller latency/cmd (ms)	3.3	4	3.6	3.2
Controller transfer rate (Mbyte/s)	12	3	10	10
One cylinder burst rate (Mbyte/s)	9.320	2.420	1.472	2.530
Controller buffer memory size	65536	4128	98304	98304
Buffer size/track size	0.42	0.10	4.00	2.29

1. These values apply to sites that initialize their volumes at NOS/VE Version 1.3.1 or later. For sites that initialize their volumes at NOS/VE Version 1.2.2 or earlier, the following values apply: 16800 (887-1), 16300 (895-2), 12600 (9836-1), and 34500 (9853-1).

(Continued)

Table E-7. Disk Unit Characteristics - 887, 895, 9836, and 9853 *(Continued)*

Characteristic	887-1	895-2	9836-1	9853-1
Performance characteristics of disk subsystem:[1]				
Bytes real memory used/spindle	9743	9709	7621	17514
Bytes/MAU	4096	4096	2048	2048
Sectors/4K page	1	1	2	2
Sectors/65K job swap	65	65	130	130
Revolutions/page fault	0.026	0.100	0.167	0.095
Revolutions/job swap	1.711	6.50	10.833	6.190
Page faults/second[2]	35.7	33.3	28.8	34.3
Job swaps/second[2]	12.0	6.1	4.1	6.3
10^6 bytes/second (paging)	0.15	0.14	0.12	0.14
10^6 bytes/second (swapping)	3.17	1.61	1.09	1.68
Page fault ratio to 885	1.34	1.25	1.08	1.29
Job swap ratio to 885	4.04	2.05	1.38	2.13
Effect of allocation size upon sequential file transfer rate:[3]				
10^6 bytes/second with allocation size:				
16,384 bytes	0.56	0.47	0.38	0.48
32,768 bytes	1.05	0.79	0.61	0.81
65,536 bytes	1.89	1.19	0.86	1.24
131,072 bytes	3.15	1.61	1.09	1.67
262,144 bytes	4.71	1.94	1.25	2.03
524,288 bytes	6.26	2.17	1.35	2.27
1 cylinder sustained rate	6.61	2.18	1.37	2.33

1. The information on disk subsystem performance is based on the following assumptions: 1) random I/O activity 2) one channel, one controller, and one spindle 3) no NOS/VE overhead 4) no NOS/VE optimizations.

2. The formulae for calculating the page fault and job swap rate are as follows:

```
If:
        1000=number of milliseconds per second
        C=controller overhead per command
        L=average drive latency (1/2 revolution)
        N=number of 128K (32 page) i/o requests per swap
        P=number of sectors per page
        R=number of sectors per track (1 revolution)
        S=random seek time of drive
        W=number of sectors per 65 page job swap

Then:

        page faults/sec= 1000/(C+L+S+((P/R)*2*L))
        job swaps/sec = 1000/((N*(C+L+S))+((W/R)*2*L))
```

3. These values assume a file size of 1,048,576 bytes.

(Continued)

Table E-7. Disk Unit Characteristics - 887, 895, 9836, and 9853 *(Continued)*

Characteristic	887-1	895-2	9836-1	9853-1
Percentage of physical cylinder capacity wasted for each supported allocation size:				
16,384 bytes	0.00%	2.10%	0.00%	1.75%
32,768 bytes	0.00%	4.74%	0.00%	3.76%
65,536 bytes	5.26%	4.74%	0.00%	3.76%
13,1072 bytes	15.79%	15.33%	11.11%	3.76%
26,2144 bytes	15.79%	15.33%	11.11%	3.76%
52,4288 bytes	15.79%	15.33%	11.11%	35.84%
Cylinder	0.00%	0.00%	0.00%	0.00%
Percent of 1 cylinder sustained rate with allocation size:				
16,384 bytes	8.45%	21.49%	27.87%	20.73%
32,768 bytes	15.95%	35.10%	44.30%	34.91%
65,536 bytes	28.67%	54.70%	62.83%	53.08%
131,072 bytes	47.67%	73.68%	79.44%	71.74%
262,144 bytes	71.30%	89.15%	91.54%	87.04%
524,288 bytes	94.80%	99.61%	99.08%	97.43%
Effect of page size upon random access transfer rate:[1]				
10^6 bytes/second with page size:				
2,048 bytes	NA	NA	0.06	0.07
4,096 bytes	0.15	0.14	0.12	0.14
8,192 bytes	0.29	0.26	0.22	0.27
16,384 bytes	0.56	0.47	0.38	0.48
32,768 bytes	1.05	0.79	.61	0.81
65,536 bytes	1.89	1.19	.86	1.24
Effect of page size upon a fixed 4096-byte transfer requirement:				
10^6 bytes/second with page size:				
2,048 bytes	NA	NA	0.06	0.07
4,096 bytes	0.15	0.14	0.12	0.14
8,192 bytes	0.14	0.13	0.11	0.13
16,384 bytes	0.14	0.12	0.10	0.12
32,768 bytes	0.13	0.10	.08	0.10
65,536 bytes	0.12	0.07	.05	0.08

1. This analyzes the additional overhead required to transfer more or less data than the 4,096 bytes needed.

(Continued)

Table E-7. Disk Unit Characteristics - 887, 895, 9836, and 9853 *(Continued)*

Characteristic	887-1	895-2	9836-1	9853-1
Connectivity:				
Controller	NA	7165-2x	FA7B5A	FA7B5A
Channel protocol	ISI	170	IPI	IPI
Requires DMA	Yes	No	Yes	Yes
PPs/channel	1	2	1	1
Adapters/channel	NA	NA	NA	NA
Ports/channel	2	1	1	2
Controllers/channel	16	2	8	16
Controllers/channel port	8	NA	NA	8
Spindles/controller	1	32	2	8
Spindles/channel	16	64	16	64
IOU connections:[1]				
I0 - NIO/IPI/DMA	NA	NA	1.2.2	1.2.3
I1 - NIO/170	NA	NA	NA	NA
I1 - NIO/ICI	NA	NA	NA	NA
I2 - NIO/170	NA	1.2.1	NA	NA
I4 - CIO/170	NA	1.2.2	NA	NA
I4/I4A/I4AC - CIO/170/DMA	NA	1.5.1	NA	NA
I4/I4A/I4AC - CIO/IPI/DMA	NA	NA	NA	1.2.3
I4/I4A/I4AC - CIO/ISI/DMA	1.2.1	NA	NA	NA
I4/I4A/I4AC - NIO/170	NA	1.2.1	NA	NA
I4C - CIO/170	NA	1.4.1	NA	NA
I4C - CIO/170/DMA	NA	1.5.1	NA	NA
I4C - CIO/IPI/DMA	NA	NA	NA	1.4.1
I4C - CIO/ISI/DMA	1.4.1	NA	NA	NA

1. This category indicates the NOS/VE version under which the specified IOU connection first became available.

Tables E-8 and E-9 list the connectivity characteristics of tape subsystems. The table is organized according to tape controllers.

Table E-8. Tape Subsystem Connectivity - 698-xx, 7021-3x, and 7221-1

Tape Controller	698-1x 698-2x	7021-3x	7221-1
Tape unit	$698-3x	679-x	639-1
Channel protocol	170	170	ICI
Requires DMA	No	No	No
PPs/channel	1 or 2	1 or 2	1 or 2
Adapters/channel	NA	NA	1
Controllers/channel	1	1	NA
Drives/controller	8	8	1
Drives/channel	8	8	1
IOU connections:[1]			
I0 - NIO/ICI/DMA	NA	NA	NA
I0 - NIO/IPI/DMA	NA	NA	NA
I1 - NIO/170	1.2.3	1.2.1	NA
I1 - NIO/ICI	NA	NA	1.2.1
I2 - NIO/170	1.2.3	1.2.1	NA
I4/I4A/I4C - CIO/170	1.4.1	1.4.1	NA
I4/I4A/I4C - NIO/170	1.2.3	1.2.1	NA
I4/I4A/I4C - CIO/IPI/DMA	NA	NA	NA
I4C - CIO/170	1.4.1	1.4.1	NA
I4C - CIO/IPI/DMA	NA	NA	NA

1. This category indicates the NOS/VE version under which the specified IOU connection first became available.

Table E-9. Tape Subsystem Connectivity - 7221-11, 5698-1x, and 5680-11

Tape Controller	7221-11	5698-1x	5680-11
Tape unit	9639-1	698-3x	5682-1x
Channel protocol	ICI	IPI	170
Requires DMA	Yes	Yes	No
PPs/channel	1	1	2
Adapters/channel	1	NA	NA
Controllers/channel	NA	8	1
Drives/controller	2	8	8
Drives/channel	2	64[2]	8
IOU connections:[1]			
I0 - NIO/ICI/DMA	1.3.1	NA	NA
I0 - NIO/IPI/DMA	NA	1.4.1	NA
I1 - NIO/170	NA	NA	NA
I1 - NIO/ICI	NA	NA	NA
I2 - NIO/170	NA	NA	1.5.1
I4/I4A/I4AC - CIO/170	NA	NA	1.5.1
I4/I4A/I4AC - NIO/170	NA	NA	1.5.1
I4/I4A/I4AC - CIO/IPI/DMA	NA	1.4.1	NA
I4C - CIO/170	NA	NA	1.5.1
I4C - CIO/IPI/DMA	NA	1.4.1	NA

1. This category indicates the NOS/VE version under which the specified IOU connection first became available.

2. A 5698-1x controller supports up to 128 drives per channel if both ports of an I4 or I4C IPI channel are configured.

Tables E-10 and E-11 list the connectivity characteristics of various communication elements and external memory devices.

Table E-10. MTI, MDI, and ICA Connectivity

Device Number	2620-21x	2621-21	2629-2
Device name	MTI	MDI	ICA-II
Peripheral type	Com	Com	Com
Channel protocol	170	170	ICI
Requires DMA	No	No	Yes
PPs/channel	1	1	1
Adapters/channel	NA	NA	1
Elements/channel	1	1	NA
IOU connections:[1]			
I0 - NIO/170	NA	NA	NA
I0 - NIO/ICI	NA	NA	1.3.1
I1 - NIO/170	1.2.1	1.2.1	NA
I2 - NIO/170	1.2.1	1.2.1	NA
I4/I4A/I4AC - CIO/170/DMA	1.4.1[2]	1.4.1[2]	NA
I4/I4A/I4AC - NIO/170	1.2.1	1.2.1	NA
I4C - CIO/170	1.4.1[2]	1.4.1[2]	NA

1. This category indicates the NOS/VE version under which the specified IOU connection first became available.

2. For an MDI or an MTI to be connected to a NOS/VE PP via a CIO channel, FCO #49716 must be installed on the MCI card of the device interface.

Table E-11. NAD, MAP V, STORNET, and ESM-II Connectivity

Device Number	380-170	65354	5380-100	7040-200
Device name	NAD	MAP V	STORNET	ESM-II
Peripheral type	Com	EXT.CP	Com	Com
Channel protocol	170	170	170	170
Requires DMA	No	No	No	No
PPs/channel	1	1	1	1
Adapters/channel	NA	NA	NA	NA
Elements/channel	1	1	1	1
IOU connections:[1]				
I0 - NIO/170	1.2.3	NA	1.3.1	NA
I0 - NIO/ICI	NA	NA	NA	1.3.1
I1 - NIO/170	1.2.2	1.2.1	1.3.1	1.3.1
I2 - NIO/170	1.2.2	1.2.1	1.3.1	1.3.1
I4/I4A/I4AC - CIO/170/DMA	NA	NA	1.3.1	1.3.1
I4I4/I4A/I4AC - NIO/170	1.2.2	1.2.1	1.3.1	1.3.1
I4C - CIO/170	1.4.1	NA	1.4.1	1.4.1

1. This category indicates the NOS/VE version under which the specified IOU connection first became available.

Alternate and Redundant Path Configurations

Some disk and tape units configured to NOS/VE may be connected to two controllers or two I/O channels. Depending on the particular model of storage unit, various combinations of alternate and redundant paths may be configured. Table E-12 explains the model dependencies for alternate and redundant connections. Figures E-1 to E-9 illustrate the path configurations that can be used for various disk and tape subsystems.

An alternate path configuration is a configuration in which the storage units in a subsystem may be alternately (that is, not simultaneously) accessed by any active channel connected to the subsystem. In this context, an active channel is any channel not configured as a redundant channel.

In a redundant path configuration, only one channel may be used to transfer data between the mainframe and the storage unit or controller, although other (redundant) channels may be physically connected. In a redundant configuration that includes more than one controller, the controllers may transfer data simultaneously to different storage units.

A storage unit may never be simultaneously accessed by two different channels or controllers. This is true in any of the possible path configurations.

The NOS/VE VED DS operator display shows all the storage units in the configuration. In the description of each storage unit, one or two channels are displayed.

Table E-12. Alternate and Redundant Access

Storage Unit	Access Characteristics
698	The channels listed may be alternate or redundant channels if the controller is a $5698_1x. Otherwise (that is, with $698_2x controllers), they are alternate channels.
834, 836, 9836, 9853	The first channel listed is the only channel that NOS/VE is using to access the disk unit. The second channel listed (if present) is referred to by NOS/VE as a redundant channel.
844, 885	If channels are listed in the display, usually this means the channels are alternate channels. However, if your site has two or more channels configured to the same 7155_1 or 7155-1x controller, the first channel listed in the display is configured to access the disk unit and the other channels are redundant.
887	If two ports of the same channel are listed in the display, the channels are redundant. For example, the IOU connections CCH0A and CCH0B are redundant.
895	If two channels are listed in the display, they are always alternate channels.
5682	Only one channel is listed. A redundant controller connection is possible. However, redundant channels to a controller are not supported.

834 Disk Subsystem

In figure E-1, the two connections to the 10395-11 controller are redundant; that is, only one channel can be used to transfer data to or from the controller. NOS/VE does not support alternate paths to the 834 disk unit.

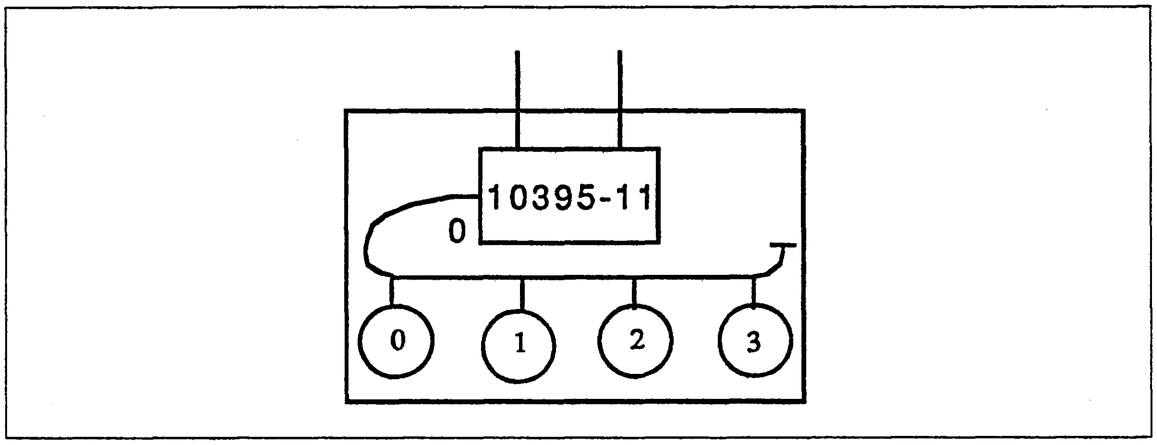


Figure E-1. 834 Disk Subsystem

836 Disk Subsystem

In figure E-2, the two connections to an 836 controller are redundant; that is, only one channel can be used to transfer data to or from a controller. NOS/VE does not support alternate paths to 836 disk units. The bold lines indicate the recommended static assignment of disk controllers to channels.

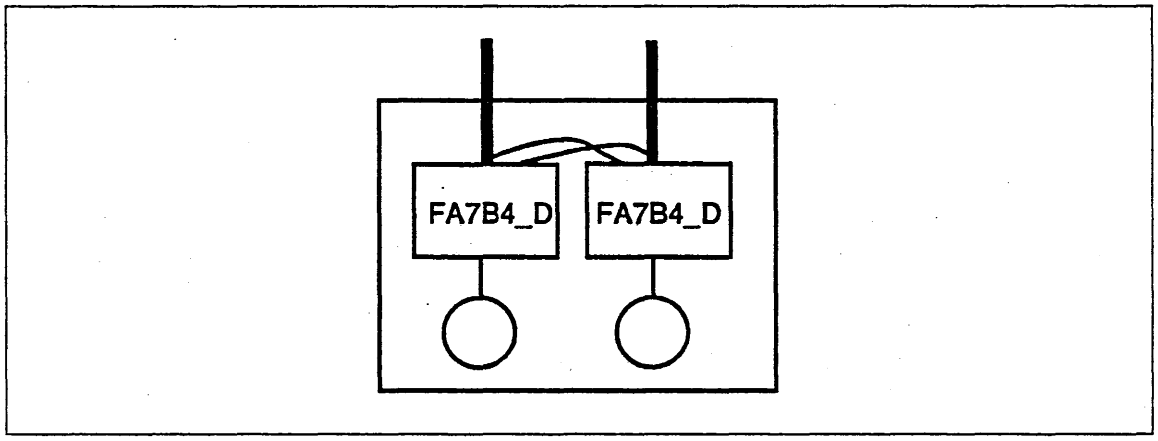


Figure E-2. 836 Disk Subsystem

844/855 Disk Subsystem

In figure E-3, all of the connections to an 844/885 controller are redundant; that is, only one of the channels connected to a controller may be used to transfer data between a mainframe and the controller. The connections to the 844 and 885 disk units are alternate connections; that is, either controller can transfer data to a particular disk unit, and both controllers may transfer data concurrently, each with a different disk unit.

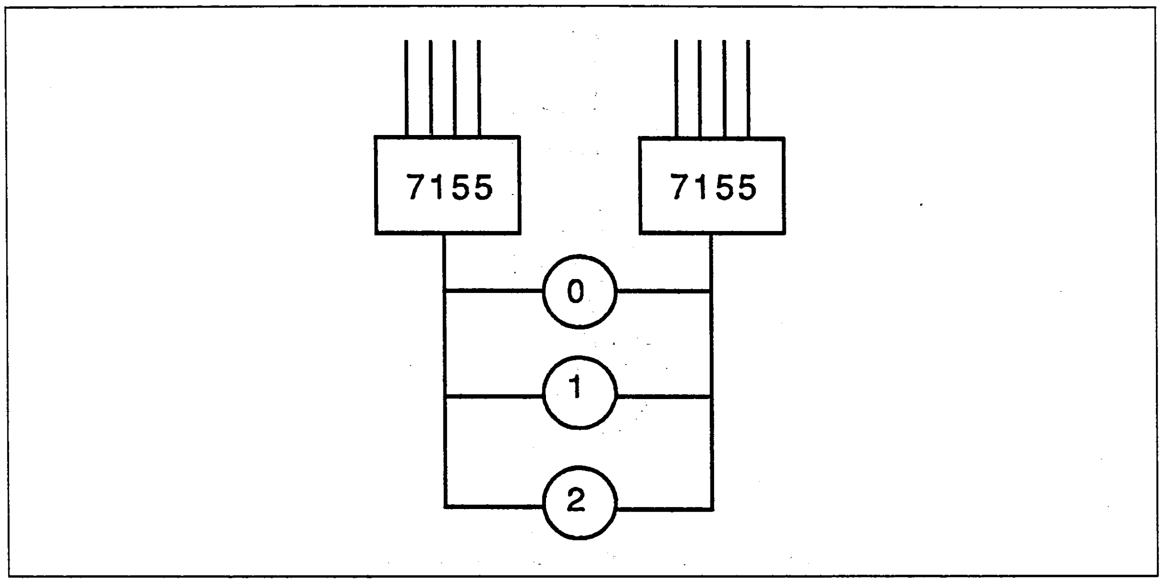


Figure E-3. 844/885 Disk Subsystem

887 Disk Subsystem

In figure E-4, two channels connected to an 887 subsystem may transfer data concurrently to two different disk units. The connections to 887 disk units are alternate connections; that is, either channel can transfer data to a particular 887 disk unit, and both channels may transfer data concurrently, each with a different disk unit.

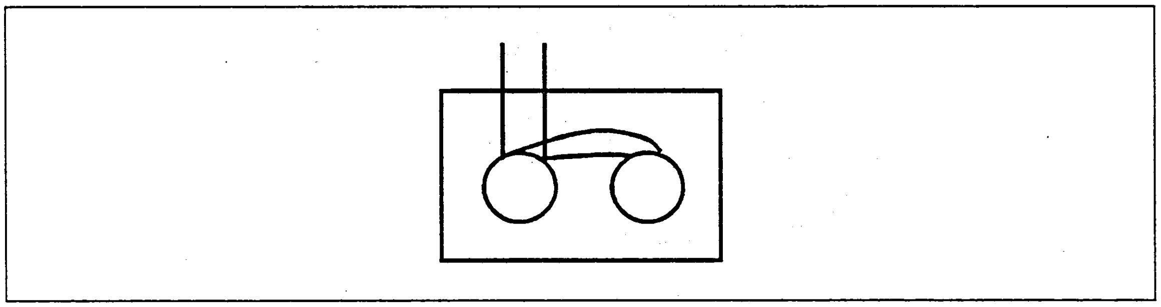


Figure E-4. 887 Disk Subsystem

895 Disk Subsystem

In figure E-5, the 7165 and 895-1 connections are concurrent; that is, any two units may transfer data simultaneously, each on a different channel. The connections to 895 disk units are alternate connections; that is, either channel can transfer data to a particular 895 disk unit, and both channels may transfer data concurrently, each with a different disk unit.

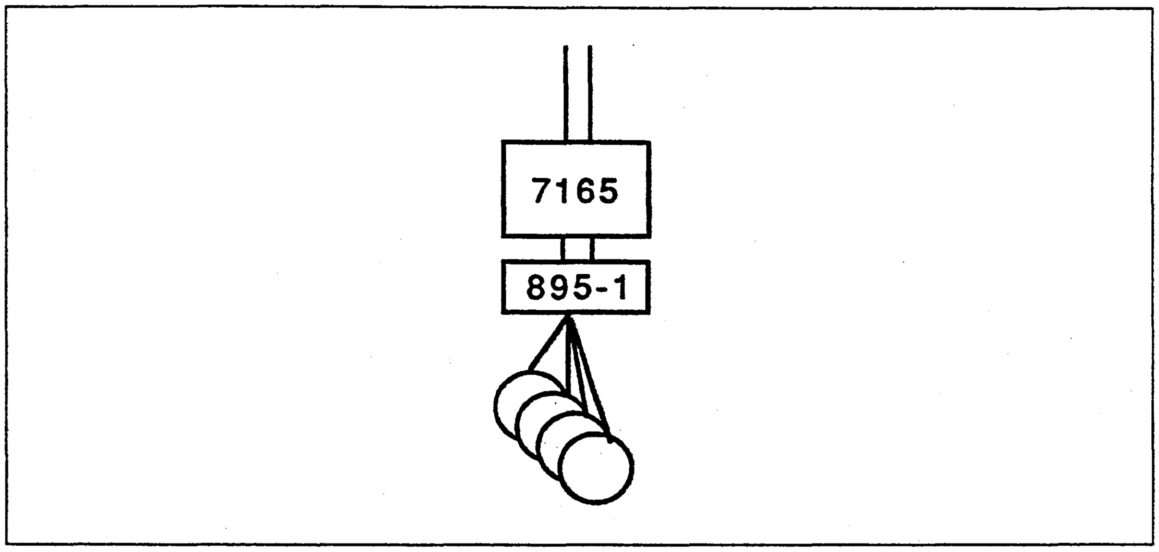


Figure E-5. 895 Disk Subsystem

9836/9853 Disk Subsystem

Figure E-6 shows the recommended path configuration for the 9836/9853 disk subsystem. The bold lines represent active connections, while the normal lines represent inactive (redundant) connections.

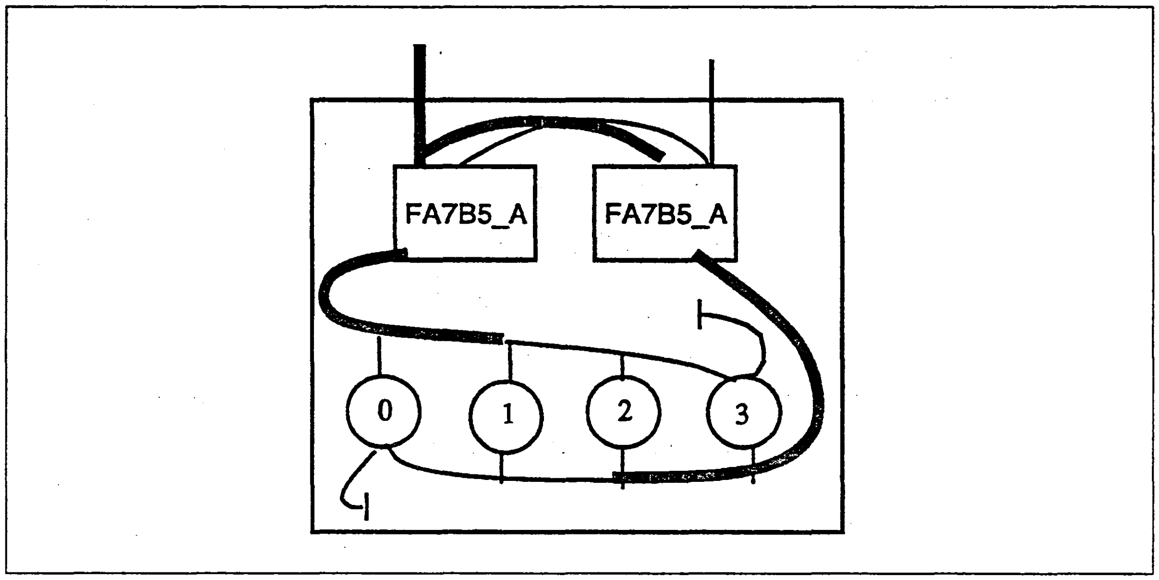


Figure E-6. 9836/9853 Disk Subsystem

As the figure shows, the channel connections to the FA7B5_A controllers are redundant; that is, only one of the channels may transfer data. However, the use of the channel is timeshared between the controllers so that both controllers are serviced by the same channel. In this configuration, the two controllers can concurrently access two different disk units and can transfer data at the sustained rate of the 9836 or 9853 disk unit. NOS/VE does not support alternate paths to 9835 or 9853 disk units.

5698/698 Tape Subsystem

The next two examples illustrate configurations showing an alternate connection and a redundant connection for 5698 tape controllers and 698 tape units. In figure E-7, each tape controller has a unique pair of channels connected to it. The connections to a 5698 controller are redundant; that is, only one of the channels may transfer data.

The connections to the tape units are alternate connections; that is, either controller may transfer data to a particular tape unit, and both controllers may transfer data concurrently, each with a different tape unit. The bold lines represent active connections, while the normal lines represent alternate connections.

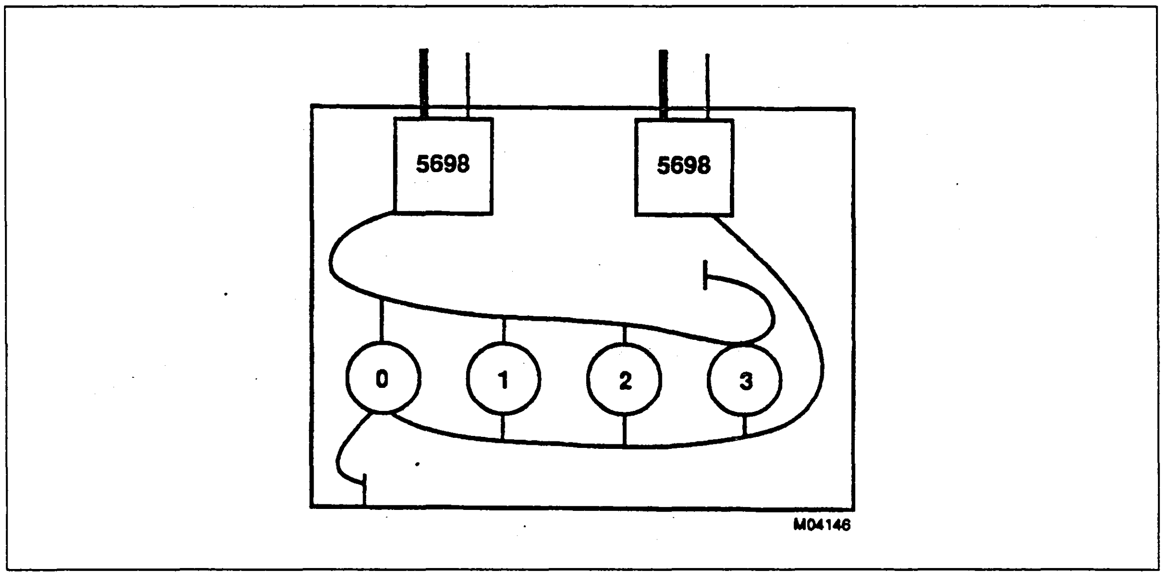


Figure E-7. 5698/698 Tape Subsystem

In figure E-8, each tape controller is connected to the same pair of channels. The connections to a 5698 controller are redundant; that is, only one of the channels may transfer data. In this configuration, the same channel is used to transfer data for both controllers, and a second channel provides redundant access to both controllers. In figure E-8, there is no redundant controller access to the tape units. The bold lines represent active connections, while the normal lines represent inactive (redundant) connections.

The two controllers may transfer data concurrently, each with one of its tape units.

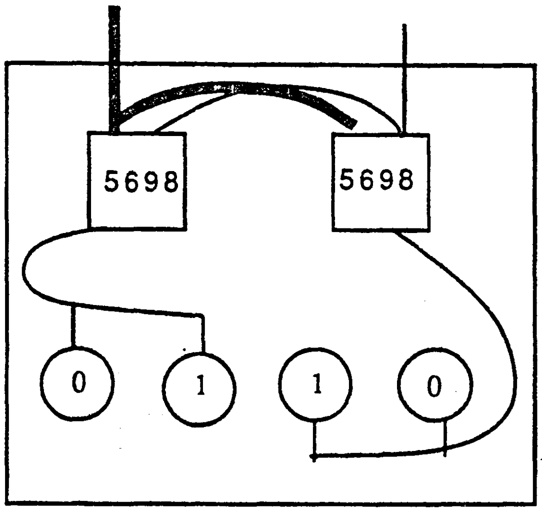


Figure E-8. 5698/698 Tape Subsystem

5680 Cartridge Tape Subsystem

In figure E-9, each tape unit is connected to two 5680 controllers. The tape units are equally divided between the controllers. If one controller fails, the faulty controller can be manually downed; access to all units reverts to the second (redundant) controller. In the figure, the bold lines represent active connections, while the normal lines represent inactive (redundant) connections.

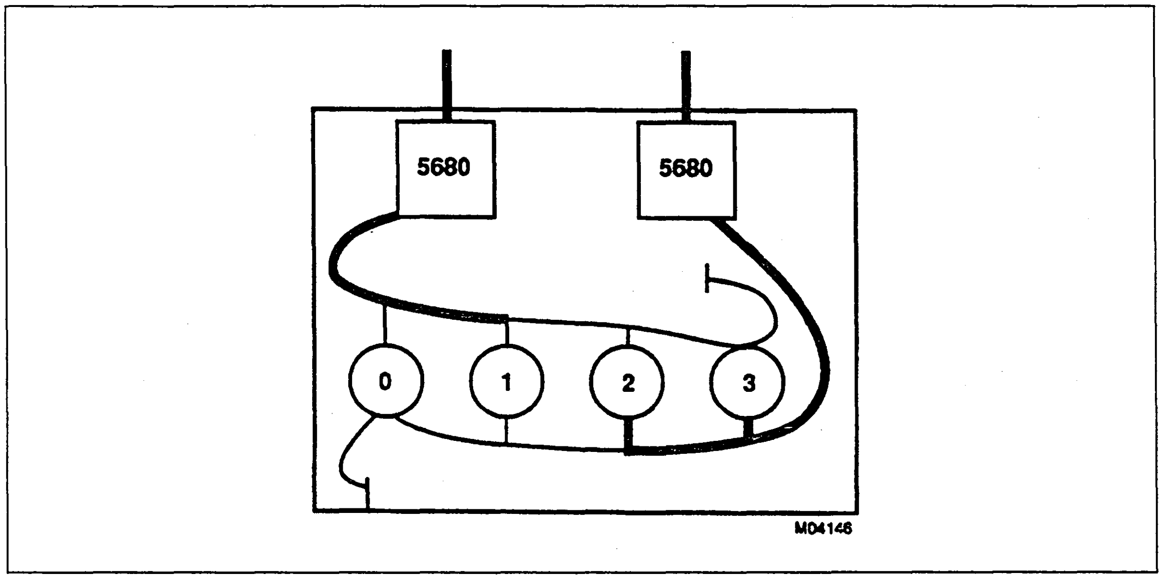


Figure E-9. 5680 Cartridge Tape Subsystem

Disk Unit Logical Constructs

The smallest unit of data transferred by NOS/VE is the minimum addressable unit (MAU). An MAU is the burst transfer supported by the disk unit's PP driver. The MAU concept is used to normalize the different device sector sizes for the central processor. When the central processor prepares requests for disk units, it specifies an integral number of MAUs. The PP driver for the specific storage unit then breaks down the MAU into physical address and sectoring considerations. MAUs always begin on sector boundaries. The page size must be a multiple of the MAU size.

If a particular type of disk unit has a sector size that is not a power-of-two bytes in length, the MAU ends in the middle of a sector and the content of the rest of the sector is indeterminate. The number of sectors spanned by an MAU is device-dependent.

A device allocation unit (DAU) is the unit of space allocation on a spindle of a disk unit. NOS/VE views a disk spindle as an array of DAUs that, as individuals or as a contiguous group, are assigned to a file. A DAU is an integral number of MAUs. The ratio of the number of MAUs to DAUs is device-dependent. A DAU can span tracks but cannot span cylinders of a device.

An allocation unit (AU) is the amount of space on a disk unit assigned to a file each time additional space is required. It is a power-of-two multiple of consecutive DAUs. Currently, the allocation unit is not selectable on a file-by-file basis. All user files default to an allocation unit size of 16,384 bytes.

Two additional terms often used in conjunction with disk units are page size and large sector formatting.

Page size, in NOS/VE, is a power-of-two number of bytes transferred as a block to or from central memory. NOS/VE currently supports page sizes of 2,048, 4,096, 8,192, and 16,384 bytes. However, not all mainframes support all page sizes, nor are certain page sizes supported on all disk products. For details, refer to the NOS/VE Software Release Bulletin.

Large sector formatting is a capability provided by controlware in the 7155 disk controllers. Large sector formatting allows the controller to group four physical sectors into one logical sector to decrease PP driver overhead. The term *MAU* is preferable to the term *large sector*, since not all controllers provide this capability to NOS/VE. One MAU on an 885 disk unit is four physical sectors of 516 8-bit bytes, which is the definition of a large sector.

Initializing Disk Units

During a NOS/VE installation deadstart, use the LCU subcommand INITIALIZE_MS_ VOLUME to prepare disk units for use by NOS/VE.

When you initialize an 885 volume that has not been previously initialized (or written in large sector format by diagnostic utilities), uncorrected checkword failures are reported for each of the three attempts to read a NOS/VE label. The reporting of these failures is expected and should not be a source of concern. The locations reported are tried in the following order:

Cylinder 0, track 0, sector 0
Cylinder 0, track 0, sector 4
Cylinder 0, track 0, sector 8

A NOS/VE label is written on each volume that is initialized.

Initializing an 895 disk unit causes NOS/VE to automatically format (soft sector) the unit. This process takes approximately five minutes.

The following is a discussion of utility map processing for the 844 and 885 disks. The 834, 836, 887, 895, 9836, and 9853 disks do not have utility maps since they are shipped without sector flaws.

844 Utility Map Processing

The INITIALIZE_MS_VOLUME subcommand uses the utility flaw map located at cylinder 822, track 0, sector 2 to flaw defects detected at the factory or by diagnostic programs administered by customer engineers. The utility map is an array of from 0 to 161 flaw entries. Each entry specifies either a track flaw or a sector flaw. Sector flaws are recorded in the physical sector numbering scheme.

For each sector flaw specified, the process logically flaws the affected DAU. For each track flaw, the process logically flaws the three DAUs affected. Because of the possibility of data mapping across tracks, from 0 to 4 physical sectors on the next track may also be logically flawed.

As a last step, the process flaws cylinders 820, 821, and 822 to reserve them for diagnostics use.

885 Utility Map Processing

The INITIALIZE_MS_VOLUME subcommand uses the utility flaw map located at cylinder 841, track 1, sector 1 to flaw defects detected at the factory or by diagnostic programs administered by customer engineering. The utility map is an array of from 0 to 161 flaw entries. Each entry specifies a track flaw. For each utility flaw map entry, the process logically flaws the four affected DAUs.

As a last step, the process flaws cylinders 841 and 842 to reserve them for diagnostic use.

Volume Defect Management Recommendations

After you have installed NOS/VE and initialized all of its disk volumes, it is possible that additional defects may appear on the disk surface.

Automatic Handling of Uncorrectable Defects

NOS/VE automatically records a device allocation unit (DAU) as flawed when a sector within the DAU is unreadable or unwritable due to a media failure. This process is performed only if all attempts at correcting the operation in the disk subsystem have failed. Since this process is performed by NOS/VE software, it is referred to as software flawing to distinguish it from flawing or relocation features that may be implemented in disk controllers.

Setting a software flaw prevents the DAU containing the defective sector from being assigned to another file. Generally, flawing a DAU implies the loss of one default allocation unit (16K bytes). However, if the cylinder in which the DAU exists is being used for a larger allocation size, the number of bytes in that larger allocation is no longer usable until the cylinder is freed and can be used for a smaller allocation unit.

The automatic actions taken by NOS/VE on an uncorrectable media defect depend upon whether the file was being written or read from the disk unit when the defect occurred.

Files Attached for Write Access

The processing of an uncorrectable defect for a file being written depends upon the life expectancy of the file.

Permanent Files

When an uncorrectable media defect is detected when writing a permanent file, NOS/VE attempts to automatically relocate data to another allocation unit on the same volume. Data in the original allocation unit is read from the device, if necessary and if possible, and merged with the data which was to be written. All pages in the allocation unit are marked as having been modified and are maintained in an error queue until they can be correctly rewritten to the new allocation unit.

NOS/VE makes at least six attempts to relocate the data in this manner to different locations on the same volume. These attempts are made each time the file is closed and each time it is detached from a job. Up to three attempts are made in each of these two cases. If these attempts fail, the modified data remains in memory until one of the following occurs:

- A relocation attempt succeeds.
- The file cycle is deleted.
- The next TERMINATE_SYSTEM (or a system interruption) occurs.

If the preceding data relocation attempts are unsuccessful, the user's task is given abnormal status when the file is closed and when it is detached. The abnormal status indicates that the file should be copied to a higher cycle or backed up immediately and then the original cycle deleted. This action succeeds because the unwritable data remains in memory.

Until the next system interruption, the original file cycle can still be attached and detached by jobs and the data retained in memory retains its integrity. However, abnormal status is still reported at each close and detach until the data is successfully written.

If the file is not copied or backed up prior to the next system interruption, the data that had been retained only in memory is lost. Following the system interruption, the file exists in a damaged state; however, the damage is not reported by the system when the file cycle is attached.

Temporary Files

If a temporary file is unwritable due to an uncorrectable media defect, the modified pages remain in memory until the file is deleted or the job terminates. The defect is completely transparent to the user and to the job environment.

The job is allowed to terminate because job termination deletes temporary files; therefore, the inability to write the data is of no concern.

Files Attached for Read Access

When a media defect is detected on a read operation, the flawed space remains assigned to the file until the file is deleted. After that, the flawed space is not reassigned to another file. The bad data is not delivered to the user.

Because automatically flawed space may remain assigned to a file, defect correction is attempted every time the file is referenced. Its success or failure is then reported to the engineering log and, ultimately, to an analyst via HPA/VE. This reporting occurs if the media defect is real and persistent; it may be neither of these.

Manual Handling of Correctable Defects

NOS/VE does not automatically record as flawed a sector that exhibits a correctable media defect. In fact, an occasional correctable defect on a disk volume is generally not a source of concern.

However, you may become concerned if the same correctable defect is reported repeatedly over a period of several days. An engineering log entry is produced every time the defective sector is read by a user, so that the frequency of occurrence of failure data reflects the frequency of use of the file involved.

NOS/VE provides subcommands in the Logical Configuration Utility to manually manage correctable media defects: DEFINE_MS_FLAW, REMOVE_MS_FLAW, and INITIALIZE_VOLUME. See chapter 3, Logical Configuration Utility, for details.

Defining Flaws

By specifying the DEFINE_MS_FLAW subcommand, you can manually define a disk flaw. More specifically, this step serves the following purposes:

- To ensure that an uncorrectable defect reported by NOS/VE has been automatically flawed.
- To remove from use a sector that has been reporting repeated, correctable read failures.

When you manually flaw a correctable media defect for a permanent file, NOS/VE automatically relocates the corrected data when the file assigned to the correctable defect is next attached. The data is relocated only if the defect can be corrected during the relocation process. If the defect is uncorrectable, the process for handling this condition is described under Automatic Handling of Uncorrectable Defects in this appendix.

Removing Flaws

You can also remove software flaws by specifying the LCU subcommand REMOVE_MS_FLAW. This subcommand returns to normal use a DAU that either had been automatically flawed by NOS/VE or manually with a DEFINE_MS_FLAW subcommand. If you remove a flaw that was automatically defined by NOS/VE, it is likely that the original problem will recur.

However, it is possible for a hardware failure in a controller or a storage unit to erroneously cause an uncorrectable media defect to be reported. This may cause NOS/VE to automatically flaw media even though the problem may not be a media defect at all. If the hardware problem is ultimately found and repaired, you may want to use the REMOVE_MS_FLAW subcommand to undo the software flaws on the device. Alternatively, you can initialize the media as described in the next section, Initializing Volumes.

You can display the list of software flaws for a particular volume by specifying the LCU subcommand DISPLAY_MS_FLAWS. This subcommand also displays reserved space on the volume. Reserved space consists of the following:

- Flaws defined by the factory or the customer engineer ($844_4x and $885_1x only).
- Space reserved for CIP.
- Space reserved for online maintenance.
- Space reserved for NOS/VE confidence testing.

Reserved space cannot be regained using the REMOVE_MS_FLAW subcommand.

Initializing Volumes

You initialize a NOS/VE volume with the LCU subcommand INITIALIZE_MS_VOLUME. When a volume is intialized, it is possible to retain or discard software flaws that may have been previously defined (manually or automatically). Before initializing a volume, be sure that you back up any files that may have resided on the volume.

Automatic Defect Prevention

When the device is idle, NOS/VE automatically moves the heads of 834, 836 and 885 disk units at 10-minute intervals to prevent media defects from occurring. This is not required for the 844 device and is automatically performed by the other disk subsystems supported by NOS/VE.

NOTE

For the 834, 836, 887, 895, 9836, and 9853 disks, there are no manufacturing flaws that need to be processed.

Maximum Tape Block Length

Magnetic tape files on NOS/VE consist of either of two block types: system-specified (SS) and user-specified (US). Open-reel tape files with a block type of SS (system-specified) are always limited to a maximum block length of 4,128 bytes. Open-reel tape files with a block type of US (user-specified) may be limited to a block length of 4,128 bytes in the case where only one PP is assigned to each tape channel. The default maximum block size for cartridge tape files is 32,640 bytes.

For more information about how to configure a system for processing tapes whose block size exceeds 4,128 bytes, see the description of the IOU_PROGRAM_NAME parameter of the DEFINE_ELEMENT subcommand in chapter 2, Physical Configuration Utility.

The maximum block length of US type tape files is determined by the smallest of these three values:

- The system page size as indicated in the following table:

Page Size	Maximum Tape Block Length
4,096 bytes	2 megabytes
8,192 bytes	8 megabytes
16,384 bytes	32 megabytes

- The maximum amount of real memory available to NOS/VE.
- The maximum job working set size minus 20 pages.

For user-specified (US) block types, selecting block sizes greater than 4,128 bytes can improve performance and capacity utilization of the tape volume as shown in tables E-13 and E-14. For the most efficient transfer, we recommend that users choose a block size that is a multiple of 960 bytes.

Table E-13. Open-Reel Tapes: Block Size, Performance, and Capacity Utilization

Maximum Block Size	Volume Capacity[1]	%Capacity Utilized[2]	Maximum Transfer Rate[3]
4128	121.66	68.77	0.86
8256	144.18	81.49	1.02
16512	158.88	89.80	1.12
33024	167.42	94.63	1.18
66048	172.04	97.24	1.22
132096	174.45	98.60	1.23
264192	175.68	99.30	1.24

1. Maximum block size is expressed in 10^6 bytes. Its calculation assumes a block size of MAXIMUM_BLOCK_LENGTH, a recording density of 6,250 bytes per inch, 0.30 inches of wasted space between blocks, and a tape volume length of 2,359 feet.

2. The following denominator is used to calculate the capacity utilized:

```
2359 ft. * 12 in./ft. * 6250 bytes/in.
```

3. Maximum transfer rate is expressed in 10^6 bytes/second. Achieving the maximum transfer rate depends on the ability of the program to sustain the data rate of the subsystem.

Table E-14. Cartridge Tapes: Block Size, Performance, and Capacity Utilization

Maximum Block Size	Volume Capacity[1]	%Capacity Utilized[2]	Maximum Transfer Rate[3]
4128	139.80	57.98	1.74
8256	176.99	73.40	2.20
16512	204.14	84.66	2.54
33024	221.10	91.69	2.75
66048	230.68	95.67	2.87
132096	235.79	97.79	2.93
264192	238.43	98.88	2.97

1. Maximum block size is expressed in 10^6 bytes. Its calculation assumes a block size of MAXIMUM_BLOCK_LENGTH, a recording density of 37,871 bytes per inch, 0.079 inches of wasted space between blocks, and a tape volume length of 530.59 feet.

2. The following denominator is used to calculate the capacity utilized:

```
530.59 ft. * 12 in./ft. * 37,871 bytes/in.
```

3. Maximum transfer rate is expressed in 10^6 bytes/second. Achieving the maximum transfer rate depends on the ability of the program to sustain the data rate of the subsystem.

Peripheral Device Descriptions

The following sections summarize characteristics of various peripheral devices.

$10395_11 Disk Controller

The $10395_11 product group consists of 834 disk controllers (control modules). This product supports two nonconcurrent channel accesses. The $10395_11 product group supports only the $834_12 disk unit.

The $10395_11 may be connected to one or two ICI (integrated controller system) channels; however, the two channels cannot use the $10395_11 concurrently or alternately. The second connection is purely for redundancy in case of a channel failure. At deadstart, NOS/VE uses the first IOU connection described in the PCU subcommand DEFINE_ELEMENT. The $10395_11 may be connected by an ISI (intelligent standard interface) cable to one or two 7255-1 channel adapters. The 7255-1 adapts an (internal) CYBER ICI channel to an ISI interface.

The $10395_11 may be connected to another $10395_11 via a daisy-chain cable arrangement. Up to eight $10395_11 control modules may be connected to the same cable. A $10395_11 control module cannot be connected twice to the same cable; it can, however, be connected to two different daisy chains. If a $10395_11 is connected to two daisy chains, it must have the same address in both chains. Each chain, in turn, must be connected to separate ICI channels.

A $10395_11 cannot be connected to the same string as a $FA7B4_D controller.

A $10395_11 may be connected to from one to four $834_12 disk units. The disk units are addressed as 0 through 3 with respect to the $10395_11.

On CYBER model 810 and 830 mainframes, only channels 0, 1, 3, 6, 16, and 22 can be connected to a $10395_11 control module.

$262x_21x MTI and MDI

The $262x_21x product group consists of the $2620_210 and $2620_211 mainframe terminal interfaces (MTI) and the $2621_210 and $2621_211 mainframe device interfaces (MDI). Only one CYBER 170 channel may be connected to a member of this product group, and the MTI or MDI must be the only element connected to the channel.

On an I1, I1CR, or I2 IOU, the channel must be a CYBER 170 channel. On an I4 or I4A IOU, the channel must be a CYBER 170 or CYBER 170/DMA channel. On an I4C IOU, the channel must be a CYBER 170/DMA channel. If a CYBER 170/DMA channel is used, the MTI or MDI must have FCO number 49715 installed.

These devices may not be connected to the CYBER 170 available on an IOU.

$2629_x ICA

The $2629_x product group consists of the $2629_1 and $2629_2 integrated communications adapters (ICAs) for CYBER 930-series systems. Only one $2629_x may be connected to an ICI channel, and only one ICI channel may be connected to a $2629_x. As of NOS/VE Version 1.5.1, the $2629_1 is no longer supported.

$380_170 NAD

The $380_170 is a network access device (NAD) communications element. It can connect to one and only one CYBER 170 channel, and it must be the only element connected to the channel.

On an I0, I1, I1CR, or I2 IOU, the channel must be a CYBER 170 NIO channel. On an I4 or I4A IOU, the channel must be a CYBER 170 or CYBER 170/DMA channel. On an I4C IOU, the channel must be a CYBER 170/DMA channel.

$5380_100 STORNET

The $5380_100 product group consists of an externally attached, word-addressable memory (STORNET). The $5380_100 has a physical memory size of four million 60-bit words. The physical memory size can be degraded by using the MAX ECS SWITCH and HALF ECS SWITCH.

A maximum of eight low-speed ports can only be connected to 12-bit CYBER 170 channels. A low-speed port may be connected to a nonconcurrent I/O (NIO) channel or a concurrent I/O (CIO) channel of an I4 IOU. The $5380_100 can be connected to a maximum of eight mainframes via these low-speed ports. Multiple connections to the same mainframe are allowed. However, no other equipment can be connected to a channel that is servicing a STORNET subsystem and a channel may only be connected to one low-speed port of a $5380_100.

Two side-door ports can be connected to a 12-bit CYBER 170 NIO channel on any CYBER 180 mainframe. Only one side-door port is active at any given time. A hardware switch controls which side-door port is active. The channel connected to the side-door port and the channel connected to the CIP device must both be accessible to the same PP. The side-door port is used for the collection of STORNET maintenance information. The side-door port is not required for proper functioning of the low-speed ports.

On an I0, I1, I1CR, or I2 IOU, the channel must be a CYBER 170 channel. On an I4 or I4A IOU, the channel must be a CYBER 170 or CYBER 170/DMA channel. On an I4C IOU, the channel must be a CYBER 170/DMA channel.

$568x Cartridge Tape Subsystem

The $568x product group consists of the $5680_11 cartridge tape controller and the $5682_12 and $5682_14 cartridge tape cabinets. The $5682_12 cabinet contains two tape units, and the $5682_14 cabinet contains four tape units.

The $5680_11 controller provides access to 2 to 16 $5682_1x tape units.

The $5682_1x products are 18 track and support 79 inches per second operation. Recording density is approximately 38,000 cpi.

The following restrictions apply to configuring $5680_11 and $5682_1x elements:

- A $5680_11 controller can only be connected to CYBER IOUs having a 4X PP speed (I2, I4, I4A, I4AC, OR I4C). CYBER models 810, 815, 825, 830, 930, and 932 are not supported.

- A $5680_1x subsystem can only be connected to a CYBER 170 NIO or CIO channel.

- Each channel configured requires two PPs.
- A maximum of one controller can be configured per CYBER 170 channel. The equipment number of the controller must be 0 (zero).
- A maximum of two controllers can be configured per subsystem.
- A maximum of eight units can be configured per controller in the subsystem; that is, if two controllers are installed, a maximum of 16 units can be configured.
- Only the first peripheral connection of a $5682_1x that is in the ON state is actively configured because the two controllers automatically divide the tape units between themselves on the basis of odd and even tape unit addresses. The second peripheral connection is used only for redundancy.
- Changing the state of a $5680_11 controller or its channel to OFF or DOWN ensures that NOS/VE does not issue commands to the controller until the controller's use is reinstated (that is, until its state is changed back to ON). However, the two controllers in a redundant access configuration are hardware interconnected. Therefore, before repairing or otherwise maintaining a controller in this configuration, set it offline (by toggling a switch). This ensures that commands sent to the online controller are not processed on the unavailable controller.

$5698_1x Tape Subsystem

The $5698_1x product group consists of $5698_10, $5698_11, and $5698_12 IPI (intelligent peripheral interface) tape controllers and $698_3x 9-track tape units.

The $5698_1x controller provides access to two to eight $698_3x tape units. The $698_30 and $698_31 products support 200 inches per second operation and allow recording at 1600 cpi (phase encoded) or 6250 cpi (group-coded recording).

The following restrictions apply to configuring $5698_1x and $698_3x elements under the current version of NOS/VE:

- A $5698_1x controller must be actively connected to only one mainframe. NOS/VE does not allow two mainframes to actively contend for the controller. However, you can physically connect a controller to more than one mainframe as long as only one mainframe has the controller actively configured at any given time.
- If two controllers are actively configured to access the same tape unit, the controllers must be actively configured to the same mainframe.
- A $5698_1x controller can only be connected to an IPI channel.
- A maximum of eight controllers can be configured per channel. On an I4 IOU, eight controllers can be connected to each port of the channel.
- A maximum of eight $698_3x tape units can be configured per controller. This means that a maximum of 64 tape units (128 tape units on an I4 IOU) can be configured per channel.
- Only the first IOU connection of a $5698_1x controller that is in an ON state is actively configured. This is because the two host ports of the controller are incapable of simultaneous data transfer. Accordingly, a $5698_1x controller reads or writes data for only one tape unit at a time. The second IOU connection is used for redundancy.

- If two or more $5698_1x controllers are connected to the same string (channel or channel port), the controllers must not be connected to common $698_3x tape units. Although two controllers in the same string can be physically connected to common units, you are allowed to define only one of the unit's two controller connections in this case. See the PERIPHERAL_CONNECTION parameter of the PCU subcommand DEFINE_ELEMENT in chapter 2, Physical Configuration Utility.

Consider the following recommendations when configuring $5698_1x and $698_3x elements:

- Although NOS/VE supports up to eight $5698_1x controllers per channel (or 16 on an I4 IOU), configure no more than two controllers per channel for maximum performance. Such a configuration provides the same performance as configuring a single controller on each of two channels, as long as the operator assigns tape units appropriately as explained in the next item.

- If you have only one IPI channel to devote to tape processing but want to configure two controllers on the same channel, do so. However, be aware that the operator must only assign a tape to one unit per controller in order to ensure maximum performance. Otherwise, the tape units connected to the same controller will contend for the controller.

- If you have two IPI channels to devote to tape processing, configure for alternate access. In this case, each tape unit is connected to two different controllers that are, in turn, connected to different host channels. This allows the tape unit to be accessed by either channel in a way similar to the alternate-access capability provided by a single $7021_32 or $698_2x controller.

- If you have more than one IPI channel to devote to tape processing, configure each controller to more than one host channel. This allows redundancy in case a host channel requires repair. For greater redundancy, connect the two $5698_1x host ports to different clusters on either an I4 or I0 IOU.

- If you connect $5698_1x controllers to different ports of the same channel, the limitation that only two controllers can be accessed with maximum performance still applies: the channel's ports are not capable of simultaneous data transfer.

$639_1 Tape Unit

The $639_1 product group consists of 9-track tape units. The 639 tape units support 25 or 75 inches per second operation and allow recording at 1600 cpi (phase encoded) or 6250 cpi (group-coded recording).

On CYBER model 810 or 830 mainframes, only channels 0, 1, 3, 6, 16, and 22 can be connected to a $639_1 channel adapter. Only one $639_1 may be configured for each CYBER model 810 or model 830 channel. The address of the $639_1 must be zero.

With one PP assigned per channel, the $639_1 can transfer block lengths of up to 4,128 bytes. Using two PPs per channel, the $639_1 can transfer blocks of any size. See Maximum Tape Block Length earlier in this appendix for more information about long block transfers.

See Maximum Tape Block Length for information about the use of tape blocks greater than 4,128 bytes.

$65354_1x MAP V

The $65354_1x product group consists of externally attached matrix algorithm processors (MAP V). The models supported are the $65354_10, $65354_11 and the $65354_12. The $65354_10 has only a CYBER 170 channel connection for both control and data transfers. The $65354_11 and the $65354_12 have a direct central memory connection and a CYBER 170 channel connection. The CYBER 170 channel is used for passing control information to the attached processor, and the central memory connection is used for high-speed data transfer. The $65354_11 only connects to a model 835, and the $65354_12 can only be connected to the models 840 to 860.

This product group runs on the CYBER 170 channel available on the I1, I1CR, or I2, I4, and I4A IOU. It will not run on the CYBER 170 channel available on the IOU.

$679_x Tape Subsystem

The $679_x product group consists of the advanced tape subsystem (ATS) 9-track tape units. The capabilities of each tape unit are listed in table E-8; the x in the product group identifier is replaced with a digit from 2 to 7.

Table E-15. Recording Density in Characters per Inch

Product Identification	Speed in Inches per Second	Nonreturn to Zero (NRZI)	Phase Encoded (PE)	Group-Coded Recording (GCR)
$679_2	100	800	1600	
$679_3	150	800	1600	
$679_4	200	800	1600	
$679_5	100		1600	6250
$679_6	150		1600	6250
$679_7	200		1600	6250

With one PP assigned per channel, the $679_1 can transfer block lengths of up to 4,128 bytes. Using two PPs per channel, the $679_1 can transfer blocks of any size. See Maximum Tape Block Length earlier in this appendix for more information about long block transfers.

See Maximum Tape Block Length for information about the use of tape blocks greater than 4,128 bytes.

$698_xx Tape Subsystem

The $698_xx product group consists of $698_1x and $698_2x tape controllers and $698_3x tape units.

The $698_10, $698_11, and $698_12 products provide single-channel access for up to eight $698_3x tape units.

The $698_20, $698_21, and $698_22 products provide dual-channel access for up to eight $698_3x tape units.

The $698_30 and $698_31 products are 9-track, 200 ips, 1600 cpi (PE) or 6250 cpi (GCR) tape units.

With one PP assigned per channel, the $698_1x and $698_2x can transfer block lengths of up to 4,128 bytes. Using two PPs per channel, they can transfer blocks of any size. See Maximum Tape Block Length earlier in this appendix for more information about long block transfers.

$7021_3x Tape Controller

The $7021_3x product group consists of advanced tape subsystem (ATS) controllers. The digit that replaces the x denotes the number of channels that may be connected to the controller; x can be 1 or 2. In a $7021_32 controller, the data and control transfers to both channels can occur simultaneously since there are two independent controllers in the product. Because there are two independent controllers in a 7021-32, NOS (or NOS/BE) and NOS/VE can each have its own controller.

In a standalone NOS/VE system, a $7021_32 controller is defined as follows:

```
define_element e=blue_ats ei=$7021_32 sn=xxx ..
iou_connection=((chxx),(chyy))
```

On a dual-state system in which one channel to the $7021_32 is used by NOS (or NOS/BE) and the other by NOS/VE, only the channels to be used by NOS/VE should be described in the NOS/VE configuration. If both channels are described, NOS/VE attempts to acquire both channels from NOS (or NOS/BE) during deadstart. If both channels are defined to, and acquired by NOS/VE, one or more may be returned to NOS (or NOS/BE) by using the CHANGE_ELEMENT_STATE subcommand.

With one PP assigned per channel, the 7021 can transfer block lengths of up to 4,128 bytes. Using two PPs per channel, the 7021 can transfer blocks of any size. See Maximum Tape Block Length earlier in this appendix for more information about long block transfers.

You can toggle the 7021 controllers from NOS (or NOS/BE) to NOS/VE, and vice versa, with the CHANGE_ELEMENT_STATE command. For more information on this command, refer to chapter 3, Logical Configuration Utility.

NOS/VE supports only 9-track tape units. The 9-track tape units can support:

- Nonreturn to zero (NRZI)
- Phase encoded (PE)
- Group-coded recording (GCR)

The 7021 controllers support from one through eight tape units. A channel can access only one tape unit at a time.

The equipment number for $7021_3x controllers in the PCU subcommand DEFINE_ELEMENT must be 0.

A channel can be connected to only one controller.

On an I1, I1CR, or I2 IOU, the channel must be a CYBER 170 channel. On an I4 or I4A IOU, the channel must be a CYBER 170 or CYBER 170/DMA channel. On an I4C IOU, the channel must be a CYBER 170/DMA channel.

These devices may not be connected to the CYBER 170 available on an IOU.

$7040_200 ESM-II

The $7040_200 product group consists of an externally attached word-addressable memory (ESM-II). The $7040_200 has a physical memory size ranging from one million to sixteen million 60-bit words. The physical memory size can be degraded by using the MAX ECS SWITCH and HALF ECS SWITCH.

A maximum of eight low-speed ports can only be connected to 12-bit CYBER 170 channels. A low-speed port may be connected to a nonconcurrent I/O (NIO) channel or a concurrent I/O (CIO) channel of an I4 IOU. The $7040_200 can be connected to a maximum of eight mainframes via these low-speed ports. Multiple connections to the same mainframe are allowed. However, no other equipment can be connected to a channel that is servicing an ESM-II subsystem and a channel may only be connected to one low-speed port of a $7040_200.

Two side-door ports can be connected to a 12-bit CYBER 170 NIO channel on any CYBER 180 mainframe. Only one side-door port is active at any given time. A hardware switch controls which side-door port is active. The channel connected to the side-door port and the channel connected to the CIP device must both be accessible to the same PP. The side-door port is used for the collection of ESM maintenance information. The side-door port is not required for proper functioning of the low-speed ports.

On an I1, I1CR, or I2 IOU, the channel must be a CYBER 170 channel. On an I4 or I4A IOU, the channel must be a CYBER 170 or CYBER 170/DMA channel. On an I4C IOU, the channel must be a CYBER 170/DMA channel.

These devices may not be connected to the CYBER 170 available on an IOU.

$7155_1x Disk Controller

The $7155_1x product group consists of disk controllers. Within the product group there are two types of products, the $7155_1 and the $7155_1x. The digit that replaces the x denotes the number of channels to which the controller may be connected; x can be 1, 2, 3, or 4. If the controller is connected to more than one channel, only one channel may be defined in the NOS/VE configuration. Except for ISHARE devices (which cannot be shared) a controller can be shared by NOS/VE and NOS (or NOS/BE) as long as each operating system has its own channel access. The 7155-1 and the 7155-11 controllers are functionally equivalent.

The 7155-1 and 7155-1x controllers support from one to eight 844 disk units and from one to eight 885 disk unit spindles (four 885 disks). There is a hardware option available to increase the number of 885 disk unit spindles to 16 (eight 885 disks).

The 7155-1 and the 7155-1x controllers support both large (16,512 bits) and small (3,864 bits) sector sizes. NOS/VE uses small sectors for data transfers when accessing the $844_4x product group and large sectors for accessing the $885_1x product group. The large sector is a logical grouping of four physical sectors. Disk units used on NOS (or NOS/BE) and then on NOS/VE do not require physical reformatting.

If a site selects a 7155 controller and an 885 disk unit combination for NOS/VE, the controller must have the large-sector format hardware option installed and use level 721-D10 or later controlware. The large-sector format is designated by the final letter B in the hardware serial number. Alternatively, you can choose a model whose serial number ends in A, provided you have option 65290-1 installed.

The equipment number for either a 7155-1 or 7155-1x controller in the PCU subcommand DEFINE_ELEMENT must be 0.

A channel can be connected to only one controller.

On an I1, I1CR, or I2 IOU, the channel must be a CYBER 170 channel. On an I4 or I4A IOU, the channel must be a CYBER 170 or CYBER 170/DMA channel. On an I4C IOU, the channel must be a CYBER 170/DMA channel.

These devices may not be connected to the CYBER 170 available on an IOU.

$7165_2x Disk Controller

The $7165_2x product group consists of disk controllers. Each $7165_2x consists of two CYBER channel couplers (CCCs) and one storage control unit (SCU). The CCCs are in one cabinet and the SCU is in another. The SCU contains two storage directors (SDs). A CCC connects a CYBER 170 channel to an SD.

There are two products in the $7165_2x product group, the $7165_21 and the $7165_22. A $7165_21 provides two single-ported SDs. A $7165_22 provides two dual-ported SDs. The dual-port option allows an SD to be accessed by up to two CCCs. In the $7165_22, the two CCCs that are connected to the same SD must be connected to different mainframes if both CCCs are described in the physical configuration; otherwise, one CCC can reference both SDs and the second CCC is omitted from the physical configuration.

The following hardware and software configuration restrictions apply to the $7165_2x product group.

Hardware Configuration Restrictions:

- A CCC must be the only device connected to the CYBER 170 channel. Its equipment number must be zero (0).
- A CCC can connect to from 0 to 8 SDs. Refer to the first software restriction; the supported maximum is actually 2 SDs.
- Each 895-1 head-of-string controller (HSC) can access from 4 to 16 disk units.
- Each SD can access up to two 895-1 HSCs; therefore, an SD can access from 8 to 32 independent storage units.
- Although each SD of a $7165_22 can be connected to two CCCs, only one CCC may actually use the SD. The second CCC provides redundant access.
- An 895-1 HSC has two ports and each port has the same address (0 or 1).
- Only the PPs in the same five-PP CIO (concurrent I/O) cluster of an I4 IOU can access the same CIO channel.
- An NIO (noncurrent I/O) PP of an I4 IOU cannot access a CIO channel, and a CIO PP cannot access an NIO channel.

Software Configuration Restrictions:

- One CCC can connect to 0 to 2 SDs. CTI and offline maintenance impose this restriction.

- Due to the preceding restriction, a CCC can access from 4 to 64 storage units.
- Achieving the maximum configuration of 64 storage units per CCC requires two $7165_2x cabinets to be daisy-chained together. The two CCCs in one of the cabinets are not used. The two SDs in one cabinet must have the address zero (0), and the two SDs in the second cabinet must have the address one (1). Each SD is connected to two 895-1 HSCs, each with 16 storage units. One 895-1 HSC has address 0, and the other has address 1.
- NOS/VE does not support the definition of the CCC and the 895-1 HSC in the physical configuration because MALET/VE can only process a physical path that consists of (channel, equipment, unit). Since there are actually five elements in the path to a $895_2 disk unit, the CCC and 895-1 HSC are omitted. Therefore, the address from the CCC to the SD is defined in the IOU_CONNECTION parameter of the $7165_2x, and the address from the SD to the disk unit is defined in the PERIPHERAL_CONNECTION parameter of the disk unit. The NOS/VE address to the disk unit is determined by multiplying the address of the 895-1 HSC (0 or 1) by 16(10) and adding the physical unit number of the disk unit (0 to 15).

  For example, if the physical unit number of a disk unit is 15 and the address of the 895-1 HSC is 1, the calculation for determining the disk unit's NOS/VE address is as follows:

  address=(1x16)+15=31

  The result of this calculation (31) is then specified in the PCU subcommand DEFINE_ELEMENT for the device. For example:

  ```
  define_element $895_2 e=blue31 ei=$895_2 sn=345 ..
  pc=((blue_controller 31))
  ```

- Each $7165_2x product (consisting of two CCCs and two SDs) is defined as a single element in the NOS/VE configuration. However, defining the $7165_2x as one element necessitates the following restrictions:
  - Both SDs in the same cabinet must have the same address; there is no way to refer to the individual SDs.
  - Both SDs must be connected to the same 895-1 HSCs; there is no way to refer to the individual SDs.
  - All of the $895_2 disk units connected to one SD must also be connected to the second SD in the same SCU cabinet.
- NOS/VE assumes that all channels connected to a $7165_2x product and in the ON state can be used concurrently. Therefore, if a $7165_22 controller is connected to more than two channels, only one channel connected to each SD can be described in the configuration of any mainframe. A $7165_22 is used primarily for redundancy in case a mainframe fails. This restriction exists because:
  - There could be up to four channels defined in the IOU connection of a $7165_22 and all four channels would have the same equipment number. This results because the two SDs in the same cabinet must have the same address and each has up to two CCCs connected to it.
  - NOS/VE cannot determine (by using only the equipment number) which two of the four possible channels can be used concurrently and which channels are for redundant access.

- Two PPs are required to drive each CYBER 170 NIO channel that is in the ON state and connected to a $7165_2x subsystem. Only one PP is required for each CYBER 170 CIO channel that is in the ON state and connected to a $7165_2x subsystem.
- In a standalone NOS/VE configuration with an $895_2 unit as the CIP device, the channel identified in the deadstart panel must be an NIO channel. This is a restriction imposed by CTI.
- The preceding restriction implies that if a $7165_21 or $7165_22 controller is to be connected only to CIO channels in a standalone NOS/VE configuration, some other disk unit connected to an NIO channel must contain CIP and must be the mainframe deadstart device. However, in this case the NOS/VE system device could be a $895_2 disk unit that is accessible only from CIO channels.

It is possible to connect a $7165_21 or $7165_22 controller to any combination of NIO (nonconcurrent) or CIO (concurrent DMA) CYBER 170 channels.

On an I1, I1CR, or I2 IOU, the channel must be a CYBER 170 channel. On an I4 or I4A IOU, the channel must be a CYBER 170 or CYBER 170/DMA channel. On an I4C IOU, the channel must be a CYBER 170/DMA channel.

These devices may not be connected to the CYBER 170 available on an IOU.

You may configure an $895_2 unit to be the NOS/VE system device regardless of which combination of NIO and CIO channels you use to access the device.

$7221_1 Tape Channel Adapter

The $7221_1 product group consists of tape channel adapters that connect a CYBER ICI channel to a $639_1 tape unit. The $7221_1 can be connected to only one CYBER ICI channel and only one $639_1 tape unit.

On CYBER model 810 and 830 mainframes, only channels 0, 1, 3, 6, 16, and 22 can be connected to a $7221_1 channel adapter. The address of the $7221_1 is 0.

With one PP assigned per channel, the 7221 can transfer block lengths of up to 4,128 bytes. Using two PPs per channel, the 7221 can transfer blocks of any size.

$7221_11 Tape Channel Adapter

The $7221_11 product group consists of tape channel adapters that connect a CYBER ICI channel to a $9639_1 tape unit. The $7221_11 is only available on a CYBER 930-series mainframe.

The $7221_11 is connected to only one CYBER ICI channel and may have up to two attached $639_1 tape units. The address of the $7221_11 is 0.

With one PP assigned per channel, the 7221 can transfer blocks of any size. See Maximum Tape Block Length for more information about long block transfers.

$834_12 Disk Unit

The $834_12 product group consists of disk units that are nonremovable. The $834_12 can only be connected to one $10395_11 control module.

The $834_12 is packaged in a cabinet that can contain from one to four $10395_11 control modules and two to four $834_12 disk units. On model 810 and model 830 systems, one or two of the cabinets may be installed.

For the CYBER model 810 and 830 mainframes, only channels 0, 1, 3, 6, 16, and 22 can be connected to the control module for the $834_12.

The address of the $834_12 can be in the range 0 to 3.

The $834_12 is defect-free at shipment and has no flaw map for use by the operating system.

$836_xxx Disk Subsystem

The $836_xxx product group consists of varying numbers of control modules, disk units, and cabinets.

- The $836_110 product consists of one $FA7B4_D control module and one disk unit, without a cabinet (it is installed in an existing cabinet).
- The $836_221 product consists of two $FA7B4_D control modules, two disk units, and one cabinet.
- The $836_441 product consists of four $FA7B4_D control modules, four disk units, and one cabinet.

The 836 disk units have nonremovable media. An 836 control module can be connected to only one disk unit and a disk unit can only be connected to one control module. Control modules and disk units are provided in pairs, with one control module and one disk unit in each pair. An 836 cabinet contains two, three, or four such pairs. One or two of the cabinets may be installed on a CYBER 810 or 830 system.

For the CYBER model 810 and 830 mainframes, only channels 0, 1, 3, 6, 16, and 22 can be connected to the control module for the $836_xxx.

The $836_xxx is defect-free at shipment and has no flaw map for use by the operating system.

$844_4x Disk Unit

The $844_4x product group consists of double-density 844 disk units. The x can be 1 or 4. A 1 means the disk unit can be connected to a maximum of two controllers. A 4 means the disk unit can be connected to a maximum of four controllers.

The 844 disk units have removable volumes, but the volumes cannot be removed during NOS/VE execution. When NOS/VE is terminated, the volumes can be put on any drive that is actively configured, and NOS/VE recognizes them during the next continuation deadstart.

The 24 sectors per track on an 844_4x disk unit are read and written in groups of five sectors, so there are logically 4.8 sectors per track. Each of the first four sectors in each group has 480 usable 8-bit bytes. The fifth sector has 128 usable bytes.

$885_1x Disk Unit

The $885_1x product group consists of disk units. The digit that replaces the x refers to the number of controllers to which the disk units can be connected; x can be 1 or 2.

The 885 disk units have two spindles per cabinet. Each spindle has its own unit number and is logically independent of the other. The spindles are not removable.

The 32 sectors on each track of an 885 disk unit are read and written in groups of four sectors, so there are logically eight sectors per track.

If NOS/VE accesses both spindles on an 885 disk unit, you need two PCU DEFINE_ELEMENT subcommands, one for each spindle.

$887_1 Disk Unit

The $887_1 product consists of two disk units. Each disk unit has a controller connecting it to one or two mainframe channels. The disk units have nonremovable media. Up to eight disk units may be connected in a string to a single CYBER 840, 850, 860, or 990 ISI channel port.

Each disk unit in the $887_1 must be defined as an element in the mainframe's physical configuration; the controllers are not defined as elements.

The $887_1 is not used as a deadstart device (that is, CIP cannot be installed on an $887_1). However, the $887_1 is used as the NOS/VE system device.

If you are running a NOS dual-state system, NOS and NOS/VE cannot share a string of $887_1 disk units, nor can one $887_1 be connected to both a NOS string and a NOS/VE string.

If an $887_1 is connected to two ports of the same channel, only one port (the first one described in the IOU connection) is used. If a $887_1 is connected to two different channels, both channels are used.

The $887_1 requires a page size of 4,096 bytes or greater at the current version of NOS/VE.

The $887_1 is defect-free at shipment and has no flaw map for use by the operating system.

WARNING

Do not configure NOS peripherals on the other port of a CIO channel to which you have configured NOS/VE peripherals.

$895_2 Disk Unit

The $895_2 product group consists of four nonremovable disk units. Each of the four disk units must be defined as an element (by the PCU subcommand DEFINE_ELEMENT) in the mainframe physical configuration.

One $895_2 product is packaged as part of the 895-1 product. In addition to the $895_2 product, the 895-1 includes two head-of-string controllers (HSCs). The two HSCs offer built-in redundancy, since each disk unit is accessible from either HSC. The HSCs can also transfer data concurrently, each to or from a different disk unit. The 895-1 is not defined as an element in the mainframe physical configuration.

Up to three additional $895_2 products (a total of 16 disk units) can be connected to an 895-1. To obtain the NOS/VE unit address (which must be specified on the PERIPHERAL_CONNECTION parameter of the DEFINE_ELEMENT subcommand) for each disk unit, multiply the address (0 or 1) of the 895-1 by 16 and add the result to the hardware unit number of the disk unit.

Up to two 895-1 products can be connected in a string (daisy chain) to the same 7165-2x controller. The address of the 895-1 can be either 0 or 1. The two HSCs in the 895-1 must have the same address and must be connected to the same 7165-2x.

The $895_2 requires a page size of 4,096 bytes or greater at NOS/VE Version 1.3.1.

The $895_2 is defect-free at shipment and has no flaw map for use by the operating system.

$9639_1 Tape Unit

The $9639_1 product group consists of 9-track tape units. This product supports either 25 or 75 ips operation and provides either 1600 cpi (PE) or 6250 cpi (GCR) recording.

One or two $9639_1 tape units may be configured on each CYBER 930-series ICI channel. Only one $7221_11 controller may be connected to a $9639_1 tape unit. The address of the $9639_1 may be in the range 0 to 1.

With one PP assigned per channel, the 7221 can transfer blocks of any size. See Maximum Tape Block Length for more information about long block transfers.

$9836_1 Disk Unit

The $9836_1 product group consists of nonremovable disk units.

The $9836_1 disk unit may be physically connected to two $FA7B5_A controllers.

The address of the $9836_1 can be in the range 0 to 7.

The $9836_1 is defect-free at shipment and has no flaw map for use by the operating system.

$9853_x Disk Unit

The $9853_x product group consists of nonremovable disk units.

This group consists of the following products:

- $9853_1: one $FA7B5_A controller and two disk units
- $9853_2: one $FA7B5_A controller and two disk units in a cabinet
- $9853_3: two $FA7B5_A controllers and four disk units in a cabinet
- $9853_4: no $FA7B5_A controllers and four disk units in a cabinet

Up to eight $FA7B5_A controllers can be connected to the same channel, and as many as eight $9853_x disk units can be connected to the same controller. A maximum of 64 disk units can be connected in any combination to the two ports. On CYBER 930-series systems, both $9836_1 and $9853_x units can be configured to the same channel, although not on the same controller.

On an I4 IOU, a $9853_x disk unit cannot serve as a CIP device. It can serve as the NOS/VE system device.

The address of the $9853_x can be in the range 0 to 7.

The $9853_x is defect-free at shipment and has no flaw map for use by the operating system.

WARNING

Do not configure NOS peripherals on the other port of a CIO channel to which you have configured NOS/VE peripherals.

$FA7B4_D Disk Controller

The $FA7B4_D product group consists of disk controllers (control modules). This product supports two nonconcurrent channel accesses. The $FA7B4_D product group supports only the $836_xxx disk unit.

The $FA7B4_D may be connected to another $FA7B4_D in a daisy-chain arrangement. Up to eight $FA7B4_D control modules can be connected to the same cable. $FA7B4_D control modules are numbered from 0 to 7 in a string. The same control module cannot be connected twice to the same cable; however, it does support connection to two different daisy chains. NOS/VE does not support intermixed $FA7B4_D and $10395_11 devices in the same daisy-chain configuration.

An $FA7B4_D may be connected to one or two host channels; however, the two channels cannot use the $FA7B4_D concurrently or alternately. The second connection is only for redundancy in case of ICI channel failure. The first IOU connection in the ON state is chosen as the active connection. If the $FA7B4_D is connected to two daisy chains, it must have the same address in both chains.

A $FA7B4_D controller may not be connected to the same channel with a $10935_11 controller.

The $FA7B4_D may be connected by an ISI cable to one or two 7255-1 channel adapters. The 7255-1 adapts an (internal) CYBER ICI channel to an ISI interface.

The $FA7B4_D may be connected to only one $836_xxx disk unit. For performance reasons, we recommend that the disk unit be connected to the $FA7B4_D as address 0; however, the addresses from 0 to 3 can be used.

On CYBER model 810 and 830 mainframes, only channels 0, 1, 3, 6, 16, and 22 can be connected to a $FA7B4_D control module.

$FA7B5_A Disk Controller

The $FA7B5_A product group consists of disk controllers (control modules) for the $9836_1 and $9853_x disk units. However, the same controller cannot simultaneously support both $9836_1 and $9853_x disk units. The $FA7B5_A supports two non-concurrent channel accesses to a CYBER 930-series mainframe.

The $FA7B5_A may be connected to another $FA7B5_A in a daisy-chain arrangement. Up to eight $FA7B5_A control modules may be connected to the same port of an IPI channel. Because the channel has two ports, a total of 16 control modules can be connected to the same channel. The $FA7B5_A control modules are numbered from 0 to 7 in a string. The same control module cannot be connected twice to the same cable; however, a control module can be connected to two different daisy chains.

A $FA7B5_A may be connected to one or two host channels; however, the two channels cannot use the $FA7B5_A concurrently or alternately. The second connection is purely for redundancy in case of a channel failure. The first IOU connection in the ON state is chosen as the active connection. If the $FA7B5_A is connected to two daisy chains, it must have the same address in both chains.

Managing Field Changes F

Managing Field Changes F

Overview

This appendix describes the operation of the MANAGE_FIELD_CHANGES (MANFC) utility. This utility extends the current NOS/VE priority PSR support process by providing sites with a method of installing, withdrawing, and tracking corrective code associated with priority PSRs.

The priority PSR support process provides packaging of corrective code in a form designed for transfer between the central support site and remote customer sites via electronic means. This process is referred to as Electronic QCU Support. QCU is an acronym for quick corrective update, a term used to describe a special container of code. A tape copy of the PSR correction can be provided to customer sites that are not equipped for any type of electronic transfer. The MANAGE_FIELD_CHANGES utility accepts either medium.

A site can manage corrections to the NOSVE_MAINTENANCE subproduct only, which, basically, are those pieces contained on the NOS/VE deadstart tape. Where restrictions apply, the utility returns informative messages and describes the available options.

Comments and criticisms with respect to the usability, user friendliness, and general operation of the MANAGE_FIELD_CHANGES utility are encouraged. Please direct correspondence to:

Control Data
Cyber Software Support, ARH213
4201 North Lexington Avenue
St. Paul, Minnesota 55126-6198
USA

PSRs against the MANAGE_FIELD_CHANGES utility will be given attention in accordance with the standard Control Data PSR support policy.

Obtaining Corrective Code Packets

The MANAGE_FIELD_CHANGES utility is used to manipulate a packet of corrective code (called a QCU) that has been retrieved from Control Data via some electronic means (network, dial-up, and so on).

Corrective code packets contain code generated in response to a priority PSR and are identified by the PSR identifier (for example, NV05915 is the name of the corrective code packet for PSR NV05915).

Corrective code packets reside in a special catalog on the Customer Introduction System at the Minneapolis Data Center. This system can be accessed via the CDCNET connection SYSTEMB or as a dialup access via rotary connection 612-858-2033. In addition, there is TELNET access via entry 61246, or sites may enter through Redi-Access if they hold a proper validation.

The network path name is SYSTEMB, the user/password is SUPPORT/REMOTE, and the family is the default.

In addition to code packets, a file titled HISTORY resides in this user catalog. This is a text file describing the correction packets that are available. The HISTORY file indicates the base system against which each correction was constructed and the approximate time required to transfer the packet at a 1200 baud rate.

Code packets can be moved from the Arden Hills system to the customer's system in whatever way is most convenient, provided the file attributes of the transferred file are RT=V, FC=FILE_BACKUP and FS=DATA. If you try to install a correction from a file with incorrect attributes, the MANAGE_FIELD_CHANGES utility returns an informative message. You must reload the correction file with the correct attributes before you continue. Figure F-1 shows the steps required to transfer a file from the host to a personal computer and then to your local system.

To download a corrective code packet from Control Data to your personal computer using CONNECT:

```
Log in to the Customer Introduction System

Press Alt-F10  (Return to CONNECT main menu)

Press 4        (Receive file from host)

Enter micro file name:  NV06754   (Your choice of file name)

Enter host file name:  :DSB.SERVICE.NV06754   (Name of the correction)

Enter transfer type:   2     (Binary transfer)
```

When the file transfer is complete, select option 2 to return to terminal mode.

To upload a corrective code packet from your personal computer to your host using CONNECT:

```
Log in to your system.

Enter: SET_FILE_ATTRIBUTES $USER.NV06754 ..   (Your choice of file name)
                            RT=VARIABLE ..
                            FS=DATA ..
                            FC=FILE_BACKUP
```

NOTE

You must set the file attributes of the file as shown. If the file attributes of the corrective code packet are not correct, MANFC generates an error message when you try to install the correction.

```
Press Alt-F10   (Return to CONNECT main menu)

Press 3         (Send file)

Enter micro file name:  NV06754   (Name used to download file)

Enter host file name:  $USER.NV06754  (File name used on SETFA)

Enter transfer type:   2     (Binary transfer)
```

When the file transfer is complete, select option 2 to return to terminal mode.

Figure F-1. File Transfer Example Using CONNECT

Processing Correction Packets

The MANAGE_FIELD_CHANGES utility is used to manipulate a correction packet once it is available locally. MANAGE_FIELD_CHANGES manages correction packets and produces a correction system by applying the correction packet to the base system.

Table F-1 lists the MANAGE_FIELD_CHANGES subcommands.

Table F-1. MANFC Subcommands

Subcommand	Function
INSTALL_FIELD_CORRECTION	Adds a PSR correction to the current system.
WITHDRAW_FIELD_CORRECTION	Removes a PSR correction from the current system.
GENERATE_CORRECTION_ SYSTEM	Generates a deadstart catalog that contains all of the installed corrections.
ESTABLISH_CORRECTION_ SYSTEM	Moves the contents of the deadstart catalog to the disk file from which the system will perform the next disk deadstart.
COMMIT_CORRECTION_SYSTEM	Sets the flag indicating a disk deadstart file is available.
WITHDRAW_CORRECTION_ SYSTEM	Backs out the current correction system and reinstate the previous base system to the deadstart disk file.
GENERATE_DEADSTART_TAPE	Creates a correction system deadstart tape.
DISPLAY_CORRECTION	Displays information about a particular correction packet.
DISPLAY_SCOREBOARD	Displays information about the state of system corrections.
QUIT	Exits the utility.

Using the Utility

The MANAGE_FIELD_CHANGES utility can be run from the system console or from terminals other than the system console, provided the user has the SYSTEM_ ADMINISTRATION capability. Exception: two commands, COMMIT_CORRECTION_ SYSTEM and WITHDRAW_CORRECTION_SYSTEM, must be executed from the system console.

NOTE

If you are running the MANAGE_FIELD_CHANGES utility from a user terminal, you must issue the following command before executing MANAGE_FIELD_CHANGES:

```
crecle
$system.osf$builtin_library
```

At the console or terminal, enter:

```
manage_field_changes
```

It is assumed at this point that the desired correction packet has been obtained, either electronically or as a correction tape (not as a deadstart tape), and is available as a file on the system.

It is not necessary to perform all the steps for a particular correction packet installation in one session. Once an INSTALL_FIELD_CORRECTION or WITHDRAW_FIELD_CORRECTION subcommand has been completed, the correction is known to the system. You can exit the MANFC utility and enter it at a later time to complete the installation.

Installing a Correction

The sequence of commands for installing a correction is as follows:

1. Add the PSR correction to the current system with the INSTALL_FIELD_CORRECTION subcommand.
2. Generate a deadstart catalog containing the installed corrections with the GENERATE_CORRECTION_SYSTEM subcommand.
3. (Optional) Create a correction system deadstart tape with the GENERATE_DEADSTART_TAPE subcommand.
4. With the ESTABLISH_CORRECTION_SYSTEM subcommand, move the contents of the deadstart catalog to the disk file from which the system will perform the next disk deadstart.
5. With the COMMIT_CORRECTION_SYSTEM subcommand, set the flag indicating that a disk deadstart file is available.

Withdrawing a Correction

The sequence for withdrawing a correction is as follows:

1. Remove the PSR correction from the current system with the WITHDRAW_FIELD_CORRECTION subcommand.
2. With the GENERATE_CORRECTION_SYSTEM subcommand, generate a deadstart catalog containing the installed corrections prior to the one withdrawn.
3. (Optional) Create a correction system deadstart tape with the GENERATE_DEADSTART_TAPE subcommand.
4. With the ESTABLISH_CORRECTION_SYSTEM subcommand, move the contents of the deadstart catalog to the disk file from which the system will perform the next disk deadstart.
5. With the COMMIT_CORRECTION_SYSTEM subcommand, set the flag indicating that a disk deadstart file is available.

Withdrawing a Correction System

Use the following subcommand for withdrawing a correction system:

WITHDRAW_CORRECTION_SYSTEM

Additional Subcommands

The following subcommands can be issued at any time in the process:

DISPLAY_CORRECTION

DISPLAY_SCOREBOARD

QUIT

Activating the New System

A TERMINATE_SYSTEM command followed by a disk deadstart is required to activate a correction system, unless the deadstart tape mechanism is used. The recommended process for installing a correction system is to complete the correction installation steps through the GENERATE_CORRECTION_SYSTEM subcommand, issue the GENERATE_DEADSTART_TAPE command to produce a deadstart tape, then deadstart the correction system to ensure that the problem has been corrected. Once the system has been verified, the ESTABLISH_CORRECTION_SYSTEM and COMMIT_CORRECTION_SYSTEM subcommands can be issued. When the COMMIT_CORRECTION_SYSTEM command is processed, the deadstart file catalog associated with the corrections is deleted to conserve space. Then it is no longer possible to generate a deadstart tape.

The MANAGE_FIELD_CHANGES utility assigns a new system identifier for each installed correction. Identifiers are assigned in the format: LxxxCn, where *xxx* indicates the PSR level of the NOS/VE version against which the correction was issued, *C* is an identifier indicating that this as a correction system, and *n* is a letter in the range A through Y. The *n* value increases monotonically each time a correction is installed. There is currently a limit of 25 corrections per system before correction cleanup is required.

The MANAGE_FIELD_CHANGES utility allows only one correction packet per system library at one time. For example, only one correction packet (PSR) that modifies the OSF$BOUND_JOB_TEMPLATE_23D library can be installed at any one time. If you attempt to install a subsequent PSR that modifies the same system library, a message is issued informing you that in order to install the subsequent PSR, the PSR that originally modified the library must be withdrawn.

The MANAGE_FIELD_CHANGES utility and its associated process add two new catalogs to the $SYSTEM environment. The primary correction information is contained in the FIELD_MAINTENANCE subcatalog. After the correction installation process is completed, information pertaining to building a new system is contained in the QCU_MAINTENANCE subcatalog. The latter catalog structure parallels the NOSVE_MAINTENANCE catalog structure and is used to avoid modifying the image of the the base system previously installed. The base system can be reinstated at any time by issuing one of the following command sequences:

```
/manage_field_changes
MANFC/withdraw_correction_system
```

or

```
/maintain_deadstart_software
MAIDS/establish_disk_based_system ..
MAIDS../dc=$system.nosve_maintenance.deadstart_catalog
MAIDS/commit_new_system
MAIDS/quit
```

After entering this sequence of commands, enter a TERMINATE_SYSTEM command, then deadstart the system.

The WITHDRAW_CORRECTION_SYSTEM subcommand of MANAGE_FIELD_CHANGES is a complete command, being a combination establish and commit. It is only necessary to terminate the system and deadstart to activate the reinstated system once this command has been issued.

Displaying Installed Corrections

The command DISPLAY_SCOREBOARD provides a summary of corrections and their associated attributes. The display is self explanatory, except for the entry titled *active*. This entry indicates whether or not the system containing the correction has yet been written to the secondary deadstart file as directed by the ESTABLISH_CORRECTION_SYSTEM command.

NOTE

Remember that once the ESTABLISH_CORRECTION_SYSTEM command has been issued, the correction is available but is not yet active. You can activate the correction either by deadstarting from the deadstart tape or by issuing a COMMIT_CORRECTION_SYSTEM command, followed by a successful TERMINATE_SYSTEM command and a deadstart.

Additional Notes

- If the file PROLOG_FILE exists in the SITE_OS_MAINTENANCE subcatalog before using MANAGE_FIELD_CHANGES, purge the file before executing MANAGE_FIELD_CHANGES functions. Otherwise, some error messages pertaining to LCU files that already exist may appear.
- Installation of a BCU or release level system automatically resets the MANAGE_FIELD_CHANGES environment and erases all correction information.
- Once a correction has been taken to the COMMIT_CORRECTION_SYSTEM level, the QCU_MAINTENANCE catalog is deleted except for the link maps and system debug tables.

- When providing support materials for a system dump taken while a correction system is running, use the following procedure to provide Control Data with the link maps and debug tables associated with your system:

  ```
  /maintain_deadstart_software
  MAIDS/collect_dump_materials ..
  MAIDS../loc = $system.qcu_maintenance.level.link_output_files..
  MAIDS../evsn = nnn
  MAIDS/quit
  ```

  where *level* is the level of the current correction system (for example, L739CB) and *nnn* is the VSN of the tape on which to copy the information.

- If you use the ANALYZE_DUMP utility at your site, using the DT=RS option to define the system debug table produces erroneous results, since the option selects the base system tables. The correct value for this parameter is the following:

  ```
  dt= ..
     $system.qcu_maintenance.level.link_output_files.system_debug_table
  ```

  where *level* is the correction system identifier (for example, L739CB).

- The catalogs associated with the MANFC process require a maximum of 12 megabytes during processing and shrink to about 4 megabytes after a correction system has been committed. The space requirement is a function of the number of corrections in place; 12 megabytes is the maximum size for a full set of corrected system libraries. Correction packets are minimal; most packets are 8,000 bytes or smaller.

- Certain MANAGE_FIELD_CHANGES subcommands parallel subcommands of the MAINTAIN_DEADSTART_SOFTWARE (MAIDS) utility. Most notable are the MANFC GENERATE_DEADSTART_TAPE subcommand and the CREATE_VE_DEADSTART_TAPE subcommand of MAIDS. Even though similar, the MANFC utility requires the use of the GENERATE_DEADSTART_TAPE subcommand to produce a proper tape image of a correction system. Use of the wrong utility could result in the loss of site modifications and an improper system level number. For the same reasons, the correction system must be committed using the COMMIT_CORRECTION_SYSTEM subcommand of MANFC, as opposed to the COMMIT_NEW_SYSTEM subcommand of MAIDS.

  In general, it is best to stay inside the MANAGE_FIELD_CHANGES environment for operations involving a correction system. The MANAGE_FIELD_CHANGES subcommands should not be used for base system work, since these subcommands key off the field maintenance catalog rather than the base system catalog. For base system work, we recommend using the MAINTAIN_DEADSTART_SOFTWARE environment.

- At present, the MANAGE_FIELD_CHANGES utility supports only the NOSVE_MAINTENANCE subproduct of the NOS/VE operating system.

MANAGE_FIELD_CHANGES Command and Subcommands

As stated earlier, the MANAGE_FIELD_CHANGES utility and all subcommands (except WITHDRAW_CORRECTION_SYSTEM and COMMIT_CORRECTION_SYSTEM) can be initiated from any terminal, provided the user has the SYSTEM_ ADMINISTRATION capability. The WITHDRAW_CORRECTION_SYSTEM and COMMIT_CORRECTION_ SYSTEM subcommands execute successfully only from the system console.

The MANAGE_FIELD_CHANGES utility resides in the $SYSTEM.OSF$BUILTIN_ LIBRARY.

Following are descriptions of the MANAGE_FIELD_CHANGES subcommands, listed alphabetically. Although it is not shown, all commands include a status variable. However, returned status values are never seen, since the utility captures status situations and interprets them into user option displays.

MANAGE_FIELD_CHANGES Command

Purpose	Initiates the MANAGE_FIELD_CHANGES utility.
Format	**MANAGE_FIELD_CHANGES** or **MANFC**
Parameters	None.

COMMIT_CORRECTION_SYSTEM Subcommand

Purpose	Activates the disk deadstart file image.
Format	COMMIT_CORRECTION_SYSTEM or COMCS
Parameters	None.

DISPLAY_CORRECTION Subcommand

Purpose Displays correction packet information.

Format **DISPLAY_CORRECTION** or
DISC
  CORRECTION_IDENTIFIER=name
  OUTPUT=file name

Parameters **CORRECTION_IDENTIFIER** or **CI**

Specifies the PSR identifier associated with the packet for which you want information to be displayed. This is a required parameter.

OUTPUT or *O*

Specifies the destination file for the output. The default is the display screen.

Examples

```
DISC nv05712
```

DISPLAY_SCOREBOARD Subcommand

Purpose Displays information about the system's correction state.

Format **DISPLAY_SCOREBOARD** or
DISSB
OUTPUT=file name

Parameters *OUTPUT* or *O*

Specifies the destination file for the output. The default is the display screen.

ESTABLISH_CORRECTION_SYSTEM Subcommand

Purpose	Moves the deadstart catalog to the disk file image.
Format	ESTABLISH_CORRECTION_SYSTEM or ESTCS
Parameters	None.

GENERATE_CORRECTION_SYSTEM Subcommand

Purpose Generates a correction system deadstart catalog.

Format **GENERATE_CORRECTION_SYSTEM** or **GENCS**

Parameters None.

GENERATE_DEADSTART_TAPE Subcommand

Purpose Creates a correction system deadstart tape.

Format **GENERATE_DEADSTART_TAPE** or
GENDT
  EXTERNAL_VSN=string
  RECORDED_VSN=string
  TYPE=keyword

Parameters **EXTERNAL_VSN** or **EV**

Specifies the external VSN of the deadstart tape. This is a required parameter.

RECORDED_VSN or **RV**

Specifies the recorded VSN of the deadstart tape. This is a required parameter.

TYPE or *T*

Specifies the deadstart tape type. You can specify one of the following keywords; the default is the system default type:

MT9$1600
MT9$6250
MT9$28000

Examples

```
GENDT ev='abc001' rv='def002' t=mt9$6250
```

INSTALL_FIELD_CORRECTION Subcommand

Purpose Installs a field correction.

Format INSTALL_FIELD_CORRECTION or
INSFC
  CORRECTION_IDENTIFIER = name
  CORRECTION_FILE_PATH = pathname

Parameters CORRECTION_IDENTIFIER or CI

Specifies the PSR identifier associated with the correction you want to add to the system. This parameter is required.

CORRECTION_FILE_PATH or CFP

Specifies the path name to the correction packet file. This parameter is required.

Examples

```
INSFC nv05712  $user.nv05712
```

QUIT Subcommand

Purpose	Exits the MANAGE_FIELD_CHANGES utility.
Format	**QUIT** or **QUI**
Parameters	None.

WITHDRAW_CORRECTION_SYSTEM Subcommand

Purpose	Withdraws the installed correction system and backs up to the previously installed base.
Format	**WITHDRAW_CORRECTION_SYSTEM** or **WITCS**
Parameters	None.

WITHDRAW_FIELD_CORRECTION Subcommand

Purpose Withdraws a previously installed field correction.

Format WITHDRAW_FIELD_CORRECTION or
WITFC
  CORRECTION_IDENTIFIER=name

Parameters CORRECTION_IDENTIFIER or CI

Specifies the PSR identifier associated with the packet you want to remove from the system. This is a required parameter.

Examples

```
WITFC nv05712
```

Index

Comments (continued from other side)

Please fold on dotted line;
seal edges with tape only.

FO

FOLD

FO

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 System Performance and Maintenance, Volume 2 — 60463925 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?	How do you use this manual?
☐ Manager	☐ As an overview
☐ Systems analyst or programmer	☐ To learn the product or system
☐ Applications programmer	☐ For comprehensive reference
☐ Operator	☐ For quick look-up
☐ Other ______	☐ 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.

Command and Subcommand Quick Index

This index lists the page numbers for NOS/VE commands, subcommands, and functions described in this manual.

NOS/VE commands, subcommands, and functions found in the NOS/VE System Performance and Maintenance manual, Volume 1, are followed by I. Those found in the NOS/VE Operations manual are followed by O.

Subcommands are indicated by the term *subc*. Functions are indicated by the term *func*.

CONTROL DATA