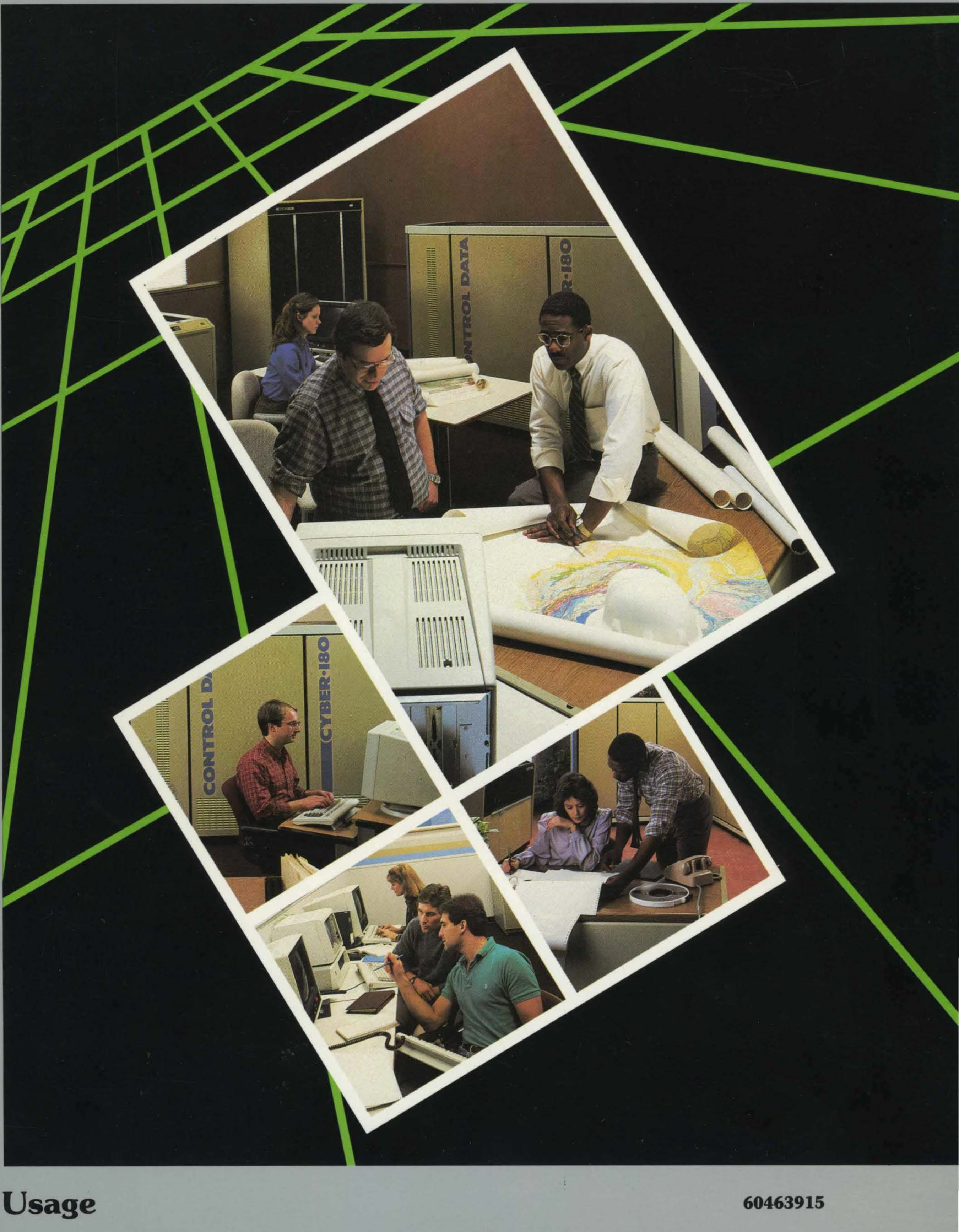
NOS/VE
System Performance and Maintenance
Volume 1: Performance

CONTROL DATA



Usage

60463915

NOS/VE System Performance and Maintenance
Volume 1: Performance

Usage

Publication Number 60463915

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:

- New system attributes:
  ACCOUNT_LOG_CRITICAL
  ACCOUNT_LOG_MAXIMUM_SIZE
  AIO_LIMIT
  ENABLE_CONSOLE_BELL
  ENGINEERING_LOG_CRITICAL
  ENGINEERING_LOG_MAXIMUM_SIZE
  FILE_SERVER_DEBUG_ENABLED
  FILE_SERVER_RECOVERY_ENABLED
  HISTORY_LOG_CRITICAL
  HISTORY_LOG_MAXIMUM_SIZE
  MAXWS_AIO_THRESHOLD
  SECURITY_LOG_CRITICAL
  SECURITY_LOG_MAXIMUM_SIZE
  SPACE_MESSAGES_TO_CONSOLE
  STATISTIC_LOG_CRITICAL
  STATISTIC_LOG_MAXIMUM_SIZE
  SYSTEM_LOG_CRITICAL
  SYSTEM_LOG_MAXIMUM_SIZE
  UNLOAD_DEADSTART_TAPE

- Removal of MAXIMUM_SEGMENT_LENGTH system attribute.

- New chapter 3, Managing Memory, describes memory attributes and includes an overview of page streaming.

- Modification of MANAGE_PERIODIC_STATISTICS utility to include information about an immediate emission set.
- New PROCESS_STORAGE command replaces EMIT_PERMANENT_FILE_ STATISTICS command.
- New ANALYZE_BINARY_LOG utility subcommands:

  CHANGE_DEFAULTS
  DISPLAY_DEFAULTS
  POP_PAGE_HEADER
  PUSH_PAGE_HEADER
  PUT_NEW_PAGE

- New statistics:

OD10	Optical disk accounting
OS7	Service class usage
OS8	Job class usage

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 NOS/VE 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.

NOS/VE 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 Anaysis 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: System Performance

Adjusting System Attributes 2

Adjusting System Attributes 2

System attributes enable you to define performance parameters and to select system options, such as operational modes and table and file sizes. In this chapter, system attributes are treated in the following manner:

- The DISPLAY_SYSTEM_ATTRIBUTE and SET_SYSTEM_ATTRIBUTE commands are described. Use these commands to display the current values of system attributes and to alter those values, respectively.
- The attributes are listed in system attribute tables that include their names, default values, range of allowable values, and other information about when and how to alter their values.
- Each system attribute is described. The description may include information about how the attribute affects NOS/VE execution, as well as how it relates to other attributes or to other system limits or values.

System attributes are discussed elsewhere in this manual in connection with CPU scheduling (chapter 5), page aging (chapter 6), and handling bottlenecks in the system (chapter 8).

Commands

The SET_SYSTEM_ATTRIBUTE command lets you alter the value of a system attribute. The DISPLAY_SYSTEM_ATTRIBUTE command displays the current value of a system attribute. These commands are part of the group of commands known as system core commands. You can interactively enter system core commands when deadstart pauses with the following prompt:

```
Enter system core commands
```

The system core commands, including SET_SYSTEM ATTRIBUTE, can also be entered automatically during deadstart by placing them in the deadstart file. For information about how to do this, see the description of the DCFILE file in chapter 5 of the NOS/VE System Performance and Maintenance manual, Volume 2.

Finally, you can enter the SET_SYSTEM_ATTRIBUTE and DISPLAY_SYSTEM_ATTRIBUTE commands during normal system operation. System core commands other than SET_SYSTEM_ATTRIBUTE and DISPLAY_SYSTEM_ATTRIBUTE are described in the NOS/VE System Performance and Maintenance manual, Volume 2.

DISPLAY_SYSTEM_ATTRIBUTE

Purpose Displays the current value of a system attribute.

Format

DISPLAY_SYSTEM_ATTRIBUTE or
DISSA
    attribute

Parameters

attribute

Specifies which system attribute is to be displayed. Refer to tables 2-1, Performance Attributes, and 2-2, Maintenance Attributes, for more information about attribute names, default values, and ranges of acceptable values. This parameter is required.

Remarks

- You can enter this command either during deadstart (when the system prompts you to enter system core commands) or at any time after deadstart.

- The use of this command requires that the SYSTEM_DISPLAYS capability be active within the System Operator Utility session. The System Operator Utility is described in the NOS/VE Operations manual.

SET_SYSTEM_ATTRIBUTE

Purpose Alters the value of a system attribute.

Format **SET_SYSTEM_ATTRIBUTE** or
SETSA
  name
  value

Parameters **name**

Specifies the name of the system attribute whose value is to be altered. Refer to tables 2-1, Performance Attributes, and 2-2, Maintenance Attributes, for information about attribute names, default values, and ranges of acceptable values. This parameter is required. There are no abbreviations for the system attribute names.

value

Specifies the new value for the system attribute. The value must be an integer. However, some attributes require a boolean value, in which case you must specify a 0 for a FALSE value or a 1 for a TRUE value. In tables 2-1 and 2-2, the *Type* column indicates whether an integer or boolean value is required for each attribute. This parameter is required.

Remarks

- Some attributes can be set only during deadstart; others can be set at any time. Tables 2-1 and 2-2 indicate when attributes can be set.
- Except where stated otherwise in the attribute descriptions, values specified for system attributes are not preserved across continuation deadstarts.
- The use of this command requires that the CONFIGURATION_ADMINISTRATION capability be active within the System Operator Utility session. The System Operator Utility is described in the NOS/VE Operations manual.

System Attribute Tables

Tables 2-1 and 2-2 provide summary information about NOS/VE attributes. Two types of attributes are provided: performance attributes and maintenance attributes. Table 2-1 lists the performance attributes used to tune the system for optimum performance. Table 2-2 lists the maintenance attributes used for purposes other than performance tuning.

In tables 2-1 and 2-2, the *Type* and *When to Enter* columns require some explanation:

Type

Two types exist:

I Integer type

B Boolean type (0 is FALSE, 1 is TRUE)

When to Enter

Indicates when you can specify the attribute:

D You can set the attribute only at deadstart, when the DCFILE deck of the deadstart file is executed, or when the system prompts the operator to enter system core commands.

A You can set the attribute either at deadstart or any time during NOS/VE operations.

Table of Performance Attributes

Table 2-1 lists the system attributes used for performance tuning. Use these attributes to tune your system according to the needs of your particular system configuration and workload.

Table 2-1. Performance Attributes

Attribute Name	Type	When to Enter	Default Value	Lowest Value	Highest Value
AIO_LIMIT	I	A	60,000	0	5,000,000
CHECK_IDLE_ DISPATCHING_ INTERVAL	I	A	15,000,000 microseconds	1,000	1,000,000,000
DEDICATE_A_ CPU_TO_NOS	B	A	0	0	1
DISDELAY	I	A	1,000 milliseconds	1	100,000
LONG_WAIT_ FORCE_SWAP_ TIME	I	A	6,000,000 microseconds	0	1,000,000,000
LONG_WAIT_ SWAP_TIME	I	A	6,000,000 microseconds	0	1,000,000,000
MAX_TIME_ SWAP_IO_NOT_ INIT	I	A	100,000,000 microseconds	0	86,400,000,000
MAXIMUM_ ACTIVE_JOBS	I	D	100	1	250
MAXIMUM_JOB_ CLASSES	I	A	10	4	255
MAXIMUM_ KNOWN_JOBS	I	A	250	1	65,535
MAXIMUM_ SERVICE_ CLASSES	I	A	10	4	255
MAXIMUM_ SWAP_ RESIDENT_TIME	I	A	200,000,000 microseconds	0	86,400,000,000

(Continued)

Table 2-1. Performance Attributes *(Continued)*

Attribute Name	Type	When to Enter	Default Value	Lowest Value	Highest Value
MAXIMUM_ THINK_TIME	I	A	60,000,000 microseconds	0	1,000,000,000
MAXWS_AIO_ THRESHOLD	I	A	10	0	100,000
MINIMUM_ SHARED_ WORKING_SET	I	A	0 pages	0	100,000
MINIMUM_ THINK_TIME	I	A	500,000 microseconds	0	1,000,000,000
READ_TU_ EXECUTE	I	A	1	1	32
READ_TU_ READ_WRITE	I	A	1	1	32
SWAP_FILE_ ALLOCATION_ SIZE	I	A	262,144	4,096	1,000,000
SWAP_JOBS_IN_ LONG_WAIT	B	A	1	0	1
THINK_ EXPIRATION_ TIME	I	A	15,000,000 microseconds	0	1,000,000,000

Table of Maintenance Attributes

Table 2-2 lists the system attributes used for purposes other than performance tuning. These are called maintenance attributes. Generally, the maintenance attributes fall into the following categories: attributes used for system debugging, and attributes that perform a specific function, such as NETWORK_ACTIVATION.

Table 2-2. Maintenance Attributes

Attribute Name	Type	When to Enter	Default Value	Lowest Value	Highest Value
ABORTED_TASK_ THRESHOLD	I	A	20 tasks	0	2**47–1
ACCOUNT_LOG_ CRITICAL	B	D	0	0	1
ACCOUNT_LOG_ MAXIMUM_SIZE	I	D	150 megabytes	1	150
AUTOMATIC_ UNSTEP_RESUME	B	A	0	0	1
COMMAND_ STATISTICS_ ENABLED	B	A	0	0	1
CONSTRAIN_ MEAPE_ SEGMENTS	B	A	0	0	1
DELETE_ UNRECONCILED_ FILES	B	D	0	0	1
DUMP_WHEN_ DEBUG	B	A	0	0	1
ENABLE_ CONSOLE_BELL	B	D	1	0	1
ENGINEERING_ LOG_CRITICAL	B	D	0	0	1
ENGINEERING_ LOG_MAXIMUM_ SIZE	I	D	150 megabytes	1	150
FILE_SERVER_ DEBUG_ENABLED	B	A	0	0	1
FILE_SERVER_ RECOVERY_ ENABLED	B	A	0	0	1

(Continued)

Table 2-2. Maintenance Attributes *(Continued)*

Attribute Name	Type	When to Enter	Default Value	Lowest Value	Highest Value
HALT_ON_ HUNG_TASK	B	A	0	0	1
HALT_ON_ PROCESSOR_ MALFUNCTION	B	A	0	0	1
HALT_ON_ SWAPIN_FAILURE	B	A	0	0	1
HALTRING	I	A	0	0	15
HISTORY_LOG_ CRITICAL	B	D	0	0	1
HISTORY_LOG_ MAXIMUM_SIZE	I	D	150 megabytes	1	150
IGNORE_IMAGE	B	D	0	0	1
JOB_RECOVERY_ OPTION	I	D	0	0[1]	3[1]
MANDATORY_ DUALSTATE	B	A	1[2]	0	1
MAXIMUM_ ALLOCATION_ SIZE	I	A	262,144 bytes	16,384	2,147,483,647
MAXIMUM_ OUTPUT_FILES	I	A	150	1	65,535
NETWORK_ ACTIVATION	B	D	1	0	1
PROCESSOR_ DUE_THRESHOLD	I	A	10 errors	0	2**47–1
RECOVER_AT_ ALL_COSTS	B	D	0	0	1
SECURITY_LOG_ CRITICAL	B	D	0	0	1

1. The only values you can specify for this attribute are 0 and 3.

2. This attribute applies only to dual-state mode; it has no effect on a NOS/VE standalone system.

(Continued)

Table 2-2. Maintenance Attributes *(Continued)*

Attribute Name	Type	When to Enter	Default Value	Lowest Value	Highest Value
SECURITY_LOG_ MAXIMUM_SIZE	I	D	150 megabytes	1	150
SPACE_ MESSAGES_TO_ CONSOLE	B	A	1	0	1
STATISTIC_LOG_ CRITICAL	B	D	0	0	1
STATISTIC_LOG_ MAXIMUM_SIZE	I	D	150 megabytes	1	150
SYSTEM_ ACTIVATION	B	D	1	0	1
SYSTEM_DEBUG_ RING	I	A	0	0	15
SYSTEM_DEBUG_ SEGMENT	I	A	0	0	4,095
SYSTEM_ERROR_ HANG_COUNT	I	A	6	0	1,000
SYSTEM_LOG_ CRITICAL	B	D	0	0	1
SYSTEM_LOG_ MAXIMUM_SIZE	I	D	150 megabytes	1	150
UNLOAD_ DEADSTART_TAPE	B	D	1	0	1
VALIDATE_ ACTIVE_SETS	B	D	0	0	1

System Attribute Descriptions

The following sections describe the system attributes in alphabetical order.

ABORTED_TASK_THRESHOLD

Specifies the number of tasks that can abort in a given CPU before the system attempts to discontinue the use of that CPU in a multiple-CPU environment or attempts to execute a STEP_SYSTEM command in a single-CPU environment. Use this system attribute when detected uncorrected errors (DUEs) are encountered in NOS/VE job mode. See table 2-2 for the allowable values and default value for this system attribute.

ACCOUNT_LOG_CRITICAL

Specifies the action NOS/VE takes when it is unable to record an entry in the account log. If the value of this system attribute is 1, NOS/VE terminates any time the system is unable to record an entry. If the value of this system attribute is 0, NOS/VE discards the entry and informs the operator by way of a message in the critical display window. See table 2-2 for the default value for this system attribute.

ACCOUNT_LOG_MAXIMUM_SIZE

Specifies the maximum size, in megabytes, of the account log. See table 2-2 for the allowable values and default value for this system attribute.

AIO_LIMIT

Specifies a limit used to prevent a task from flooding the system with I/O requests. If a task causes a page to be read from disk when the task's job has reached the specified limit of active I/O requests, then the task is forced to give up the CPU (task switch). This allows another task to execute. If the new task needs to initiate I/O, it can do so because the I/O queues are not filled by the first task.

This system attribute is effective in controlling tasks that initiate numerous write I/O requests by modifying pages of a large file sequentially. The modified pages are removed from the working set and written to disk as new pages are added to the working set and modified. The result is that the job working set never reaches its maximum. See table 2-1 for the allowable values and default value for this system attribute.

NOTE

By default, the value for this sytem attribute is set to a large number. If you experience problems because of excessive I/O, we recommend setting the value to 30.

AUTOMATIC_UNSTEP_RESUME

Specifies automatic or manual mode of system resumption. When set to automatic, the system automatically resumes operations after an environmental condition has caused system suspension. When set to manual, resumption of operations is left up to the operator or analyst. For further information about system suspension and resumption, see the NOS/VE Operations manual.

If the value of this system attribute is 1, the system is set to automatic. If the value is 0, the system is set to manual. See table 2-2 for the default value for this system attribute.

CHECK_IDLE_DISPATCHING_INTERVAL

Specifies the interval at which the system checks whether any dispatching priority queues are idle (nondispatchable) and whether any previously idle dispatching queues are now dispatchable.

A dispatching priority queue is considered idle if the dispatcher has not tried to dispatch any tasks of that dispatching priority in the length of time defined by the IDLE_DISPATCHING_QUEUE_TIME scheduling attribute (described in chapter 4, Job Scheduling). The dispatcher does not attempt to dispatch tasks of a certain dispatching priority if tasks with a higher dispatching priority are CPU-bound. All jobs in an idle dispatching priority queue are swapped out.

If a dispatching priority queue was previously idle and is now dispatchable, jobs of that dispatching priority that were swapped out will be swapped back in.

The interval defined by this system attribute is checked within the NOS/VE memory manager, which is called at intervals established by the PERIODIC_CALL_INTERVAL memory attribute (described in chapter 3, Managing Memory). Accordingly, the value of the CHECK_IDLE_DISPATCHING_INTERVAL system attribute should always be greater than the value of the PERIODIC_CALL_INTERVAL memory attribute. See table 2-1 for the allowable values and default value for this system attribute.

COMMAND_STATISTICS_ENABLED

Specifies whether the command resources statistic (CL2) can be activated and collected. (The CL2 statistic is described in chapter 9, Statistics Facility.)

If the value of this system attribute is 1, the CL2 statistic can be collected; if its value is 0, the CL2 statistic is not collected. See table 2-2 for the default value for this system attribute.

NOTE

Activating the CL2 statistic generates a large amount of data; the resulting overhead can degrade system performance.

CONSTRAIN_MEAPE_SEGMENTS

Specifies whether to allow any MEASURE_PROGRAM_EXECUTION task to continue processing after it has filled one segment with collected data. The MEASURE_PROGRAM_EXECUTION task is documented in the NOS/VE Object Code Management manual.

If the value of this system attribute is 1, the MEASURE_PROGRAM_EXECUTION task aborts at the point where it requests its second segment. If the value of this system attribute is 0, the MEASURE_PROGRAM_EXECUTION task is allowed to open and write data to as many segments as are necessary for completion. A program being measured requires more than 3,000,000 calls to fill up one segment if the MAXIMUM_SEGMENT_LENGTH system attribute is 150,000,000. See table 2-2 for the default value for this system attribute.

DEDICATE_A_CPU_TO_NOS

Specifies whether CPU 0 (the NOS or NOS/BE CPU) of a dual-processor mainframe is to be dedicated to NOS or NOS/BE activity. This system attribute is valid only for dual-state systems, it has no effect on standalone systems.

If the value of this system attribute is 1, no NOS/VE jobs run on CPU 0. If the value of this system attribute is 0, NOS/VE jobs are allowed to run on CPU 0. See table 2-1 for the default value for this system attribute.

DELETE_UNRECONCILED_FILES

Specifies whether NOS/VE deletes catalogs and permanent file cycles that are known either to permanent file management or to device management, but not to both, during the permanent file recovery phase of a continuation deadstart. This system attribute applies to all mass storage sets that are active at deadstart time.

If the value of this system attribute is 1, NOS/VE deletes unreconciled catalogs and permanent file cycles during permanent file recovery. If the value is 0, catalogs and file cycles are not deleted as a result of being unreconciled.

If this system attribute is set to 1, it is possible that more disk space will be available following deadstart than if the attribute were set to 0. Set this system attribute to 1 only when all disks in the set are set to ON and can be accessed through channels and controllers that are also set to ON; otherwise files or catalogs may be lost. If you set the DELETE_UNRECONCILED_FILES system attribute to 1 when a catalog volume is missing, you delete all catalogs subordinate to the missing catalogs and all files described by the missing catalogs and deleted catalogs.

If you do a continuation deadstart following a system crash that did not recover the memory image (see the IGNORE_IMAGE system attribute), there may be catalogs and permanent files that are known to device management but not to permanent file mangement. This is more likely to happen if the system crashes while attempting to deadstart without recovering active jobs (see the JOB_RECOVERY_OPTION system attribute). If this situation occurs, you may want to delete these catalogs and permanent files by setting the DELETE_UNRECONCILED_FILES system attribute to 1 at the next continuation deadstart. See table 2-2 for the default value for this system attribute.

DISDELAY

Specifies how often the system refreshes the system console displays. See table 2-1 for the allowable values and default value for this system attribute.

DUMP_WHEN_DEBUG

Specifies whether the automatic dump command list is to be executed when the system core debugger is called into a task because of an error condition. If the value of this system attribute is 1, the automatic dump command list is executed. The NOS/VE monitor initiates the normal action for the fault when the dump completes. If the value of this system attribute is 0, no dump is initiated. See table 2-2 for the default value for this system attribute.

ENABLE_CONSOLE_BELL

Specifies whether to enable the bell at the system console. If the value of this system attribute is 1, NOS/VE enables the console bell. If the value of this system attribute is 0, NOS/VE turns the console bell off. See table 2-2 for the default value for this system attribute.

ENGINEERING_LOG_CRITICAL

Specifies the action NOS/VE takes when it is unable to record an entry in the engineering log. If the value of this system attribute is 1, NOS/VE terminates any time the system is unable to record an entry. If the value of this system attribute is 0, NOS/VE discards the entry and informs the operator by way of a message in the critical display window. See table 2-2 for the default value for this system attribute.

ENGINEERING_LOG_MAXIMUM_SIZE

Specifies the maximum size, in megabytes, of the engineering log. See table 2-2 for the allowable values and default value for this system attribute.

FILE_SERVER_DEBUG_ENABLED

Normally, this system attribute should be set to 0. It is used primarily as a debugging aid to test the file server. If the value of this system attribute is 1, detailed messages are displayed at the system console or written to the job log when significant or unusual events occur concerning the file server. If the value of this system attribute is 0, no file server related messages are displayed or logged. See table 2-2 for the default value for this system attribute.

NOTE

When this system attribute is set to 1, the NOS/VE monitor halts if file server related errors occur that are considered unrecoverable or critical.

FILE_SERVER_RECOVERY_ENABLED

Specifies whether jobs using the file server should attempt to recover following a client mainframe failure. If the value of this system attribute is 1, jobs using the file server attempt to recover. In this case, the server mainframe saves any files that were attached by jobs on the client mainframe. Access to these files continues when the connection from the client mainframe to the server mainframe is reestablished. If the value of this system attribute is 0, jobs using the file server do not attempt to recover. In addition, no file server recovery takes place if the JOB_RECOVERY_OPTION system attribute is set to 3 (do not recover active jobs) or the image file is unavailable. See table 2-2 for the default value for this system attribute.

HALT_ON_HUNG_TASK

Specifies whether a hung task causes the NOS/VE monitor to halt. This system attribute is used primarily for debugging software problems.

If the value of this system attribute is 1, a hung task causes the NOS/VE monitor to halt. If the value of this system attribute is 0, the NOS/VE monitor sends the hung task a signal that causes it to go into an infinite wait loop. Unless you are debugging system problems, set this system attribute to 0.

The status field on the VEDISPLAY ACTIVE_JOBS display for the job shows *H, indicating a hung task. The job, however, does not terminate. Therefore, all resources being used by the job (such as files and tapes) remain assigned. See table 2-2 for the default value for this system attribute.

HALT_ON_PROCESSOR_MALFUNCTION

Specifies whether the NOS/VE system terminates when a processor has a hardware error that cannot be corrected. This system attribute is used primarily for debugging hardware problems.

If the value of this system attribute is 1, the system terminates when an uncorrectable error occurs. If the value of this system attribute is 0, the system retries tasks and terminates those tasks that are affected by the hardware error. See table 2-2 for the default value for this system attribute.

HALT_ON_SWAPIN_FAILURE

Specifies whether bad swap file descriptor data causes the NOS/VE monitor to halt. If the value of this system attribute is 1, bad swap file descriptor data causes the NOS/VE monitor to halt. If the value of this system attribute is 0, the NOS/VE monitor kills the job associated with the corrupt swap file descriptor data, and the monitor continues to run. In addition, a message indicating that the job is dead is sent to the critical display window of the NOS/VE console, the job is swapped out of memory, and the memory associated with the job is freed. See table 2-2 for the default value for this system attribute.

HALTRING

Specifies a P register ring number. A broken task or MCR fault occurring in a ring with a ring number less than or equal to the value of this system attribute causes the NOS/VE monitor to halt. (For more information on broken tasks and MCR faults, refer to appendix D, NOS/VE Processing of Job Mode Software Errors.) Broken tasks occurring in rings with ring numbers greater than the value for the HALTRING system attribute cause a monitor fault to be sent back to the task in job mode. This system attribute is used primarily for debugging software problems.

Do not change the default value for the HALTRING system attribute during normal NOS/VE operations, including deadstart. If you change this value, the system may abort due to an error it would normally tolerate. Change the value of HALTRING only when producing a memory dump for a particular failure and only at the direction of a site analyst. See table 2-2 for the allowable values and default value for this system attribute.

HISTORY_LOG_CRITICAL

Specifies the action NOS/VE takes when it is unable to record an entry in the history log. If the value of this system attribute is 1, NOS/VE terminates any time the system is unable to record an entry. If the value of this system attribute is 0, NOS/VE discards the entry and informs the operator by way of a message in the critical display window. See table 2-2 for the default value for this system attribute.

HISTORY_LOG_MAXIMUM_SIZE

Specifies the maximum size, in megabytes, of the history log. See table 2-2 for the allowable values and default value for this system attribute.

IGNORE_IMAGE

Specifies whether NOS/VE disables the process that recovers modified pages from a memory image during a continuation deadstart. This system attribute is used only during deadstart.

If the value of this system attribute is 1, the system disables the recovery of modified pages from a memory image during a continuation deadstart. If the value of this system attribute is 0, the system uses the memory image during a continuation deadstart. See table 2-2 for the default value for this system attribute.

JOB_RECOVERY_OPTION

Specifies whether the system is to recover active jobs at continuation deadstart. Active jobs are those jobs that are initiated before the system terminates. You can specify one of two values for this system attribute: 0 or 3.

0	Recover active jobs. Jobs that are initiated before the system terminates are recovered on this continuation deadstart. Recovered jobs continue execution from the point at which they were left when the system terminated.
3	Do not recover active jobs. Jobs that are initiated before the system terminates are discarded on this continuation deadstart. The job name of each discarded job appears in the system log.

See table 2-2 for the default value for this system attribute.

All jobs are unrecoverable under any of the following conditions:

- The system fails and a recovery without image is performed.
- An active mass storage volume is disabled at the time a TERMINATE_SYSTEM command is executed.
- A deadstart is performed using a different version of NOS/VE than was in use when the system last went down.
- The operator has disabled job recovery via an operator command during deadstart.
- The system page size has been changed since the last deadstart.

A single job is unrecoverable under any of the following conditions:

- Inconsistencies exist in NOS/VE data structure job descriptions.
- The system is performing critical operations (for example, task initiation, task termination, or permanent file attaches).
- The job has opened a magnetic tape file and a tape unit is currently assigned to the job. A job that has tape units reserved but not assigned is recoverable. Also, a job that has opened a magnetic tape file but is waiting for a tape to be mounted is recoverable.
- A permanent file that the job was using prior to the system failure cannot be reattached. (A permanent file that was deleted while it was attached does not prevent recovery of a job.)
- The system was running in dual-state mode and the job was using interstate communications before failure, and the system is brought up in standalone mode after the failure occurred.
- The JOB_RECOVERY_DISPOSITION parameter for the job has a value other than CONTINUE. This parameter is controlled by the CHANGE_JOB_ATTRIBUTES, CHANGE_JOB_ATTRIBUTE_DEFAULT, LOGIN, JOB, and SUBMIT_JOB commands. With the exception of CHANGE_JOB_ATTRIBUTE_DEFAULT (described in the NOS/VE Operations manual), these commands are described in the NOS/VE Commands and Functions manual.

LONG_WAIT_FORCE_SWAP_TIME

Specifies an ultimate time limit for jobs to remain in memory when all tasks in the job are in wait state (that is, when they are in a timed wait or queued for certain resources) and when the expected wait time is less than the value specified by the LONG_WAIT_SWAP_TIME system attribute. If the time since the job was last executed exceeds the value set by LONG_WAIT_FORCE_SWAP_TIME, the job is swapped out. However, job swapping does not occur if the SWAP_JOBS_IN_LONG_WAIT system attribute is set to 0 (FALSE). See table 2-1 for the allowable values and default value for this system attribute.

LONG_WAIT_SWAP_TIME

Determines whether a job's expected wait time causes it to be swapped out. That is, a job is swapped out if the time it is expected to wait exceeds this value.

The following are instances of how a job goes into wait state:

- After issuing a call to procedures PMP$WAIT or OSP$AWAIT_ACTIVITY_COMPLETION.
- After issuing an SCL WAIT command.
- When, in an interactive session, the terminal is waiting for input. This is also referred to as think state. In this case, the job is always in long wait.

See table 2-1 for the allowable values and default value for this system attribute.

MANDATORY_DUALSTATE

Specifies whether a mainframe is permitted to run in standalone mode if CPU 0 fails. This system attribute is valid only for dual-state systems; it has no effect on standalone systems.

If the value of this system attribute is 1, NOS/VE standalone operations are not permitted. If the value of this system attribute is 0, NOS/VE standalone operations are permitted. If CPU 0 fails, the NOS or NOS/BE system is eliminated, and NOS/VE reverts to standalone mode. See table 2-2 for the default value for this system attribute.

MAX_TIME_SWAP_IO_NOT_INIT

Specifies the maximum amount of time a job swapped out for long wait remains in real memory without swapout I/O taking place after it has exceeded its estimated think time. Estimated think time is described in the MAXIMUM_THINK_TIME system attribute description later in this chapter.

Until the time specified by this system attribute elapses, the job is logically swapped but remains memory-resident. After the time specified by this system attribute elapses, the job is written to disk. Its memory is not freed, however, until it has exceeded its estimated think time by the value set by the MAXIMUM_SWAP_RESIDENT_TIME system attribute.

The job is swapped to disk and its memory freed anytime the number of free and available pages falls below the value specified by the MINIMUM_AVAILABLE_PAGES memory attribute (described in chapter 3, Managing Memory). See table 2-1 for the allowable values and default value for this system attribute.

MAXIMUM_ACTIVE_JOBS

Specifies the maximum number of jobs that can be active in memory at any one time. If this limit is reached, no additional jobs are started or scheduled for execution until an active job terminates or is swapped out of memory. See table 2-1 for the allowable values and default value for this system attribute.

MAXIMUM_ALLOCATION_SIZE

Specifies the maximum allocation size, in bytes, assigned to new files. A file creation request (for example, REQUEST_MASS_STORAGE) that specifies an allocation size greater than the current value of the MAXIMUM_ALLOCATION_SIZE system attribute is automatically reduced to the value of this system attribute.

Whenever a file overflows to a new volume, NOS/VE requires that the new volume supports the allocation size defined for that file. The default file allocation size is 16,384 bytes (16 kilobytes), for which all disks support overflow with no wasted space.

All disks support overflow with some wasted space for file allocation sizes of 32, 64, or 128 kilobytes. For file allocation sizes of 256 kilobytes, 512 kilobytes, or a full cylinder, not all disks can be used for overflow. A file can overflow to any disk that supports an allocation size equal to or greater than the file's allocation size. A file cannot overflow to a disk that has a cylinder size less than the file's allocation size.

If your site has model 844 or 834 disk units along with other types of disk units, you may want to change the value of the MAXIMUM_ALLOCATION_SIZE system attribute to 131,072 bytes. This allows the 844 and 834 disk units to be used for overflow of files from other disk types.

To minimize overflow conflicts, NOS/VE limits a file's maximum allocation size to the current value of the MAXIMUM_ALLOCATION_SIZE system attribute. This attribute's default value of 262,144 bytes maximizes I/O performance for a file with a large allocation size while minimizing overflow conflicts. If the MAXIMUM_ALLOCATION_SIZE system attribute is set to a value greater than the cylinder size for a particular disk device, the system considers the maximum allocation size to be the cylinder size for files on that disk. See table 2-2 for the allowable values and default value for this system attribute.

MAXIMUM_JOB_CLASSES

Specifies the maximum number of job classes that can be defined. For information on defining job classes, refer to the CREATE_CLASS subcommand of the ADMINISTER_JOB_CLASS subutility in chapter 4, Job Scheduling. See table 2-1 for the allowable values and default value for this system attribute.

MAXIMUM_KNOWN_JOBS

Specifies the maximum number of simultaneous jobs. This maximum is the total number of active batch jobs, batch jobs that have been queued for execution, interactive jobs, detached jobs, and jobs destined for QTF (Queue File Transfer Facility) or NTF (Network Transfer Facility). Interactive jobs submitted after this limit is reached are rejected by NOS/VE, causing one of the following error messages (depending on the job's network access method) to be issued to the terminal:

- NAM issues the following message:

```
ILLEGAL APPLICATION
```

- INTERCOM issues the following message:

```
CONNECTION TO VEIAF REJECTED
```

- NAMVE/CDCNET issues the following message:

```
The job resources are temporarily unavailable
```

Batch jobs submitted to NOS/VE from NOS (or NOS/BE) remain queued under the control of the initiating system. See table 2-1 for the allowable values and default value for this system attribute.

MAXIMUM_OUTPUT_FILES

Specifies the maximum number of output files that can be queued for transfer to the batch transfer facility on NOS/VE standalone systems, to NOS (or NOS/BE), and/or to QTF on dual-state systems. An output file is either a file that contains the output and log files from a job, or a file routed with a PRINT_FILE command. If this maximum is reached, additional PRINT_FILE requests are rejected. However, the system routes output and log files as soon as the number of queued files falls below the maximum specified.

See table 2-2 for the allowable values and default value for this system attribute.

NOTE

Setting the value of this system attribute too low can severely degrade system performance.

MAXIMUM_SERVICE_CLASSES

Specifies the maximum number of service classes that can be defined. For information on defining service classes, refer to the CREATE_CLASS subcommand of the ADMINISTER_SERVICE_CLASS subutility in chapter 4, Job Scheduling. See table 2-1 for the allowable values and default value for this system attribute.

MAXIMUM_SWAP_RESIDENT_TIME

Specifies the maximum amount of time a job in long wait remains in real memory with a swapped-out status (the job is written to disk and a copy remains in real memory) after it has exceeded its estimated think time (estimated think time is defined under the MAXIMUM_THINK_TIME system attribute description in this chapter). After the time specified by this system attribute elapses, the job is removed from memory.

The job's memory is freed anytime the number of free and available pages falls below the value specified by the MINIMUM_AVAILABLE_PAGES memory attribute (described in chapter 3, Managing Memory). See table 2-1 for the allowable values and default value for this system attribute.

MAXIMUM_THINK_TIME

For interactive jobs, think time is the amount of time it takes a user to enter the next input. While the job is waiting for input, the job is in long wait.

The system calculates a new estimated think time whenever a job comes out of long wait. It is calculated as follows:

1. The think time is set equal to the time the system waited for terminal input.
2. The think time is then compared with the values specified by the MINIMUM_THINK_TIME and MAXIMUM_THINK_TIME system attributes.
    - If the think time is less than the MINIMUM_THINK_TIME value, the previous think time is used. This happens, for example, if the user used the type-ahead feature. If this is the first calculation of think time and there is no previous think time, the MAXIMUM_THINK_TIME value is used.
    - If the think time is greater than the MINIMUM_THINK_TIME value and less than the MAXIMUM_THINK_TIME value, the think time is not changed.
    - If the think time is greater than the MAXIMUM_THINK_TIME value, the think time is set to the MAXIMUM_THINK_TIME value.

The next time a job goes into long wait, the system calculates when it expects input from the user (estimated ready time). The estimated ready time is the current time plus the think time.

```
estimated ready time = current time + last estimated think time
```

Jobs that are in long wait are swapped out. If enough memory is available, the jobs remain memory resident and are queued according to when the system expects the next input for the job. Refer to the MINIMUM_AVAILABLE_PAGES memory attribute (described in chapter 3, Managing Memory) for a description of when jobs are allowed to remain memory resident.

If an active job needs more pages of real memory and not enough pages are available, one of the memory resident jobs may be freed and its pages made available. The job to be freed is selected as follows:

1. If a job's estimated ready time has elapsed and the additional time specified by the THINK_EXPIRATION_TIME system attribute has also elapsed, the job with the oldest estimated ready time is freed.

2. Otherwise, the job with the most distant estimated ready time is freed.

See table 2-1 for the allowable values and default value for this system attribute.

MAXWS_AIO_THRESHOLD

Specifies a limit used to prevent a task from flooding the system with I/O requests when the task's job reaches its maximum working set size. If a task causes a page to be read from disk when the task's job has reached the specified limit of active I/O requests and the job's working set is at the maximum size, then the task is forced to give up the CPU (task switch). This allows another task to execute. If the new task needs to initiate I/O, it can do so because the I/O queues are not filled by the first task.

This system attribute is effective in controlling jobs that read large files and cause numerous disk reads.

MINIMUM_SHARED_WORKING_SET

Specifies the minimum number of pages to which the task service shared queue can be reduced as a result of page aging. The system will not release shared pages (as a result of page aging) when the number of shared pages is at or below the value specified for this system attribute. This system attribute does not prevent task service shared queue pages from being released as a result of a close file request or job termination. The system will age pages below the number specified by this system attribute if the system is performing aggressive aging (see the AGGRESSIVE_AGING_LEVEL and AGGRESSIVE_AGING_LEVEL_2 memory attribute descriptions in chapter 3, Managing Memory).

For most systems, use the default value of 0. Do not set this system attribute to a value higher than approximately half the memory available to user jobs after deadstart. See chapter 6, Page Aging, for more information about how this system attribute applies to page aging. See table 2-1 for the allowable values and default value for this system attribute.

NOTE

We recommend you use the MINIMUM_SIZE attribute of each shared queue rather than the MINIMUM_SHARED_WORKING_SET system attribute. See the CHANGE_SHARED_QUEUE_ATTRIBUTE subcommand in chapter 3, Managing Memory, for a description of the MINIMUM_SIZE attribute.

MINIMUM_THINK_TIME

See the MAXIMUM_THINK_TIME description in this chapter for a description of the MINIMUM_THINK_TIME system attribute. See table 2-1 for the allowable values and default value for both system attributes.

NETWORK_ACTIVATION

Specifies whether the system automatically attempts to activate the NOS/VE network during execution of the ACTIVATE_PRODUCTION_ENVIRONMENT command.

If the value of this system attribute is 1, the system attempts to activate NAM/VE; if its value is 0, no attempt is made to activate NAM/VE. See table 2-2 for the default value for this system attribute.

PROCESSOR_DUE_THRESHOLD

Specifies the number of nonfatal detected uncorrected errors (DUEs) that a NOS/VE job mode task can encounter in a given CPU before the system removes that CPU from the set of processors on which the task can execute. In a multiple-CPU environment, the task executes on another CPU. In a single-CPU environment, the task aborts and the system attempts to execute a STEP_SYSTEM command. For additional information, see the description of the ABORTED_TASK_THRESHOLD system attribute in this chapter.

READ_TU_EXECUTE

The following description applies to both the READ_TU_EXECUTE and the READ_TU_READ_WRITE system attributes.

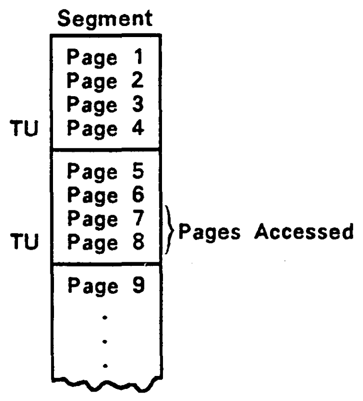
For NOS/VE Version 1.4.1 or later, the system automatically reads multiple pages for each page fault in a segment that is being referenced sequentially. Therefore, the importance of system attributes READ_TU_EXECUTE and READ_TU_READ_WRITE is greatly reduced. The appropriate value for both of these system attributes is simply the number of pages to be read in for each normal (nonsequential) page fault.

We recommend that these system attributes be left at their release values of 1. However, those sites that have increased one or both of these values prior to installing NOS/VE Version 1.4.1 or a later version may find it useful to set both system attributes to 2.

The value of these system attributes must not exceed the number of pages that make up a transfer unit (16K bytes of data). In versions previous to NOS/VE 1.4.1, values greater than that had little effect, but now larger values inhibit the automatic detection of sequential page faults and thus degrade performance.

The NOS/VE hardware divides all segments into transfer units (TU). A transfer unit consists of 16,384 bytes. If your page size is 4,096 bytes (the default), you have four pages per transfer unit. If your page size is 8,192 bytes, you have two pages per transfer unit. (The page size is selected through the CYBER Initialization Package (CIP) at deadstart.)

With these system attributes you can specify the number of pages the system attempts to read into memory when it needs a page. The word *attempt* is used because the system cannot read beyond a transfer unit boundary. For example, if you specified a value of 3 for the READ_TU_EXECUTE system attribute and a read request is made for page 7 in a segment, pages 7 and 8 are read into memory as shown in the following figure. The system cannot read page 9 since it cannot cross the transfer unit boundary. This situation assumes a page length of 4,096 bytes.



Increasing the value of this system attribute may reduce the number of disk accesses for read/execute data, but more memory is used. See table 2-1 for the allowable values and default value for this system attribute.

READ_TU_READ_WRITE

See the READ_TU_EXECUTE description in this chapter for a description of the READ_TU_READ_WRITE system attribute. See table 2-1 for the allowable values and default value for this system attribute.

RECOVER_AT_ALL_COSTS

Specifies whether to ignore certain errors during a recovery deadstart. During a continuation deadstart, NOS/VE may fail with any of the following errors:

```
ERR=VEOS1100- <recorded_vsn> log starts incorrectly
ERR=VEOS1100- <recorded_vsn> log check byte missing
ERR=VEOS1100- <recorded_vsn> invalid log entry
ERR=VEOS1100- Invalid DAT change.
ERR=VEOS1100- Invalid DFL change.
```

These errors indicate that the transaction log for the specified mass storage device has become corrupted or that errors have been encountered while trying to apply changes to system tables. This can occur due to a recovery without image. The system's inability to make these changes means that the tables may be inconsistent, causing user tasks to terminate.

Specifying a value of 1 for this system attribute instructs the system, during a recovery deadstart, to ignore any transaction log that has become corrupted or that contains any change considered invalid. See table 2-2 for the allowable values and default value for this system attribute.

NOTE

As an alternative to using this system attribute, remove the device from the configuration and restore any lost cycles. Refer to the NOS/VE System Performance and Maintenance manual, Volume 2.

SECURITY_LOG_CRITICAL

Specifies the action NOS/VE takes when it is unable to record an entry in the security log. If the value of this system attribute is 1, NOS/VE terminates any time the system is unable to record an entry. If the value of this system attribute is 0, NOS/VE discards the entry and informs the operator by way of a message in the critical display window. See table 2-2 for the default value for this system attribute.

SECURITY_LOG_MAXIMUM_SIZE

Specifies the maximum size, in megabytes, of the security log. See table 2-2 for the allowable values and default value for this system attribute.

SPACE_MESSAGES_TO_CONSOLE

Specifies whether to display the number of volumes low message and the number of volumes out message at the critical display window of the system console. If the value of this system attribute is 1, NOS/VE displays these messages regarding mass storage space. If the value of this system attribute is 0, NOS/VE does not display these messages. See table 2-2 for the default value for this system attribute.

STATISTIC_LOG_CRITICAL

Specifies the action NOS/VE takes when it is unable to record an entry in the statistic log. If the value of this system attribute is 1, NOS/VE terminates any time the system is unable to record an entry. If the value of this system attribute is 0, NOS/VE discards the entry and informs the operator by way of a message in the critical display window. See table 2-2 for the default value for this system attribute.

STATISTIC_LOG_MAXIMUM_SIZE

Specifies the maximum size, in megabytes, for the statistic log. See table 2-2 for the allowable values and default value for this system attribute.

SWAP_FILE_ALLOCATION_SIZE

Specifies the number of bytes per allocation unit for swap files. This system attribute may have an effect on system swapping performance.

If your site has faster swapping devices, you may want to increase this value. However, the value should not exceed the cylinder size of the swapping device. In addition, the value should be a power of two. See table 2-1 for the allowable values and default value for this system attribute.

SWAP_JOBS_IN_LONG_WAIT

Specifies a boolean value that turns long-wait job swapping on or off. If you specify a value of 0 (FALSE), jobs in long wait are not swapped out, regardless of the values entered for the LONG_WAIT_FORCE_SWAP_TIME and LONG_WAIT_SWAP_TIME system attributes. If you specify 1 (TRUE), job-swapping occurs according to the values entered for the LONG_WAIT_FORCE_SWAP_TIME and LONG_WAIT_SWAP_TIME system attributes.

We recommend that you leave this system attribute set to TRUE. You can control swapping to disk to some extent by using the MAX_TIME_SWAP_IO_NOT_INIT and MAXIMUM_SWAP_RESIDENT_TIME system attributes. See table 2-1 for the default value for this system attribute.

SYSTEM_ACTIVATION

Specifies whether system activation is automatically performed (through execution of the ACTIVATE_PRODUCTION_ENVIRONMENT command) during system initiation. System activation refers to making the system available to users at the end of the deadstart process. The ACTIVATE_PRODUCTION_ENVIRONMENT command is described in the NOS/VE Operations manual.

If the SYSTEM_ACTIVATION system attribute is set to 0 (FALSE), the system prompts you to determine whether the system should be activated or whether installation work (file reloading, file upgrading, and installing deferred files) should first be performed. If this system attribute is set to 1 (TRUE), the system is activated without manual intervention. See table 2-2 for the default value for this system attribute.

SYSTEM_DEBUG_RING

Specifies a P register ring number. This system attribute is used primarily for debugging software problems.

Any error that occurs in any task with a ring number less than or equal to the ring number you specify causes the NOS/VE monitor to interrupt the task and call the system core debugger. You can then use the system core debugger to examine the task environment as outlined in the NOS/VE System Performance and Maintenance manual, Volume 2. If you enter the system core debugger subcommand RUN, the NOS/VE monitor takes its normal action for the specific fault. See table 2-2 for the allowable values and default value for this system attribute.

SYSTEM_DEBUG_SEGMENT

Allows use of the system core debugger to be limited to NOS/VE operating system segments only. If an error occurs that meets the criteria for the SYSTEM_DEBUG_RING system attribute, the P register segment number is compared to the current value of the SYSTEM_DEBUG_SEGMENT system attribute to determine whether the system core debugger should be called.

The debugger is called if either of the following is true:

- The value of the SYSTEM_DEBUG_SEGMENT system attribute is 0.
- The P register segment number is less than or equal to the value of the SYSTEM_DEBUG_SEGMENT system attribute.

See table 2-2 for the allowable values and default value for this system attribute.

SYSTEM_ERROR_HANG_COUNT

Specifies the number of broken task errors allowed in any given task before that task is considered a hung task. For more information on broken tasks, refer to appendix D, NOS/VE Processing of Job Mode Software Errors. See table 2-2 for the allowable values and default value for this system attribute.

SYSTEM_LOG_CRITICAL

Specifies the action NOS/VE takes when it is unable to record an entry in the system log. If the value of this system attribute is 1, NOS/VE terminates any time the system is unable to record an entry. If the value of this system attribute is 0, NOS/VE discards the entry and informs the operator by way of a message in the critical display window. See table 2-2 for the default value for this system attribute.

SYSTEM_LOG_MAXIMUM_SIZE

Specifies the maximum size, in megabytes, for the system log. See table 2-2 for the allowable values and default value for this system attribute.

THINK_EXPIRATION_TIME

For a description of this system attribute, refer to the MAXIMUM_THINK_TIME system attribute. See table 2-1 for the allowable values and default value for this system attribute.

UNLOAD_DEADSTART_TAPE

Specifies whether to rewind and unload the deadstart tape or just rewind it at the completion of a tape deadstart. If the value of this system attribute is 1, the deadstart tape is rewound and unloaded. If the value of this system attribute is 0, the deadstart tape is rewound only. See table 2-2 for the default value for this system attribute.

VALIDATE_ACTIVE_SETS

Specifies whether file and catalog management validation processes are performed during a continuation deadstart. If the value of this system attribute is 1, the validation processes are performed. If the value is 0, the processes are not performed.

This system attribute causes the following file and catalog management validation processes to occur for all mass storage sets that are active during a continuation deadstart:

- Breaking file attachments. This process resets the interlocks that were set by jobs in the system prior to an interruption. The interlocks are maintained in the parent catalog of each file.

  If the value of the VALIDATE_ACTIVE_SETS system attribute is 0, file attachments are broken when the file is first attached after a continuation deadstart, rather than during deadstart.

- Catalog validation. This process validates catalog integrity by recalculating and comparing checksums on individual objects in each catalog. The system discards any catalog object that does not validate, provided the system attribute DELETE_UNRECONCILED_FILES is set to 1.

  If the value of the VALIDATE_ACTIVE_SETS system attribute is 0, catalog validation is not performed.

- Catalog reorganization. This process rebuilds the data structure associated with each catalog to conserve mass storage space. This process also automatically moves catalogs if mass storage class membership has changed or if a rebuilt catalog cannot be rewritten without error.

  If the value of the VALIDATE_ACTIVE_SETS system attribute is 0, catalog reorganization and catalog movement are not performed.

- Recovery of overflowed files. When a permanent file overflows from one mass storage volume to another, the catalog is updated to identify the additional volume(s) when the file is detached. A system interruption prevents the overflow update operation.

  If the value of the VALIDATE_ACTIVE_SETS system attribute is 0, recovery of overflowed files occurs when the file is first attached during or after a continuation deadstart.

NOTE

NOS/VE automatically sets the value of the VALIDATE_ACTIVE_SETS system attribute to 1 when one or more of the following events occur:

- A continuation deadstart is attempted without a memory image.
- The system device is initialized.
- The $SYSTEM master catalog is recreated because the volume on which it resides is missing or unavailable during a deadstart.

See table 2-2 for the default value for this system attribute.

Managing Memory 3

Managing Memory 3

This chapter describes the MANAGE_MEMORY utility. This utility lets you display and change the values of various attributes that affect the management of memory in NOS/VE. These attributes are divided into two subsets: memory attributes and shared queue attributes.

Memory Attributes

Most of the memory attributes define limits and timed intervals that directly impact the management of memory. The page streaming attributes enable a site to fine tune page fault processing to improve the effective disk transfer rate for files that are accessed sequentially. The memory attributes are as follows:

AGE_INTERVAL_CEILING
AGE_INTERVAL_FLOOR
AGING_ALGORITHM
AGGRESSIVE_AGING_LEVEL
AGGRESSIVE_AGING_LEVEL_2
JOB_WORKING_SET_AGE_INTERVAL
MINIMUM_AVAILABLE_PAGES
PAGE_STREAMING_PRESTREAM
PAGE_STREAMING_RANDOM_LIMIT
PAGE_STREAMING_READS
PAGE_STREAMING_THRESHOLD
PAGE_STREAMING_TRANSFER_SIZE
PERIODIC_CALL_INTERVAL
SHARED_WORKING_SET_AGE_INTERVAL
SWAPPING_AIC

Table of Memory Attributes

Table 3-1 lists the attributes you can manipulate under the MANAGE_MEMORY utility. The table shows the default value, possible range, and recommended range for each memory attribute.

Table 3-1. Memory Attributes

Attribute Name	Default Value	Possible Range	Normal Range[1]
AGE_INTERVAL_ CEILING	10	1 to 255	5 to 50
AGE_INTERVAL_ FLOOR	3	1 to 255	2 to 20
AGING_ ALGORITHM	4	0 to 100	0 to 8
AGGRESSIVE_ AGING_LEVEL	10 pages	0 to 65,535	8 to 200
AGGRESSIVE_ AGING_LEVEL_2	18 pages	0 to 65,535	10 to 400
JOB_WORKING_ SET_AGE_INTERVAL	8,000,000 microseconds	1,000,000 to 999,999,999	1,000,000 to 20,000,000
MINIMUM_ AVAILABLE_PAGES	400 pages	0 to 65,535	200 to 1,000
PAGE_STREAMING_ PRESTREAM	4 page faults	1 to 255	1 to 10
PAGE_STREAMING_ RANDOM_LIMIT	3 page faults	1 to 255	1 to 10
PAGE_STREAMING_ READS	3 transfer units	1 to 5	2 to 3
PAGE_STREAMING_ THRESHOLD	65,536 bytes	0 to 99,999,999	32,768 to 1,000,000
PAGE_STREAMING_ TRANSFER_SIZE	0 bytes	0 to 4,194,304	See footnote 2.
PERIODIC_CALL_ INTERVAL	1,000,000 microseconds	500,000 to 10,000,000	500,000 to 8,000,000
SHARED_WORKING_ SET_AGE_INTERVAL	8,000,000 microseconds	1,000,000 to 999,999,999	1,000,000 to 20,000,000
SWAPPING_AIC	1	0 to 10,000	1 to 10

1. The normal range given in this table is not the normal range on any one system. Rather, it is a range large enough to normally satisfy all sizes of systems, different CPUs, large and small amounts of real memory, and different page sizes. A value outside of the normal range is not recommended on any system unless unique circumstances warrant it.

2. This memory attribute should be 0 unless you are testing the effects of changing the transfer size. See the PAGE_STREAMING_TRANSFER_SIZE memory attribute description for more information.

Shared Queue Attributes

Each page of the shared working set is assigned to one of six shared queues based on the type of file to which the page belongs. Each shared queue has its own set of attributes that define the aging interval and queue size. The six shared queues are as follows:

Queue	Description
Task services	Contains task services code (system code).
Executable files	Contains shareable permanent file segments that are executable.
Nonexecutable files	Contains shareable permanent file segments that are not executable.
Device files	Contains device file segments.
File server	Contains file server segments.
Other	Reserved for future use.

The following shared queue attributes define the aging interval and size of each shared queue. Together, these attributes define how NOS/VE ages the pages of the shared working sets.

AGE_INTERVAL_CEILING
MINIMUM_SIZE
MAXIMUM_SIZE

These shared queue attributes are set using the CHANGE_SHARED_QUEUE_ATTRIBUTE subcommand of the MANAGE_MEMORY utility.

Page Streaming Overview

Page streaming is a form of page fault processing where multiple pages are read from disk in response to a page fault. Page streaming works only with files that are accessed sequentially. If your workload includes jobs that access large files sequentially, page streaming can improve overall disk throughput by reducing the number of individual disk requests and disk seeks; thus, improving the effective disk transfer rate for sequential files.

Page Streaming Process

Figure 3-1 shows a flow chart of the page streaming process.

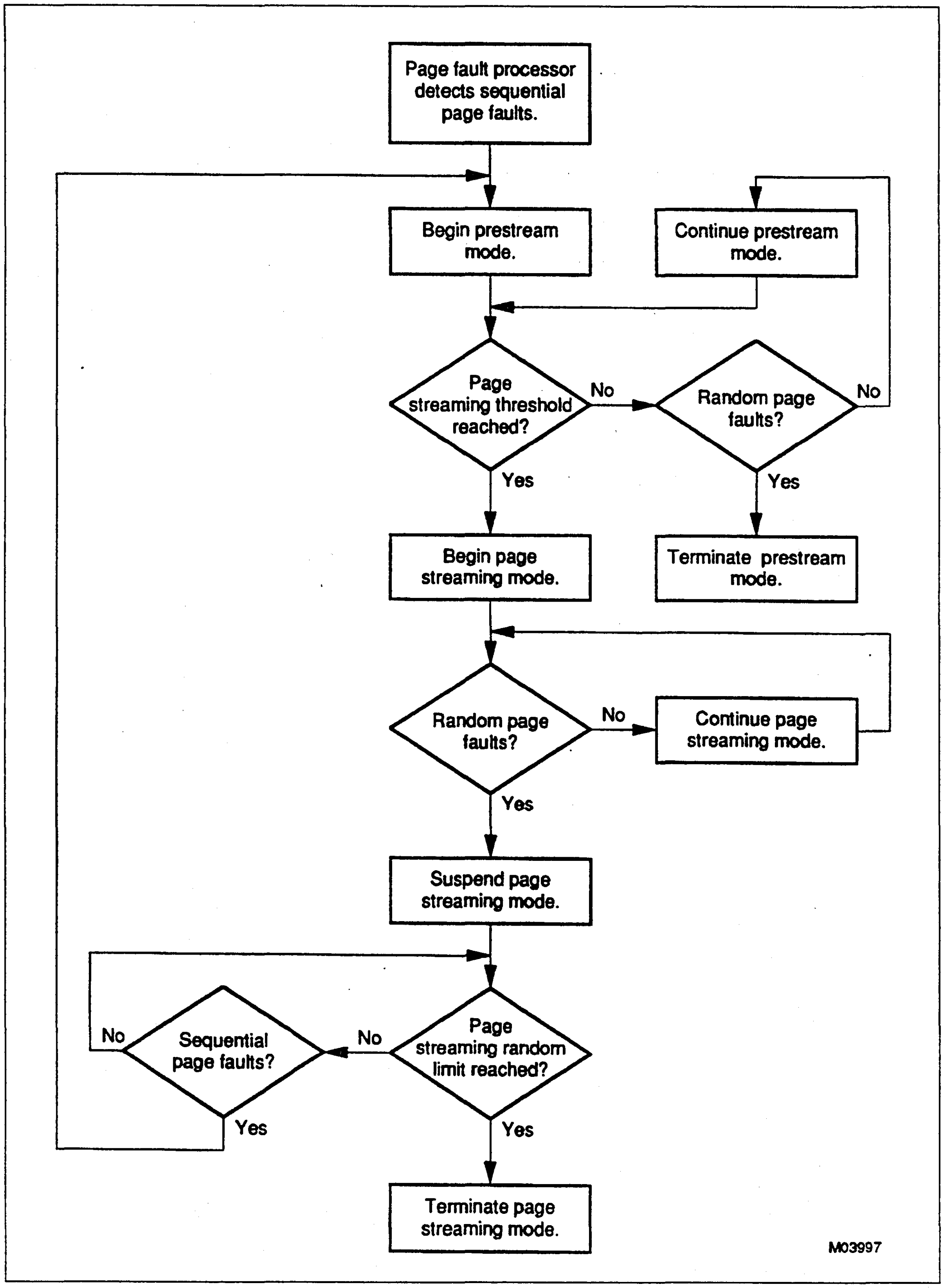

Page fault processor detects sequential page faults.
Begin prestream mode.
Continue prestream mode.
Page streaming threshold reached?
No
Random page faults?
No
Yes
Yes
Begin page streaming mode.
Terminate prestream mode.
Random page faults?
No
Continue page streaming mode.
Yes
Suspend page streaming mode.
No
Sequential page faults?
No
Page streaming random limit reached?
Yes
Yes
Terminate page streaming mode.
M03997

Figure 3-1. Page Streaming Process

The following memory attributes control page streaming.

PAGE_STREAMING_PRESTREAM
PAGE_STREAMING_RANDOM_LIMIT
PAGE_STREAMING_READS
PAGE_STREAMING_THRESHOLD

Initiating Prestream Mode

When the page fault processor detects sequential page faults, it initiates the prestream mode of page fault processing. In prestream mode, the page fault processor satisfies additional page faults by reading in several pages at a time without attempting to stream pages from disk. The PAGE_STREAMING_PRESTREAM memory attribute specifies the number of sequential page faults required to initiate prestream mode.

Initiating Page Streaming Mode

The page fault processor continues to process sequential page faults in prestream mode until a predetermined number of data bytes is read. When this threshold is reached, the page fault processor initiates the page streaming mode of page fault processing. The PAGE_STREAMING_THRESHOLD memory attribute specifies the amount of data required to reach the threshold that initiates page streaming mode.

When operating in page streaming mode, the page fault processor reads a set number of transfer units at one time. The size of the default transfer unit is 16,384 bytes. If your page size is 4,096 bytes (the default), you have four pages per transfer unit. If your page size is 8,192 bytes, you have two pages per transfer unit. (The page size is selected through the CYBER Initialization Package (CIP) at deadstart.) The PAGE_STREAMING_READS memory attribute specifies the number of transfer units read at one time in page streaming mode.

Terminating Page Streaming Mode

When a segment is being processed in page streaming mode, a page fault is considered to be a random page fault if it is not within the same page streaming transfer unit or the next page streaming transfer unit as the last page fault. Random page faults cause a temporary suspension of the page streaming mode for that segment. If the count of random page faults reaches the specified limit before sequential page faults are detected again, the page streaming mode terminates. Later, sequential page faults may again cause the page fault processor to initiate prestream mode and then page streaming mode for the segment. The PAGE_STREAMING_RANDOM_LIMIT memory attribute specifies the number of random page faults required to terminate page streaming mode.

Managing Disks for Streaming

This section describes ways to increase the probability that an application will stream. The ability of an application to stream is determined by the amount of CPU time it consumes, its CPU scheduling priority, and the transfer rate of the disk.

Configuring Disks for Sequential Files

Streaming improves the performance of all applications that access files sequentially. Most sites can benefit from page streaming without changing the way they manage their disk drives. However, if your site runs applications that have a very high I/O to CPU ratio, you can benefit even more by designating certain disks to be used specifically for sequential files. You can do this in the following ways:

- Set up a disk volume with a special mass storage class that can be accessed with the REQUEST_MASS_STORAGE command. For more information about device assignment, refer to appendix D, Assigning Files to a Specific Device, documented in the NOS/VE System Performance and Maintenance manual, Volume 2.
- Set up one or more volumes as a mass storage set. Establish a special family on that set to be used for streaming. Set the default allocation size for these volumes to a large value using the LOGICAL_CONFIGURATION_UTILITY subcommands CHANGE_MS_VOLUME or INITIATE_MS_VOLUME. The Logical Configuration Utility is described in the NOS/VE System Performance and Maintenance manual, Volume 2.

Increasing Allocation Size

Allocation size is the number of bytes that are assigned physically contiguous on a disk volume. A large allocation size has the following advantages:

- Larger amounts of data can be read from disk without repositioning the heads.
- Multiple I/O requests issued by the page fault processor encounter sequential data on disk.
- When accompanied by a large transfer size, the CPU requires less overhead to read files sequentially.
- It takes less memory to describe the file's allocation.

If you use large allocation sizes, we recommend that you do not mix allocation sizes on a particular disk volume. Because mixing allocation sizes increases the data fragmentation of the device. You can specify large allocation sizes for files in the following ways:

- Use the ALLOCATION_SIZE paramenter on the LOGICAL_CONFIGURATION_UTILITY subcommands CHANGE_MS_VOLUME or INITIALIZE_MS_VOLUME. This ensures that all files assigned to the disk volume have large allocation sizes by default.
- Use the ALLOCATION_SIZE parameter on the REQUEST_MASS_STORAGE command. This allows a single file to benefit from a larger allocation size.
- Use the ALLOCATION_SIZE parameter on the RESTORE_PERMANENT_FILES utility subcommand SET_RESTORE_OPTONS. This allows you to place groups of files with large allocation sizes on the same disk volume.

You may want to set the allocation size to a very large value if all your permanent file devices are the same type. This allows users of the REQUEST_MASS_STORAGE command to request allocation sizes up to a cylinder size.

If your site uses a mixture of device types, set the MAXIMUM_ALLOCATION_SIZE system attribute to a value that ensures that files can overflow to all devices that are candidates for the overflow.

Whenever a file overflows to a new volume, NOS/VE requires that the new volume support the allocation size defined for that file. Since large allocation sizes may waste space on a device, there is a trade-off between performance and wasted space. The default file allocation size is 16,384 bytes, for which all disk types support overflow with no unusable space. Cylinder allocation size never wastes space, but files cannot overflow between device types. Other large allocation sizes waste space and may restrict the devices to which a file can overflow. The following table shows the percentage of wasted space on a given disk type for each allocation size.

Allocation Size	**834 Disk**	**836 Disk**	**844 Disk**	**885 Disk**	**887 Disk**	**895 Disk**	**9836 Disk**	**9853 Disk**
16,384	0%	0%	0%	0%	0%	0%	0%	0%
32,768	0%	3%	9%	0%	0%	3%	0%	2%
65,536	20%	9%	27%	0%	5%	3%	0%	2%
131,072	20%	9%	27%	0%	16%	14%	11%	2%
262,144		9%		20%	16%	14%	11%	2%
524,288		9%		20%	16%	14%	11%	35%
Cylinder	0%	0%	0%	0%	0%	0%	0%	0%

Increasing Transfer Size

Even when an application cannot stream, its performance can be improved by using a transfer size that allows it to read large amounts of data each time it accesses a file. For example, a batch job with a low CPU priority could show an improved elapsed time by increasing the transfer size on very large files (those files with a large allocation size). However, a transfer size that is too large can cause swapouts and page aging that may nullify any benefit.

Transfer size specifies how much data is read on each page fault when a file is being accessed sequentially. In page streaming mode, the system attempts to read ahead a fixed number of transfer units at one time. The number of transfer units is determined by the PAGE_STREAMING_READS memory attribute. This allows an overlap of data and CPU processing. Transfer size can be less than, equal to, or greater than allocation size. Transfer size can be specified on the CHANGE_MS_VOLUME and INITIALIZE_MS_VOLUME subcommands of the LOGICAL_CONFIGURATION_UTILITY, on the REQUEST_MASS_STORAGE and ATTACH_FILE commands, and on the FSP$OPEN_FILE program interface.

In most cases, applications need not be concerned with transfer size because the system defaults provide good performance.

The transfer size has the following effects on an application:

- A large transfer size increases the working set size of a job or the shared queue. The effect of this varies depending on the amount of memory.
- A large transfer size accompanied by a large allocation size reduces the CPU requirements to read a file. For example, a transfer size of 65,536 bytes halves the amount of CPU time required to read a file compared to the default transfer size of 16,384 bytes. Increasing the transfer size above 65,536 bytes has a negligible additional improvement.
- A large transfer size with a small allocation size results in multiple I/O requests. Since device management assigns allocation units sequentially, some streaming occurs even with small allocation units. This means that a file written with the default allocation size of 16,384 bytes might benefit by increasing the transfer size before reading the file.
- A small transfer size with a large allocation size can result in reading disks at their maximum rate. NOS/VE's disk drivers process sequential requests without repositioning the heads if the I/O request arrives before the previous request is completed. Tests indicate that the model 887 disk requires a transfer size of at least 32,768 bytes to stream. The default transfer size for the model 887 disk is 32,768 bytes. All other disks stream at the default transfer size of 16,384 bytes.

Using the FREE_BEHIND Parameter

The FREE_BEHIND parameter indicates to the system how an application accesses a file after it has been read. This parameter can be specified on the ATTACH_FILE command and the FSP$OPEN_FILE program interface. Use the FREE_BEHIND parameter if an application's access to a file is in a window smaller than the transfer size. This avoids the necessity of aging the file's pages after they have been processed. The FREE_BEHIND parameter limits the size of working set used in accessing a sequential file to four transfer units.

MANAGE_MEMORY Commands and Subcommands

To use the MANAGE_MEMORY utility to change attributes, you must have system job class administrator privileges. The subcommands that can change attribute values must be executed from the system console. However, subcommands that display the current values can be executed from any NOS/VE terminal.

The MANAGE_MEMORY command initiates the MANAGE_MEMORY utility. The MANAGE_MEMORY utility subcommands are:

CHANGE_MEMORY_ATTRIBUTE
CHANGE_SHARED_QUEUE_ATTRIBUTE
DISPLAY_MEMORY_ATTRIBUTE
DISPLAY_SHARED_QUEUE_ATTRIBUTE
QUIT
SET_TO_DEFAULT

MANAGE_MEMORY Command

Purpose Calls the MANAGE_MEMORY utility.

Format **MANAGE_MEMORY** or
MANM
  STATUS=status variable

Parameters *STATUS*

Returns the completion status for this command.

Remarks

- The MANAGE_MEMORY utility lets you display and change attributes that affect the management of memory in NOS/VE.
- When you enter the MANAGE_MEMORY utility, the following prompt appears:

```
MMU/
```

CHANGE_MEMORY_ATTRIBUTE Subcommand

Purpose Changes the value of the specified memory attribute.

Format **CHANGE_MEMORY_ATTRIBUTE** or
CHAMA
AGE_INTERVAL_CEILING=integer
AGE_INTERVAL_FLOOR=integer
AGGRESSIVE_AGING_LEVEL=integer
AGGRESSIVE_AGING_LEVEL_2=integer
AGING_ALGORITHM=integer
JOB_WORKING_SET_AGE_INTERVAL=integer
MINIMUM_AVAILABLE_PAGES=integer
PAGE_STREAMING_PRESTREAM=integer
PAGE_STREAMING_RANDOM_LIMIT=integer
PAGE_STREAMING_READS=integer
PAGE_STREAMING_THRESHOLD=integer
PAGE_STREAMING_TRANSFER_SIZE=integer
PERIODIC_CALL_INTERVAL=integer
SHARED_WORKING_SET_AGE_INTERVAL=integer
SWAPPING_AIC=integer
STATUS=status variable

Parameters *AGE_INTERVAL_CEILING* or *AIC*

Sets the value of the AGE_INTERVAL_CEILING memory attribute.

AGE_INTERVAL_FLOOR or *AIF*

Sets the value of the AGE_INTERVAL_FLOOR memory attribute.

AGGRESSIVE_AGING_LEVEL or *AAL*

Sets the value of the AGGRESSIVE_AGING_LEVEL memory attribute.

AGGRESSIVE_AGING_LEVEL_2 or *AAL2*

Sets the value of the AGGRESSIVE_AGING_LEVEL_2 memory attribute.

AGING_ALGORITHM or *AA*

Sets the value of the AGING_ALGORITHM memory attribute.

JOB_WORKING_SET_AGE_INTERVAL or *JWSAI*

Sets the value of the JOB_WORKING_SET_AGE_INTERVAL memory attribute.

MINIMUM_AVAILABLE_PAGES or *MINAP*

Sets the value of the MINIMUM_AVAILABLE_PAGES memory attribute.

PAGE_STREAMING_PRESTREAM or *PSP*

Sets the value of the PAGE_STREAMING_PRESTREAM memory attribute.

PAGE_STREAMING_RANDOM_LIMIT or *PSRL*

Sets the value of the PAGE_STREAMING_RANDOM_LIMIT memory attribute.

PAGE_STREAMING_READS or *PSR*

Sets the value of the PAGE_STREAMING_READS memory attribute.

PAGE_STREAMING_THRESHOLD or *PST*

Sets the value of the PAGE_STREAMING_THRESHOLD memory attribute.

PAGE_STREAMING_TRANSFER_SIZE or *PSTS*

Sets the value of the PAGE_STREAMING_TRANSFER_SIZE memory attribute.

PERIODIC_CALL_INTERVAL or *PCI*

Sets the value of the PERIODIC_CALL_INTERVAL memory attribute.

SHARED_WORKING_SET_AGE_INTERVAL or *SWSAI*

Sets the value of the SHARED_WORKING_SET_AGE_INTERVAL memory attribute.

SWAPPING_AIC or *SAIC*

Sets the value of the SWAPPING_AIC memory attribute.

STATUS

Returns the completion status for this subcommand.

Remarks

- For descriptions of memory attributes, see Memory Attribute Descriptions later in this chapter.
- The values specified for memory attributes are not preserved across continuation deadstarts.

CHANGE_SHARED_QUEUE_ATTRIBUTE Subcommand

Purpose Changes the values of the attributes for one or more shared queues.

Format **CHANGE_SHARED_QUEUE_ATTRIBUTE** or
CHASQA
  QUEUE=list of keyword
  AGE_INTERVAL_CEILING=integer
  MINIMUM_SIZE=integer
  MAXIMUM_SIZE=integer
  STATUS=status variable

Parameters **QUEUE** or **Q**

Specifies one or more shared queues whose attributes you want to change. This parameter is required. You must specify one or more of the following keywords:

TASK_SERVICE or TS

Shared queue containing task services code segments (system code).

EXECUTABLE_FILE or EF

Shared queue containing shareable permanent file segments that are executable.

NON_EXECUTABLE_FILE or NEF

Shared queue containing shareable permanent file segments that are not executable.

DEVICE_FILE or DF

Shared queue containing device file segments.

FILE_SERVER or FS

Shared queue containing file server segments.

OTHER

Shared queue reserved for future use.

SYSTEM or S

All system-shared queues.

ALL

Both system-shared queues and site-defined shared queues.

AGE_INTERVAL_CEILING or *AIC*

Helps determine which pages are to be removed from a shared queue when the shared queues are aged. All of the shared queues are aged at an interval defined by the SHARED_WORKING_SET_AGE_INTERVAL memory attribute. Under low memory conditions the system may age shared queues more frequently.

Each time a shared queue is aged, the age of any page that has been used since the last aging is set to zero. The age of any page that has not been used is incremented by one. Any page whose age now exceeds the value of the AGE_INTERVAL_CEILING attribute for that queue is removed from the queue.

Increasing the value of the AGE_INTERVAL_CEILING attribute for a shared queue increases the time that a page of that queue must be unused before it is removed. This allows that shared queue to retain pages in memory longer. However, setting the value too high allows unused pages to remain in memory, which decreases the amount of available memory.

See the DISPLAY_SHARED_QUEUE_ATTRIBUTE subcommand description for the default value of the AGE_INTERVAL_CEILING attribute for each of the shared queues.

MINIMUM_SIZE or *MINS*

Sets the value of the MINIMUM_SIZE shared queue attribute. This attribute defines the minimum number of pages to which a shared queue will be aged. The range of minimum pages is from 0 to 65,535. If the number of pages in a shared queue is reduced to this minimum, the system suspends the process of aging pages from that queue until additional pages are added.

When a shared queue is relatively inactive, increasing the MINIMUM_SIZE attribute causes some of the pages to remain in memory even though they are not being used. This can be useful for pages that are being used at a frequency that is larger than the aging interval.

See the DISPLAY_SHARED_QUEUE_ATTRIBUTE subcommand description for the default value for this parameter for each of the shared queues.

MAXIMUM_SIZE or *MAXS*

Sets the value of the MAXIMUM_SIZE shared queue attribute. This attribute defines the maximum number of pages that a shared queue can contain. The maximum number of pages ranges from 0 to 65,535. See the DISPLAY_SHARED_QUEUE_ATTRIBUTE subcommand description for the default value for this parameter for each of the shared queues.

NOTE

The limit specified by the MAXIMUM_SIZE parameter is not enforced for the current version of NOS/VE.

STATUS

Returns the completion status for this subcommand.

Remarks

- To monitor the impact of changing the shared queue attributes, you can determine the size of the shared queues by using the following methods:
  - Emit the OS0 periodic statistic using the MANAGE_PERIODIC_STATISTICS utility described in chapter 9, Statistics Facility. This statistic contains the number of pages in each of the shared queues.
  - Issue the DISPLAY_SYSTEM_DATA command. This command generates a report that lists the number of pages in each of the shared queues under the subsection titled Job/Memory Statistics. This command is documented in the NOS/VE Software Release Bulletin.
  - Examine the VEDISPLAY GENERAL_STATISTICS display at the system console. This display contains the total number of pages in all of the shared queues. For more information about this display, see the NOS/VE Operations manual.
- The values specified for the shared queue attributes are not preserved across continuation deadstarts.

DISPLAY_MEMORY_ATTRIBUTE Subcommand

Purpose Displays memory attributes and their current values.

Format **DISPLAY_MEMORY_ATTRIBUTE** or
DISMA
DISPLAY_OPTION=list of keyword
OUTPUT=file
STATUS=status variable

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

Specifies one or more memory attributes whose values are to be displayed. You can specify the following keywords; the default is ALL:

AGE_INTERVAL_CEILING or AIC

Displays the current value of the AGE_INTERVAL_CEILING memory attribute.

AGE_INTERVAL_FLOOR or AIF

Displays the current value of the AGE_INTERVAL_FLOOR memory attribute.

AGING_ALGORITHM or AA

Displays the current value of the AGING_ALGORITHM memory attribute.

AGGRESSIVE_AGING_LEVEL or AAL

Displays the current value of the AGGRESSIVE_AGING_LEVEL memory attribute.

AGGRESSIVE_AGING_LEVEL_2 or AAL2

Displays the current value of the AGGRESSIVE_AGING_LEVEL_2 memory attribute.

JOB_WORKING_SET_AGE_INTERVAL or JWSAI

Displays the current value of the JOB_WORKING_SET_AGE_INTERVAL memory attribute.

MINIMUM_AVAILABLE_PAGES or MINAP

Displays the current value of the MINIMUM_AVAILABLE_PAGES memory attribute.

PAGE_STREAMING_PRESTREAM or PSP

Displays the current value of the PAGE_STREAMING_PRESTREAM memory attribute.

PAGE_STREAMING_RANDOM_LIMIT or PSRL

Displays the current value of the PAGE_STREAMING_RANDOM_LIMIT memory attribute.

PAGE_STREAMING_READS or PSR

Displays the current value of the PAGE_STREAMING_READS memory attribute.

PAGE_STREAMING_THRESHOLD or PST

Displays the current value of the PAGE_STREAMING_THRESHOLD memory attribute.

PAGE_STREAMING_TRANSFER_SIZE

Displays the current value of the PAGE_STREAMING_TRANSFER_SIZE memory attribute.

PERIODIC_CALL_INTERVAL or PCI

Displays the current value of the PERIODIC_CALL_INTERVAL memory attribute.

SHARED_WORKING_SET_AGE_INTERVAL or SWSAI

Displays the current value of the SHARED_WORKING_SET_AGE_INTERVAL memory attribute.

SWAPPING_AIC or SA

Displays the current value of the SWAPPING_AIC memory attribute.

ALL

Displays the current value of all memory attributes.

OUTPUT or *O*

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

STATUS

Returns the completion status for this subcommand.

Remarks For descriptions of the memory attributes, see Memory Attribute Descriptions later in this chapter.

DISPLAY_SHARED_QUEUE_ATTRIBUTE Subcommand

Purpose Displays a table of shared queue attributes and their current values.

Format **DISPLAY_SHARED_QUEUE_ATTRIBUTE** or
DISSQA
QUEUE=keyword
OUTPUT=file
STATUS=status variable

Parameters *QUEUE* or *Q*

Specifies a keyword that selects which table of shared queues is to be displayed. You can specify one of the following keywords; the default is SYSTEM:

SYSTEM or S

System-defined shared queues.

ALL

All shared queues.

NOTE

Under the current version of NOS/VE, the keyword ALL displays the same information as the keyword SYSTEM.

OUTPUT or *O*

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

STATUS

Returns the completion status for this subcommand.

Remarks The table generated by this subcommand shows a list of the shared queues and the current values for each shared queue's attributes. In the following example, the values shown are the system defaults:

Queue	AIC	MINS	MAXS
SHARED_TASK_SERVICE	3	0	65,535
SHARED_EXECUTABLE_FILE	1	0	65,535
SHARED_NON_EXECUTABLE_FILE	1	0	65,535
SHARED_DEVICE_FILE	1	0	65,535
SHARED_FILE_SERVER	1	0	65,535
SHARED_OTHER	1	0	65,535

In the table, AIC refers to the AGE_INTERVAL_CEILING shared queue attribute, MINS refers to the MINIMUM_SIZE shared queue attribute, and MAXS refers to the MAXIMUM_SIZE shared queue attribute.

QUIT Subcommand

Purpose Terminates the MANAGE_MEMORY utility.

Format **QUIT** or
QUI
STATUS=status variable

Parameters *STATUS*

Returns the completion status for this subcommand.

Remarks This subcommand causes the MANAGE_MEMORY utility to begin processing the subcommands entered. After processing is completed, the utility is exited.

SET_TO_DEFAULT Subcommand

Purpose Sets default values for the shared queue attributes and the memory attributes.

Format **SET_TO_DEFAULT** or
SETTD
  SET=list of keyword

Parameters **SET**

Specifies one or more keywords that select a shared queue or a set of attributes for which the default values are to be set. This parameter is required. You must specify one or more of the following keywords:

TASK_SERVICE or TS

Sets the default values of the attributes for the task service shared queue.

EXECUTABLE_FILE or EF

Sets the default values of the attributes for the executable file shared queue.

NON_EXECUTABLE_FILE or NEF

Sets the default values of the attributes for the nonexecutable file shared queue.

DEVICE_FILE or DF

Sets the default values of the attributes for the device file shared queue.

FILE_SERVER or FS

Sets the default values of the attributes for the file server shared queue.

OTHER

Reserved for future use.

SHARED_QUEUE_ATTRIBUTE or SQA

Sets the default values of the attributes for all shared queues.

SYSTEM or S

Sets the default values of the attributes for all system-shared queues. Specifying this parameter has the same effect as specifying SHARED_QUEUE_ATTRIBUTE.

MEMORY_ATTRIBUTE or MA

Sets the default values of all memory attributes.

ALL

Sets the default values of all shared queue attributes and memory attributes.

Memory Attribute Descriptions

The following sections describe the memory attributes in alphabetical order.

AGE_INTERVAL_CEILING and AGE_INTERVAL_FLOOR

Helps determine which pages are to be removed from a job's working set when the working set is aged. For an explanation of when and why a working set is aged, refer to the description of the PAGE_AGING_INTERVAL scheduling attribute in chapter 4, Job Scheduling.

Each time a job's working set is aged, the age of each page that has not been used (accessed) since the working set was last aged is incremented by the amount of CPU time the job has used (since the working set was last aged) divided by the value set by the PAGE_AGING_INTERVAL scheduling attribute. All pages with an age exceeding the value of the AGE_INTERVAL_CEILING memory attribute are removed from the job's working set.

Next, the age of the least-recently used page left in the working set is compared to the value of the AGE_INTERVAL_FLOOR memory attribute. If the page's age exceeds the value of the AGE_INTERVAL_FLOOR memory attribute, that page is also removed from the job's working set. (Each time the working set is examined, only one unused page can be removed from the working set as a result of exceeding the value of the AGE_INTERVAL_FLOOR memory attribute.)

NOTE

Only unused pages are affected by the AGE_INTERVAL_CEILING and AGE_INTERVAL_FLOOR memory attributes. If a page in the working set has been used (accessed) since the working set was last aged, the age of that page is reset to zero when the working set is aged; thus, the age of a used page is in no danger of exceeding the values of these memory attributes.

NOTE

Each shared queue has an AGE_INTERVAL_CEILING attribute that affects shared queue aging in much the same way that the AGE_INTERVAL_CEILING memory attribute affects the aging of a job's working set.

See chapter 6, Page Aging, for more information about how this memory attribute applies to page aging. See table 3-1 for the allowable values and default values for these memory attributes.

AGGRESSIVE_AGING_LEVEL

Specifies a value at which the system forces the aging of the shared queues and the job working sets. When the number of reassignable (free and available) pages falls below this value, the system ages and removes pages from the shared queues and job working sets even though the times specified by the SHARED_WORKING_SET_AGE_INTERVAL and JOB_WORKING_SET_AGE_INTERVAL memory attributes have not elapsed.

See table 3-1 for the allowable values and default value for this memory attribute.

AGGRESSIVE_AGING_LEVEL_2

Signals the scheduler that not enough reassignable (free and available) pages reside in real memory. When the scheduler receives this signal, it starts swapping out jobs until the number of reassignable pages reaches a target value. This value is specified by the target memory value in the SCHEDULING_MEMORY_LEVELS scheduling attribute described in chapter 4, Job Scheduling.

If the value of the AGGRESSIVE_AGING_LEVEL_2 memory attribute is set too high, memory is wasted because the scheduler swaps out jobs that could remain in memory if this memory attribute were set properly. If the value of this memory attribute is set too low, the number of reassignable pages may too often fall to the level set by the AGGRESSIVE_AGING_LEVEL memory attribute.

The value specified by this memory attribute also determines the point at which page assignments are restricted to system tasks. When the number of reassignable pages falls below the minimum value, only system tasks are assigned additional pages and no pages are assigned to user tasks. At the same time, the scheduler is swapping user jobs out of memory so that more reassignable pages are made available. Once there is a sufficient number of reassignable pages, user jobs are again assigned pages.

To check the page fault statistics, enter the following command:

```
/display_system_data pf
```

or use the OS1 periodic statistic. See the MANAGE_PERIODIC_STATISTICS utility described in chapter 9, Statistics Facility.

NOTE

The DISPLAY_SYSTEM_DATA command was originally intended for the internal use of Control Data only and may be changed or replaced in a future version of NOS/VE.

Check the LOW_ON_MEMORY and the NO_MEMORY values for the page fault statistics. LOW_ON_MEMORY is a count of the number of times page faults are rejected for user jobs. NO_MEMORY is a count of the number of times page faults are rejected for system-critical tasks. If either of these counts is too high, increase the value of this memory attribute.

NOTE

The value of this memory attribute should be greater than the value specified by the AGGRESSIVE_AGING_LEVEL memory attribute. The values for the AGGRESSIVE_AGING_LEVEL_2 memory attribute and for the thrashing level entry in the SCHEDULING_MEMORY_LEVELS scheduling attribute (described in chapter 4, Job Scheduling) should be equivalent or within a few pages of each other.

See table 3-1 for the allowable values and default value for this memory attribute.

AGING_ALGORITHM

Specifies an integer value indicating how the job's accumulated CPU time is computed for comparison with the page aging interval. This computation helps determine how the pages in a job are aged.

You can specify one of the following entries:

- A value greater than or equal to 4. In this case, CPU time is based on job mode CPU time. Although you can specify a value between 4 and 100, the effect is the same.
- A value less than 4. In this case, CPU time is based on the sum of job mode and monitor mode CPU time. Any value less than 4 has the same effect.

Values greater than or equal to 4 are useful when the system has jobs with very large working sets. In these cases, specifying a value less than 4 can cause the working set to be aged too often. This is because the process of aging a large working set accumulates monitor mode CPU time.

See chapter 6, Page Aging, for more information about how this memory attribute applies to page aging. See table 3-1 for the allowable values and default value for this memory attribute.

JOB_WORKING_SET_AGE_INTERVAL

Governs the aging of the working set when a page fault occurs infrequently for a job. When the job generates page faults often enough (that is, when a page fault occurs before the time specified by the JOB_WORKING_SET_AGE_INTERVAL memory attribute elapses), the PAGE_AGING_INTERVAL scheduling attribute controls the process. For more information about the aging of job working sets, refer to the description of the PAGE_AGING_INTERVAL scheduling attribute in chapter 4, Job Scheduling. If a job is CPU-bound and is no longer page faulting, the job's working set is aged when the time specified by the JOB_WORKING_SET_AGE_INTERVAL memory attribute elapses.

Increasing the value of this memory attribute can increase the probability that a page from a job's working set remains in memory, but it also increases the size of a job's working set, which increases the swap time of a job and decreases the number of jobs that can reside in memory.

See chapter 6, Page Aging, for more information about how this memory attribute applies to page aging. See table 3-1 for the allowable values and default value for this memory attribute.

NOTE

The value of the JOB_WORKING_SET_AGE_INTERVAL memory attribute should be greater than the value of the PERIODIC_CALL_INTERVAL memory attribute.

MINIMUM_AVAILABLE_PAGES

Specifies the minimum number of reassignable (free and available) pages that must be in real memory in order for swapped jobs to remain memory resident.

When a job is swapped out and the number of free and available pages is greater than the value specified by the MINIMUM_AVAILABLE_PAGES memory attribute, the job is put in the long wait queue and not written to disk. If the number of free and available pages is below the value specified by the MINIMUM_AVAILABLE_PAGES memory attribute, the swapped job is written to disk and put in the swap resident queue. The pages of a swap resident job are freed when they are needed. The MAX_TIME_SWAP_IO_NOT_INIT system attribute and the MAXIMUM_SWAP_RESIDENT_TIME system attribute determine how long a job is allowed to remain memory resident. These system attributes are described in chapter 2, Adjusting System Attributes.

When a page of real memory is needed to satisfy a page fault or swap-in request, the page that is used is selected by the following rules:

1. Take pages from the free queue until it is empty.

2. Take pages from the available queue until the next request would make the number of pages less than that specified by the MINIMUM_AVAILABLE_PAGES memory attribute.

3. Free the memory of a swap resident job and use those pages.

4. Take pages from the available queue, even though it already has fewer pages than the amount specified by the MINIMUM_AVAILABLE_PAGES memory attribute.

Setting the MINIMUM_AVAILABLE_PAGES memory attribute to a large number allows jobs to remain memory resident. However, this has the effect of reducing the size of the available queue which may lead to more page faults from disk.

See table 3-1 for the allowable values and default value for this memory attribute.

PAGE_STREAMING_PRESTREAM

Specifies the number of page faults required to initiate the prestream mode of page fault processing for a segment. The page faults must occur sequentially within the segment. See table 3-1 for the allowable values and default value for this memory attribute.

PAGE_STREAMING_RANDOM_LIMIT

Specifies the number of random page faults required to terminate the page streaming mode of page fault processing. Since the determination of whether page faults are random is relative to the previous page fault, a limit of at least three page faults is required to allow a random page fault to occur in the middle of a series of sequential page faults. This is because the page fault that returns to continue the sequential accesses is counted as a second random page fault, since it is random relative to the first random page fault. See table 3-1 for the allowable values and default value for this memory attribute.

PAGE_STREAMING_READS

Specifies the number of transfer units read at one time when the page fault processor is operating in page streaming mode. The page fault processor is designed for a value of three transfer units. If this value is set to 2, it reduces memory requirements but also reduces the probability of streaming data from a disk at the device's maximum rate. See table 3-1 for the allowable values and default value for this memory attribute.

PAGE_STREAMING_THRESHOLD

Specifies the amount of data, in bytes, that the page fault processor must read before it can initiate the page streaming mode. While in prestream mode, the page fault processor continues to access pages through additional sequential page faults and totals the pages until it reaches the data threshold. See table 3-1 for the allowable values and default value for this memory attribute.

PAGE_STREAMING_TRANSFER_SIZE

Normally, this attribute should be set to 0. It is used primarily as a debugging aid for testing changes in transfer size. The page fault processor uses a logical transfer size to determine how much data to read while operating in prestream mode or page streaming mode. The transfer size is also used to define sequential accesses. Normally, the transfer size is obtained as a default from the device type, the volume attributes, a REQUEST_MASS_STORAGE command that allocates space for the segment, an ATTACH_FILE command, or opening a file. If the PAGE_STREAMING_TRANSFER_SIZE memory attribute is set to a nonzero value, the specified value is used for all segments in the system unless an ATTACH_FILE command or an FSP$OPEN_FILE program interface specifies a transfer size. See table 3-1 for the allowable values for this memory attribute.

PERIODIC_CALL_INTERVAL

Specifies the amount of time, in microseconds, between calls to the NOS/VE memory manager to perform memory cleanup and aging functions. The timing for the memory cleanup and aging functions is determined by the SHARED_WORKING_SET_AGE_INTERVAL memory attribute and the JOB_WORKING_SET_AGE_INTERVAL memory attribute.

Increasing the value of the PERIODIC_CALL_INTERVAL memory attribute can reduce memory manager overhead but can increase the working set sizes. See chapter 6, Page Aging, for more information about how this memory attribute applies to page aging. See table 3-1 for the allowable values and default value for this memory attribute.

NOTE

The values of the SHARED_WORKING_SET_AGE_INTERVAL and JOB_WORKING_SET_AGE_INTERVAL memory attributes should be greater than the value of the PERIODIC_CALL_INTERVAL memory attribute.

SHARED_WORKING_SET_AGE_INTERVAL

Specifies the rate, in microseconds, at which the system ages shared queues. When a queue is aged, a page is removed if its age exceeds the value specified by the AGE_INTERVAL_CEILING memory attribute.

Expiration of the time specified by the SHARED_WORKING_SET_AGE_INTERVAL memory attribute is checked within the NOS/VE memory manager. The memory manager is called at intervals established by the PERIODIC_CALL_INTERVAL memory attribute. The value of the SHARED_WORKING_SET_AGE_INTERVAL memory attribute should be greater than the value of the PERIODIC_CALL_INTERVAL memory attribute.

See chapter 6, Page Aging, for more information about how this memory attribute applies to page aging. See table 3-1 for the allowable values and default value for this memory attribute.

SWAPPING_AIC

Specifies the number of times an unused page in a job's working set can be swapped out of memory before it is removed from the working set. (AIC stands for aging interval ceiling.)

This memory attribute may have a significant effect on system performance. Increasing this value may increase the chances that a page will be in memory but may also increase the swap file size. An increase in swap file size may, in turn, increase swap time.

See chapter 6, Page Aging, for more information about how this memory attribute applies to page aging. See table 3-1 for the allowable values and default value for this memory attribute.

Job Scheduling 4

Job Scheduling 4

This chapter tells you how to set up and manage the NOS/VE job scheduler. After providing an overview of job scheduling, this chapter describes the utilities that create and manage job scheduling profiles, as well as the job scheduling attributes you can set to tune the scheduler. The last section of the chapter contains detailed descriptions of the job scheduling attributes listed in alphabetical order.

Introduction to the Job Scheduler

The NOS/VE job scheduler controls how jobs proceed through the system from the time they are submitted until the resulting output files are printed. The job scheduler orders jobs and allocates the available resources of the system according to need and according to equity. Since what is equitable varies from system to system, the scheduler must be tailorable to the requirements of each system.

NOS/VE provides two utilities you can use to control the activities of the job scheduler and to tailor those activities according to your particular needs:

- ADMINISTER_SCHEDULING utility
- MANAGE_ACTIVE_SCHEDULING utility

To use either of these utilities, your user name must be validated for the SCHEDULING_ADMINISTRATION capability as described in the NOS/VE User Validation manual.

You use the ADMINISTER_SCHEDULING utility to create or modify an overall scheduling profile for your site. The scheduling profile defines your normal job scheduling requirements in terms of job class structure, service class structure, and applications processing. The profile can reside on any file of your choice; the default file is $USER.SCHEDULING_PROFILE. The output of an ADMINISTER_SCHEDULING utility session is a file containing the new or modified profile. Changes are written to the file containing the scheduling profile only when you issue a WRITE_PROFILE subcommand or a QUIT subcommand with the SAVE_CHANGE parameter set to TRUE.

To activate the new or modified scheduling profile, you exit the ADMINISTER_SCHEDULING utility, enter the MANAGE_ACTIVE_SCHEDULING utility, and issue an ACTIVATE_PROFILE subcommand. This subcommand uses your specified changes to modify the system scheduler tables that govern scheduler activity. The ACTIVATE_PROFILE subcommand also creates the following permanent file in the $SYSTEM catalog: $SYSTEM.SCHEDULING.OSF$SYSTEM_PROFILE.

For sites with multiple mainframes, you can create a single scheduling profile to describe the job scheduling characteristics for all mainframes.

In addition to activating the scheduling profile created or modified under the ADMINISTER_SCHEDULING utility, the MANAGE_ACTIVE_SCHEDULING utility is used to make routine or local modifications to the job scheduler characteristics defined in the scheduling tables. For example, you would use the MANAGE_ACTIVE_SCHEDULING utility to make routine adjustments for weekend or nonprime-time system usage.

Structural Changes to the Scheduling Profile

Structural changes to the scheduling profile occur by using the ADMINISTER_SCHEDULING utility to create or delete a scheduling control, service class, job class, job category, or application or by changing an attribute in the membership group. Two scheduling profiles are structurally identical only if they originated from the same profile file and you used the ADMINISTER_SCHEDULING utility to change only attributes.

For multimainframe sites using the load leveling feature, it is important that the scheduling profiles on all load-leveled mainframes be structurally identical. For multimainframe sites using the NOS/VE File Server without load leveling, it is recommended, but not required, that all mainframes use a structurally identical scheduling profile.

You can make minor changes to a scheduling profile without changing the profile's structure. For example, you can change the attributes associated with a particular service class. But any change that adds or deletes a job category, job class, or service class does constitute a structural change to the profile.

Job Scheduling Categories and Attributes

The following table contains basic scheduling profile terminology. These terms define the primary characteristics of a scheduling profile.

Term	Description
Job categories	Optional, logical groupings of jobs that you may define according to whatever characteristics are important to you. For example, if you want to define special processing for jobs requiring a tape mount, you can create a job category called TAPE_JOB to identify jobs requiring that special processing. Similarly, you can define a job category called BIG_JOB to help define special processing needed for jobs with heavy demands on CPU time. A job can be assigned more than one job category. Job categories are created using the CREATE_JOB_CATEGORY subcommand of the ADMINISTER_SCHEDULING utility.
Scheduling control attributes	Scheduling attributes that define processing characteristics such as memory assignment and controls. These characteristics are applicable to all jobs regardless of their job class, service class, or job categories. Scheduling control attributes are defined using the ADMINISTER_CONTROLS subutility of the ADMINISTER_SCHEDULING utility.
Job class attributes	Scheduling attributes that govern the input and initiation phases of job processing. For example, job class attributes determine when or on which mainframe a particular job is initiated. Job classes may be restricted to jobs with certain job categories. Job classes are defined using the ADMINISTER_JOB_CLASS subutility of the ADMINISTER_SCHEDULING utility.

Term	Description
Service class attributes	Scheduling attributes that govern the execution phase of job processing, including job swapping and determining dispatching priorities. Service class attributes are defined using the ADMINISTER_SERVICE_CLASS subutility of the ADMINISTER_SCHEDULING utility.
Application scheduling attributes	Scheduling attributes that apply only to certain site-selected application programs. While a job is executing one of these programs, the normal service class attributes for the job are preempted by the service class attributes defined for the application program. When execution of the application terminates, the job reverts to its normal service class attributes. Applications are defined using the ADMINISTER_APPLICATION subutility of the ADMINISTER_SCHEDULING utility.

The individual attributes used to define job scheduling controls, job class characteristics, service class characteristics, and application scheduling characteristics are described in Scheduling Attributes later in this chapter. When creating or changing a scheduling profile entry, you assign values to individual attributes using the CHANGE_ATTRIBUTE subcommand of the appropriate ADMINISTER_SCHEDULING subutility. For example, job class attributes (of which INITIAL_SERVICE_CLASS and INITIAL_WORKING_SET are examples) are specified as parameters on the CHANGE_ATTRIBUTE subcommand of the ADMINISTER_JOB_CLASS subutility.

The following paragraphs provide a general outline of the tasks involved in creating a scheduling profile. While you read this information, it may be helpful to look at the Scheduling Profile Examples section of this chapter.

Creating Job Categories

Normally, you should not need to create your own job categories. The default scheduling profile automatically supplies two job categories, BATCH and INTERACTIVE. These two categories should suffice in meeting the processing needs of most sites. However, if your site runs a variety of jobs for which special processing needs exist (for example, tape jobs, large jobs, vector jobs), you may find it beneficial to define your own job categories. These job categories can either replace or supplement the default BATCH and INTERACTIVE job categories.

When creating a set of job categories for your site, keep the following in mind:

- The decision of whether a job may be assigned to a job class may be made using job categories. Therefore, you may want to define job categories based on job characteristics that a job must or must not possess to be placed into a job class.

- The decision of whether a mainframe has the necessary resources to run a job can be made using job categories. Therefore, you may want to define job categories based on job characteristics that a job must or must not possess to be run.

  For example, your site may want to prevent tape jobs from initiating when operators are not present. If you create a job category based on the MAGNETIC_TAPE_LIMIT job attribute set by a LOGIN, SUBMIT_JOB, or JOB command, you can control these jobs using the MANAGE_ACTIVE_SCHEDULING utility independently of the job class to which they are assigned.

- A job may be assigned no job category, or it may be assigned more than one job category.

To create a job category and define the job characteristics required for a job to be assigned that category, use the CREATE_JOB_CATEGORY subcommand of the ADMINISTER_SCHEDULING utility.

Defining Job Classes and Service Classes

Before you begin defining the job classes and service classes for your site, you need to familiarize yourself with the job class and service class attributes. These attributes are listed in table 4-13, Scheduling Attributes, and are described in the Scheduling Attributes section.

One of the job class attributes, INITIAL_SERVICE_CLASS, specifies the service class at which jobs in the new job class begin executing. It is this INITIAL_SERVICE_CLASS parameter that establishes the link between a job class and its associated service class.

In order to reference this service class, you must first create it. You create a service class using the CREATE_CLASS subcommand of the ADMINISTER_SERVICE_CLASS subutility. You can tailor the attribute values using the CHANGE_ATTRIBUTE subcommand.

The manner in which you create a job class is similar to that of creating a service class. You initially create a job class using the CREATE_CLASS subcommand of the ADMINISTER_JOB_CLASS subutility. On the CREATE_CLASS subcommand, you assign a name to the new job class and specify a set of initial values for the job class attributes. If you want to modify any of the initial values for an attribute, you can do so by entering a subsequent CHANGE_ATTRIBUTE subcommand.

One of the job classes you define can be designated as the site default job class. You specify a site default job class on the CHANGE_JOB_ATTRIBUTE_DEFAULTS operator command, described in the NOS/VE Operations manual. You can also specify default job classes on an individual user basis. For information about how to define individual user default job classes, see the NOS/VE User Validation manual.

Assigning Job Categories to Job Classes

Before a job can be accepted by the job scheduler for processing, it must satisfy two requirements: it must meet any membership requirements you have established for the mainframe on which you want it to execute, and it must qualify for processing under a valid job class.

Specifying What Jobs Can Process on a Mainframe

In a multimainframe environment, you can restrict the types of jobs that can be run on each mainframe by using specific scheduling control attributes. These attributes are described in the following table and are changed using the ADMINISTER_CONTROLS subutility.

Use the VALIDATION_REQUIRED_CATEGORIES and VALIDATION_EXCLUDED_CATEGORIES scheduling control attributes to restrict the acceptance of user jobs into the input queue to those jobs that are compatible with your site's configuration. Use the INITIATION_REQUIRED_CATEGORIES and INITIATION_ EXCLUDED_CATEGORIES scheduling control attributes to restrict the initiation of the jobs in the input queue to a particular mainframe and time.

The use of these scheduling control attributes is optional. The default values allow any job to be entered and initiated on any mainframe.

Scheduling Control Attribute	**Description**
VALIDATION_REQUIRED_CATEGORIES	Defines a list of job categories that a job must have to be accepted into the input queue on a specified mainframe. These job categories are used only to determine if the job should be accepted or rejected. If the job is assigned all of these job categories, the job is accepted. Since this attribute only controls acceptance of jobs into the input queue, you must also specify the INITIATION_REQUIRED_CATEGORIES control attribute to prevent initiation of these jobs.
VALIDATION_EXCLUDED_CATEGORIES	Defines a list of job categories that a job must not have and still be accepted into the input queue on a specified mainframe. These job categories are used only to determine if the job should be accepted or rejected. If the job has none of these job categories, the job is accepted. Since this attribute only controls acceptance of jobs into the input queue, you must also specify the INITIATION_EXCLUDED_CATEGORIES control attribute to prevent initiation of these jobs.
INITIATION_REQUIRED_CATEGORIES	Defines a list of job categories that a job must have to initiate (start executing) on a specified mainframe. To start executing on this mainframe, a job must be assigned all of these job categories. Generally, this control attribute should have the same values as the VALIDATION_REQUIRED_CATEGORIES control attribute; but as conditions warrent, it may be more or less restrictive.
INITIATION_EXCLUDED_CATEGORIES	Defines a list of job categories that a job cannot have and still initiate on a specified mainframe. To start executing on this mainframe, a job can have none of these job categories. Generally, this control attribute should have the same values as the VALIDATION_EXCLUDED_CATEGORIES control attribute; but as conditions warrent, it may be more or less restrictive.

For example, consider a site where two mainframes are connected by a file server and each is a client of the other. Assume that this site has both a CYBER 990 and a CYBER 860. The CYBER 860 contains the job category VECTOR_JOB in its VALIDATION_EXCLUDED_CATEGORIES and INITIATION_EXCLUDED_CATEGORIES lists. VECTOR_JOB represents a job category whose JOB_QUALIFIER input attribute is set to VECTORS. If a user attempts to access the CYBER 860 and specifies JOB_QUALIFIER=VECTORS on the LOGIN command, the job is rejected. If job leveling is active and the user submits a batch job with JOB_QUALIFIER=VECTORS, that job is accepted but it only runs on the CYBER 990 since the INITIATION_EXCLUDED_CATEGORIES control attribute prevents it from initiating on the CYBER 860.

As you can see, these control attributes provide a convenient method of indicating that only jobs with the given job categories can be processed on a mainframe. Since the INITIATION_REQUIRED_CATEGORIES and INITIATION_EXCLUDED_CATEGORIES control attributes can be changed using the MANAGE_ACTIVE_SCHEDULING utility, they let you temporarily prevent jobs with certain job categories from initiating on a mainframe while still allowing the jobs to be submitted.

The following shows how the decisions are made using categories:

```
leveled=(submitting from client of login_family AND this client has LEVELED
   access to the login_family) OR (submitting from server of login_family AND at
   least one client mainframe has LEVELED access to the login_family);

IF (NOT leveled) OR (job_destination_usage=VE_LOCAL) THEN
   Validate using the validation categories of the submitting mainframe;

   {The job can run only on the submitting mainframe.  The job initiates only
   {if the submitting mainframe's initiation categories allow it to initiate.

ELSEIF (job_destination_usage=VE_FAMILY) THEN
   Validate using the validation categories of the server mainframe for the
   login_family;

   {The job can run only on the server mainframe.  The job initiates only if
   {the server mainframe's initiation categories allow it to initiate.

ELSE (leveled) AND (job_destination_usage=VE)
   Validate using the validation categories of the server mainframe and any
   client mainframe that has LEVELED access to the login_family;

   {Submission is successful if the validation succeeded on one or more of the
   {mainframes.  The job can run on the server or any other mainframe that
   {has LEVELED access to the login_family, even mainframes on which the job is
   {not valid.  The job initiates only on a mainframe whose initiation
   {categories allow it to initiate.

IFEND
```

Example 3 in the Scheduling Profile Examples section of this chapter illustrates how you can restrict the types of jobs that can be run on a particular mainframe. The scheduling profile created during this sample ADMINISTER_SCHEDULING utility session defines two mainframes: a CYBER 830 and a CYBER 860. The site does not want tape jobs to run on the CYBER 860, which has no tape units. In addition, the site has chosen not to run jobs with large working sets on the smaller CYBER 830 machine. Other examples of how to control job initiation using job categories and job classes appear in Scheduling Profile Examples later in this chapter.

Qualifying Jobs for Processing Under a Job Class

You can determine the types of jobs that can be processed under a given job class by defining the specific job categories that are either permitted to, or excluded from, processing under that job class.

For each job class defined in the scheduling profile, the scheduling profile maintains two lists of associated job categories: a required categories list and an excluded categories list. The required categories list is a list of zero or more job categories that a job must be assigned in order to process under the job class. That is, if the required categories list for a job class lists three job category names, a job must be assigned all three of those job categories before it can be processed under that job class.

As the name implies, the excluded categories list does just the opposite; it specifies a list of job categories which, if assigned to a job, prevents that job from processing under the job class. These two lists are mutually exclusive; no job category can appear in both the required categories list and the excluded categories list for the same job class.

If no required or excluded categories are specified for a job class, no membership restrictions are imposed on the class; any validated job may be a member of that job class.

The lists of required and excluded categories are maintained using the ADMINISTER_JOB_CLASS subutility. Once you enter the ADMINISTER_JOB_CLASS subutility, you have a choice of two methods by which to modify the required and excluded lists:

- The CHANGE_ATTRIBUTES subcommand has a REQUIRED_CATEGORIES parameter and an EXCLUDED_CATEGORIES parameter. You can change either list by specifying a new list of values on the corresponding CHANGE_ATTRIBUTES parameter.

- The ADD_JOB_CATEGORY_ENTRY and DELETE_JOB_CATEGORY_ENTRY subcommands provide a way for you to add or delete entries from either the required categories list or the excluded categories list without having to respecify the entire list. Each of these subcommands has a REQUIRED_CATEGORIES parameter and an EXCLUDED_CATEGORIES parameter on which you can specify additions to, or deletions from, either list.

Defining Special Processing for Application Programs

You may want to define special processing characteristics for some applications programs that run at your site independent of the jobs from which the applications are called. You can do this using the CREATE_APPLICATION subcommand of the ADMINISTER_APPLICATION subutility. The CREATE_APPLICATION subcommand specifies:

- The name of the application, as defined in the module header of the object library containing the program. Special application processing is supported only for program descriptions, command procedures, and load modules defined as applications on an object library. The application name specified on the CREATE_APPLICATION subcommand must be unique within the scheduling profile.
- A set of initial values for the application scheduling attributes. These attributes define the special processing requirements to be used for the application. You may use the CHANGE_ATTRIBUTE subcommand to modify any of the individual values created by the CREATE_APPLICATION subcommand.

Once an application definition is added to the active scheduling profile, the application always runs under the special characteristics defined in the definition, regardless of the job from which the program was called.

Load Leveling

The load leveling feature is available for multimainframe sites using the File Server. Load leveling refers to the automatic distribution of jobs between mainframes connected by the File Server. The purpose of load leveling is to distribute the job workload across mainframes in a manner that is transparent to users. For the current version of NOS/VE, the load leveling feature is available only for batch mode jobs.

For information about how to configure the File Server for load leveling, see the NOS/VE File Server for STORNET and ESM-II manual.

All mainframes that participate in load leveling must have active profiles that are structurally identical. This is accomplished by using the same profile file generated by the ADMINISTER_SCHEDULING utility or the MANAGE_ACTIVE_SCHEDULING utility. Then avoid making changes under the ADMINISTER_SCHEDULING utility that cause structural changes. Structural changes occur by creating or deleting a scheduling control, service class, job class, job category, or application or by changing an attribute in the membership group.

Load leveling is done by the NOS/VE system task called *job leveler*. This task periodically checks the mainframes with families that are designated as leveled families for jobs that are candidates to run on the current machine. A family is designated as served for leveled access by using the CHANGE_CLIENT_ACCESS subcommand of the MANAGE_FILE_SERVER utility at the server mainframe. The MANAGE_FILE_SERVER utility is described in the NOS/VE File Server for STORNET and ESM-II manual. The CHANGE_CLIENT_ACCESS subcommand specifies how a family is served to a specific mainframe. This subcommand is also used to assign leveled access for a family to a client mainframe.

The job leveler task activates when leveled families are detected on a server mainframe, even if the ENABLE_JOB_LEVELING control attribute is FALSE. The job leveler task periodically scans each server mainframe for jobs in leveled families that could be run on the client mainframe. The length of the scanning period is determined by the JOB_LEVELING_INTERVAL control atribute. The number of jobs brought back to the client is based on the client's need and on the comparative selection priorities of the jobs already queued on the client and the available server jobs. Two biases are applied to these priorities when making this comparison: the controls attribute JOB_LEVELING_PRIORITY_BIAS and the job class attribute JOB_LEVELING_PRIORITY_BIAS. A job on the server mainframe must have a selection priority that is greater, by the sum of these biases, than the comparative queued job on the client mainframe before the server job is considered for assignment to the client.

Load Leveling Objectives

Load leveling can be used to accomplish two things. First, it can be used to keep each mainframe in a configuration at its maximum load. This is done automatically for as long as there are sufficient jobs to saturate the configuration. Second, it can be used to provide an automated means to restrict or direct jobs with special requirements to a specific mainframe or a specific set of mainframes.

In order to activate load leveling, the following steps must be done:

1. Each server mainframe must indicate for each family those mainframes to which the families' jobs may be leveled. This is done with the CHANGE_CLIENT_ACCESS subcommand of the MANAGE_FILE_SERVER utility. See the NOS/VE File Server for STORNET and ESM-II manual for information about this subcommand.

2. The same scheduling profile generated by the ADMINISTER_SCHEDULING utility must be installed on each mainframe participating in load leveling.

3. Each mainframe participating in load leveling must set the scheduling control attribute ENABLE_JOB_LEVELING to TRUE.

Scheduling Profile Examples

A scheduling profile is a series of tables created by the ADMINISTER_SCHEDULING utility. Once the utility completes, the job scheduling profile contains the defined job categories, control attributes, job class attributes, service class attributes, and application scheduling attributes described earlier in this chapter. The following examples illustrate how to display, create, and use a scheduling profile.

Example 1: Displaying a Scheduling Profile

The following example displays all of the elements of the scheduling profile on file $USER.SCHEDULING_PROFILE to output file DISPLAY. If the site has not yet created a scheduling profile, the default scheduling profile is displayed.

```
/administer_scheduling
AS/display_profile_summary display_options=all output=display
AS/display_job_category category_name=all output=display.$eoi
```

```
AS/administer_controls
AC/display_attribute mainframe_name=all display_option=all output=display.$eoi
AC/quit

AS/administer_job_class
AJC/display_attribute class_name=all display_option=all output=display.$eoi
AJC/quit

AS/administer_service_class
ASC/display_attribute class_name=all display_option=all output=display.$eoi
ASC/quit

AS/administer_application
AA/display_attribute application_name=all display_option=all output=display.$eoi
AA/quit
AS/quit save_change=false
```

Example 2: Creating a Simple Scheduling Profile

The following example creates a scheduling profile by making simple changes to the default scheduling profile. It shows how to control the number of jobs that are allowed to execute concurrently in the default service classes by setting the MAXIMUM_ ACTIVE_JOBS attribute. In addition, the example shows how, by setting the INITIATION_LEVEL attribute, you can control the number of jobs that can be initiated from the default job classes. This sample profile also creates a new job class and an associated service class based on the default BATCH class which a site can tailor to its own needs.

```
/administer_scheduling
AS/create_default_profile result=$user.scheduling_profile.$next

AS/administer_service_class
ASC/change_attribute class_name=interactive maximum_active_jobs=100
ASC/change_attribute class_name=batch maximum_active_jobs=50
ASC/change_attribute class_name=system maximum_active_jobs=20
ASC/change_attribute class_name=maintenance maximum_active_jobs=100
ASC/create_class class_name=site_class default_values=batch
ASC/change_attribute class_name=site_class abbreviation=sc ..
ASC../maximum_active_jobs=5
ASC/quit

AS/administer_job_class
AJC/change_attribute class_name=interactive initiation_level=unlimited
AJC/change_attribute class_name=batch initiation_level=10
AJC/change_attribute class_name=system initiation_level=unlimited
AJC/change_attribute class_name=maintenance initiation_level=10
AJC/create_class class_name=site_class default_values=batch
AJC/change_attribute class_name=site_class abbreviation=sc ..
AJC../initiation_level=5 initial_service_class=site_class ..
AJC../selection_priority=(2000 5000 10)
AJC/quit
AS/quit save_change=true
```

Example 3: Creating a More Complex Scheduling Profile

The following more complex example creates a scheduling profile for a site that wants to establish job classes based on two main factors: the CPU time a job is expected to use and the job's maximum working set size. Two mainframes are used: a CYBER 830 ($SYSTEM_0830_0222) with tape units and a CYBER 860 ($SYSTEM_0860_0128) without tape units. The site does not want jobs with large working sets to run on the CYBER 830.

```
/administer_scheduling
AS/create_default_profile my_profile
AS/create_job_category tape_job magnetic_tape_limit=1..unlimited
AS/create_job_category short_job cpu_time_limit=0..20
AS/create_job_category long_job cpu_time_limit=21..unlimited
AS/create_job_category small_job maximum_working_set=0..199
AS/create_job_category big_job maximum_working_set=220..999
AS/create_job_category huge_job maximum_working_set=1000..unlimited

AS/administer_job_class
AJC/create_class short_small
AJC/change_attributes required_categories=(short_job,small_job) ..
AJC../automatic_class_selection=yes initial_service_class=batch

AJC/create_class short_bigger
AJC/change_attributes excluded_categories=(long_job,small_job) ..
AJC../automatic_class_selection=yes initial_service_class=batch

AJC/create_class long_smaller
AJC/change_attributes required_categories=(long_job,small_job) ..
AJC../automatic_class_selection=yes initial_service_class=batch

AJC/create_class long_huge
AJC/change_attributes required_categories=(long_job,huge_job) ..
AJC../automatic_class_selection=yes initial_service_class=batch
AJC/quit

AS/administer_controls
AC/create_controls $system_0830_0222
AC/create_controls $system_0860_0128

AC/change_attributes mainframe_name=$system_0830_0222 ..
AC../validation_excluded_categories=huge_job ..
AC../initiation_excluded_categories=huge_job
AC/change_attributes mainframe_name=$system_0860_0128 ..
AC../validation_excluded_categories=tape_job ..
AC../initiation_excluded_categories=tape_job
AC/quit
AS/quit save_change=true
```

If you submit a job using the following form of the LOGIN command, the job is assigned the categories LONG_JOB and SMALL_JOB. These categories only fit the LONG_SMALLER job class; therefore, the job is placed into that job class. You can submit the job on either mainframe since its categories are compatible with the membership requirements for both.

```
/login login_user=mlb password=mlbx login_family=nve ..
../cpu_time_limit=30 maximum_working_set=100 job_class=automatic
```

In this example, note that the CYBER 860 cannot allow tape jobs, since this category of jobs was excluded from the CYBER 860 as a valid category. To allow jobs with that category to be submitted on the CYBER 860, you must reenter the ADMINISTER_SCHEDULING utility and remove the category from the VALIDATION_EXCLUDED_CATEGORIES entry. You then must activate the new profile.

If you do not want jobs with the LONG_JOB category to be initiated on the CYBER 860, you can dynamically change the scheduling profile by entering the following commands on the CYBER 860:

```
/manage_active_scheduling
MAS/add_job_category_entry initiation_excluded_categories=long_job
MAS/quit
/
```

NOTE

The ADMINISTER_SCHEDULING and MANAGE_ACTIVE_SCHEDULING utilities work with a copy of the scheduling profile. The copy, with changes, is not written to the RESULT file (that is, is not made permanent) until you do one of the following:

- Enter a QUIT command with SAVE_CHANGE=TRUE in effect.
- Issue a WRITE_PROFILE subcommand under the ADMINISTER_SCHEDULING utility.

Example Set 4: Controlling File Transfer

The following examples show how you can use job categories to control system usage. For instance, installations that support the transfer of permanent files between mainframes via the Permanent File Transfer Facility (PTF) will probably want to create a job class and an associated service class that provide a relatively high level of service to this class of jobs. Since the file transfer server job runs under the initiating user name, the file transfer job class name must be valid to all users using PTF. Job categories can be created that prevent users from using the file transfer job class outside of PTF.

In this example, a job category called FILE_TRANSFER is created and assigned to all jobs submitted to the system through the file transfer server subsystem. The REQUIRED_CATEGORIES parameter of the CHANGE_ATTRIBUTE subcommand restricts membership to jobs submitted from that application. Users will be unable to submit their jobs under the FILE_TRANSFER job class even when their jobs are valid for that class, since only jobs that originate from the file transfer server application are permitted membership into that job class.

```
/administer_scheduling
AS/use_profile base=my_profile
AS/create_job_category category_name=file_transfer ..
AS../originating_application_name=osa$file_transfer_server
AS/administer_job_class
AJC/create_class class_name=file_transfer
AJC/change_attribute required_categories=file_transfer
  ⋮
(additional scheduling attributes)
  ⋮
```

In the next example, a job category called FILE_TRANSFER is created and assigned to all jobs submitted to the system through the file transfer server application and the file transfer protocol application.

```
/administer_scheduling
AS/use_profile base=my_profile
AS/create_job_category category_name=file_transfer ..
AS../originating_application_name=(osa$file_transfer_server ftpd)
```

Example Set 5: Controlling Job Initiation

The following set of examples shows how you can use job categories to control the initiation of various types of jobs. In the first example, two job categories are created: SYSTEM_JOBS (defined as all jobs belong to the $SYSTEM user in the $SYSTEM family) and INTERNAL_USERS (defined as any user under the INTERNAL account name). The site can then control the initiation of any jobs with these job categories.

```
/administer_scheduling
AS/use_profile base=my_profile
AS/create_job_category category_name=system_jobs login_family=$system ..
AS../login_user=$system
AS/create_job_category category_name=internal_users login_account=internal
```

In the next example, all jobs except those to which the job category SYSTEM_JOBS has been assigned are prevented from being initiated. Typically, you would perform this action in preparation for system shutdown or the reloading of files.

```
/manage_active_scheduling
MAS/add_job_category_entry initiation_required_categories=system_jobs
```

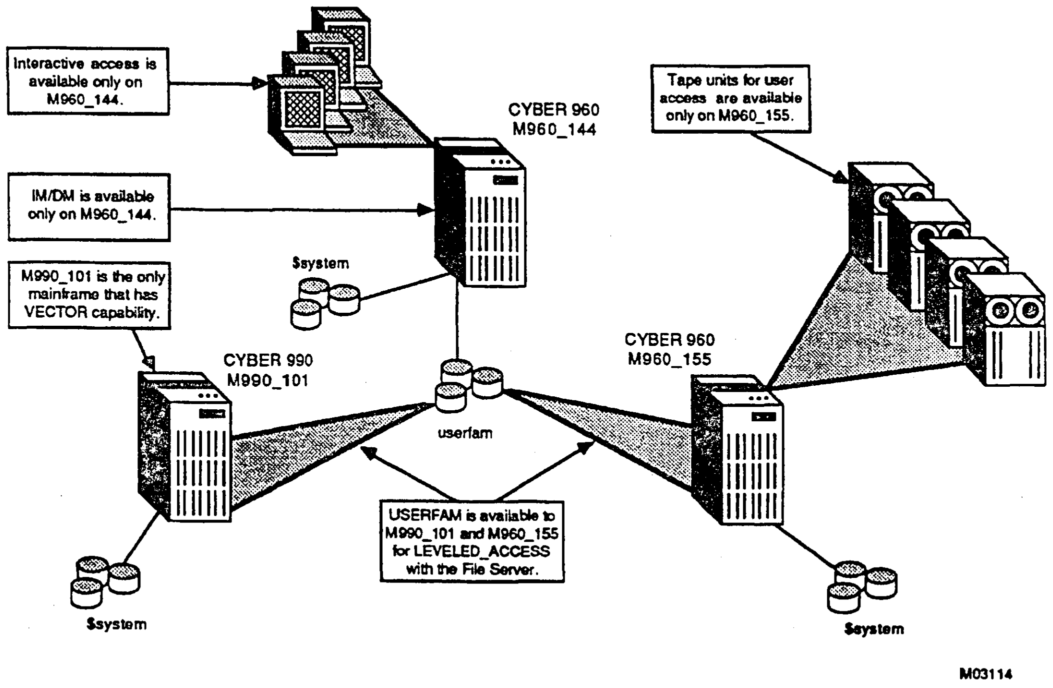
In the next example, all jobs to which the job category INTERNAL_USERS has been assigned are prevented from being initiated. You can perform this action to minimize the impact of internal jobs during peak system usage.

```
/manage_active_scheduling
MAS/add_job_category_entry initiation_excluded_categories=internal_users
```

After these actions are performed, use the DELETE_JOB_CATEGORY_ENTRY subcommand of the MANAGE_ACTIVE_SCHEDULING utility to place the system into its previous scheduling state.

Example 6: Creating a Scheduling Profile for Load Leveling

Assume your site has the following configuration:


Interactive access is available only on M960_144.
IM/DM is available only on M960_144.
M990_101 is the only mainframe that has VECTOR capability.
CYBER 960
M960_144
Tape units for user access are available only on M960_155.
$system
CYBER 990
M990_101
CYBER 960
M960_155
userfam
USERFAM is available to M990_101 and M960_155 for LEVELED_ACCESS with the File Server.
$system
$system
M03114

Defining Leveled Access for Family USERFAM

Access to the family USERFAM must be defined. The family USERFAM is available for leveled access to the M990_101 mainframe and the M960_155 mainframe. Enter the following commands at the system console of the M960_144 mainframe:

```
/manage_file_server
MANFS/change_client_access client_mainframe_identifier=$system_0990_0101 ..
MANFS../family=userfam family_access=leveled_access
MANFS/change_client_access client_mainframe_identifier=$system_0960_0155 ..
MANFS../family=userfam family_access=leveled_access
MANFS/quit
```

Defining Scheduling Controls

To define scheduling controls, use the ADMINISTER_SCHEDULING utility. Scheduling controls must be defined for all mainframes in the configuration.

```
/administer_scheduling
AS/administer_controls
AC/create_controls $system_0990_0101
AC/change_attributes $system_0990_0101 abbreviation=M990_101
AC/create_controls $system_0960_0144
AC/change_attributes $system_0960_0144 abbreviation=M960_144
AC/create_controls $system_0960_0155
AC/change_attributes $system_0960_0155 abbreviation=M960_155
AC/change_attributes all enable_job_leveling=true
AC/quit
```

Scheduling controls must be defined to control vector processing. Since the M990_101 mainframe is the only mainframe that can execute vector instructions, you must limit jobs that require vector programs to execute only on the M990_101 mainframe. This is accomplished by preventing those jobs from running on the nonvector mainframes.

```
AS/create_job_category vectors job_qualifier=(vectors, vector, v)
AS/administer_controls
AC/add_job_category_entry mainframe_name=(m960_144, m960_155) ..
AC../validation_excluded_categories=vectors ..
AC../initiation_excluded_categories=vectors
AC/quit
```

To improve vector processing throughput, a separate job class can be defined because vector jobs require a larger working set size.

```
AS/administer_job_class
AJC/create_class vectors default_values=batch
AJC/change_attributes vectors initiation_level=3 ..
AJC../maximum_working_set=(2000,500,2000) required_categories=vectors ..
AJC../initial_service_class=batch
AJC/change_attributes batch excluded_categories=vectors
```

Automatic job class selection can be used to automatically place vector jobs into the VECTOR job class and all other batch jobs into the BATCH job class.

```
AJC/change_attributes vectors selection_rank=batch ..
AJC../automatic_class_selection=true
AJC/change_attributes batch automatic_class_selection=true
AJC/quit
```

The ADMINISTER_VALIDATIONS utility should be used to change the default batch job class for users to AUTOMATIC. This allows automatic job class selection for batch jobs to take place. This utility is documented in the NOS/VE User Validation manual.

Scheduling controls must be defined for IM/DM. The M960_144 mainframe is the only mainframe that has IM/DM installed, so IM/DM jobs must be excluded from initiating on the other mainframes.

```
AS/create_job_category dm_job job_qualifier=(imdm, dm_job)
AS/administer_controls
AC/add_job_category_entry mainframe_name=(m990_101, m960_155) ..
AC../validation_excluded_categories=dm_job ..
AC../initiation_excluded_categories=dm_job
AC/quit
```

Scheduling controls must be defined to limit interactive access. Since interactive access is limited to the M960_144 mainframe, interactive mode jobs must be excluded from initiating on the other mainframes.

```
AS/administer_controls
AC/add_job_category_entry mainframe_name=(m990_101, m960_155) ..
AC../validation_excluded_categories=interactive ..
AC../initiation_excluded_categories=interactive
AC/quit
```

Scheduling controls must be defined to control jobs requiring tape units. Since the M960_155 mainframe has the only tape units available for user access, tape jobs must be excluded from initiation on the other mainframes.

```
AS/create_job_category tape_job magnetic_tape_limit=1..unlimited
AS/administer_controls
AC/add_job_category_entry mainframe_name=(m990_101, m960_144) ..
AC../validation_excluded_categories=tape_job ..
AC../initiation_excluded_categories=tape_job
AC/quit
```

Enter the following command at the system console for each of the three mainframes:

```
/change_job_attribute_defaults magnetic_tape_limit=0
```

Submitting Jobs

To execute a job that requires vectors, supply a job qualifier of VECTORS, VECTOR, or V. In this configuration, vector processing is limited to batch mode jobs only.

```
/job job_qualifier=vector
```

To execute a job that requires IM/DM, supply a job qualifier of DM_JOB or IMDM. In this configuration, this is required only for jobs that execute in batch mode since interactive access is restricted to the mainframe with IM/DM installed.

```
/job job_qualifier=imdm
```

To execute a job that requires one or more tape units, supply a magnetic tape limit when the job is submitted. In this configuration, access to tape units is restricted to batch mode jobs only.

```
/job magnetic_tape_limit=2
```

ADMINISTER_SCHEDULING Utility

The ADMINISTER_SCHEDULING utility is used to to define a scheduling profile or make changes to the profile without affecting the active scheduling tables. You can use the ADMINISTER_SCHEDULING output file to update or establish the mainframe's active scheduling tables using the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility.

To initiate the ADMINISTER_SCHEDULING utility, you must be validated for the SCHEDULING_ADMINISTRATION user capability as described in the NOS/VE User Validation manual.

We recommend that you create a single scheduling profile to encompass all mainframes at your site. This profile should define all job classes and job categories. You can then use the MANAGE_ACTIVE_SCHEDULING utility to make adjustments for any mainframe-specific requirements.

Table 4-1 summarizes the ADMINISTER_SCHEDULING subcommands.

Table 4-1. ADMINISTER_SCHEDULING Utility Subcommands

Subcommand	Description
ADMINISTER_APPLICATION	Calls the ADMINISTER_APPLICATION subutility.
ADMINISTER_CONTROLS	Calls the ADMINISTER_CONTROLS subutility.
ADMINISTER_JOB_CLASS	Calls the ADMINISTER_JOB_CLASS subutility.
ADMINISTER_SERVICE_CLASS	Calls the ADMINISTER_SERVICE_CLASS subutility.
CHANGE_JOB_CATEGORY	Changes the definition of an existing job category.
CHANGE_LIST_OPTION	Changes the default name for the OUTPUT parameter on all the ADMINISTER_SCHEDULING utility subcommands.
CREATE_DEFAULT_PROFILE	Sets the working profile to the standard scheduling profile (the profile containing the default job categories, job classes, and service classes) and establishes the result file for the utility.
CREATE_JOB_CATEGORY	Creates a job category.
DELETE_JOB_CATEGORY	Removes a job category from the scheduling profile.
DISPLAY_JOB_CATEGORY	Displays the current job category definitions and their attributes.

(Continued)

Table 4-1. ADMINISTER_SCHEDULING Utility Subcommands *(Continued)*

Subcommand	Description
DISPLAY_PROFILE_SUMMARY	Displays information about the scheduling profile.
GENERATE_PROFILE_DEFINITION	Creates a source file of subcommands that describes the current scheduling profile. You can use this source file to recreate the scheduling profile.
QUIT	Terminates the ADMINISTER_SCHEDULING utility.
USE_PROFILE	Specifies the name of a base (existing) scheduling profile to be used as the working profile for this ADMINISTER_SCHEDULING utility session. This subcommand also specifies the name of the result file to be written at the end of the utility session.
WRITE_PROFILE	Writes the current scheduling profile and all updates to an output file.

Functions Used by the Utility

Within this utility, variables exist that provide defaults for many of the subcommands. For example, you can specify a particular job class and then use it in other commands without needing to specify that name again. The utility accesses these default variables by examining a set of functions, described in table 4-2. None of the functions have parameters.

Table 4-2. Functions Used by the ADMINISTER_SCHEDULING Utility

Function Name	Description
$CURRENT_APPLICATION	Returns the name of the application that was last specified within the utility. This function remains undefined within the utility until you enter a CHANGE_ATTRIBUTE or DISPLAY_ATTRIBUTE subcommand within the ADMINISTER_APPLICATION subutility.
$CURRENT_JOB_CATEGORY	Returns the job category name that was last specified within the utility. This function remains undefined within the utility until you enter a DISPLAY_JOB_CATEGORY subcommand.
$CURRENT_JOB_CLASS	Returns the name of the job class that was last specified within the utility. This function remains undefined within the utility until you enter a CHANGE_ATTRIBUTE, DISPLAY_ATTRIBUTE, ADD_JOB_CATEGORY_ENTRY, or DELETE_JOB_CATEGORY_ENTRY subcommand within the ADMINISTER_JOB_CLASS subutility.
$CURRENT_MAINFRAME	Returns the mainframe identifier of the MAINFRAME_NAME parameter that was last specified within the utility. If no controls subcommands have been issued, this function returns the name of the system on which the utility is currently processing.
$CURRENT_PROFILE_LEVEL	Returns the name of the scheduling profile level (job category, controls, job class, service class, application) under which you are currently processing. For example, if you enter the ADMINISTER_APPLICATION subutility, the value returned by the $CURRENT_PROFILE_LEVEL function is APPLICATION. When you enter the ADMINISTER_SCHEDULING utility, the value of this function is set to JOB_CLASS.
$CURRENT_SERVICE_CLASS	Returns the name of the service class that was last specified within the utility. This function remains undefined within the utility until you enter a CHANGE_ATTRIBUTE or DISPLAY_ATTRIBUTE subcommand within the ADMINISTER_SERVICE_CLASS subutility.

Reserved Words

Several reserved words are defined for use with the ADMINISTER_SCHEDULING utility subcommands. Table 4-3 lists these words. Avoid using them when defining job classes, service classes, job qualifiers, and job category names.

Table 4-3. Reserved Words

Reserved Word	Description
ALL	Indicates that all defined names or values are to be used.
AUTOMATIC	Indicates, when specified as a job's job class name either explicitly or through the validation default, that the actual job class assigned is determined by the attributes associated with the job category. Do not use this name when defining classes, applications, or scheduling attributes. For information about establishing validation defaults, see the NOS/VE System Usage manual and the description of the LOGIN command in the NOS/VE Commands and Functions manual.
DEFAULT	Assigns the default value to an attribute. The default values for each attribute are described in table 4-13, Scheduling Attributes. If an attribute is defined as a list of one or more values, the DEFAULT keyword assigns the default value to the entire list. If one or more of the values in the list are to be assigned the default value, specifying DEFAULT for that value causes the default value to be used.
NONE	Specifies that no name or value is defined for the specific attribute.
SYSTEM_ DEFAULT	Specifies that the currently-defined system default value should be used. The system default values are defined or changed using the CHANGE_JOB_ATTRIBUTE_DEFAULTS operator command. Changing the system default value dynamically changes the value currently used for this parameter.
UNLIMITED	Indicates, for selected numeric parameters, that no limit exists for this value. In many cases, an overall system limit exists (for example, the MAXIMUM_ACTIVE_JOBS parameter is constrained by a table size of 250). In this case, the UNLIMITED keyword implies the value associated with the system limit.
UNSPECIFIED	Indicates that the specific parameter has no definition. For example, if the MAGNETIC_TAPE_LIMIT attribute is assigned a value of UNSPECIFIED, this indicates that a magnetic tape limit does not exist and knowledge of a job's tape requirements is not required.
$nnnn	Defines system names and functions. Do not use names with a dollar sign ($). Those names are reserved for system names and functions.

Optional Parameter Handling

Most subcommands contain optional parameters. If you omit an optional parameter, it is handled in one of the following ways:

- The value set by the scheduling attribute that corresponds to the subcommand parameter remains in effect.
- The most recent value specified for this parameter during the current utility session is used. Assume, for example, that you specified QUICK_BATCH for the CLASS_NAME parameter of the CHANGE_ATTRIBUTE subcommand of the ADMINISTER_JOB_CLASS subutility earlier in a utility session. The next time you enter a CHANGE_ATTRIBUTE subcommand during the same utility session, the CLASS_NAME parameter defaults to QUICK_BATCH. In this example, the most recently specified value for the CLASS_NAME parameter is returned by the $CURRENT_JOB_CLASS function. Table 4-2, Functions Used by the ADMINISTER_SCHEDULING Utility, lists the functions that return such variables.
- The ADMINISTER_SCHEDULING utility and its subutilities include a number of display subcommands. Each display subcommand has an OUTPUT parameter that specifies the output file for the subcommand. You can use the CHANGE_LIST_OPTION subcommand to specify a default output file name to be used on all subsequent display commands entered during the utility session.

Saving Changes and Activating the Updated Profile

Scheduling profile changes specified by ADMINISTER_SCHEDULING utility subcommands are written to the scheduling profile only after you do one of the following:

- Issue a WRITE_PROFILE subcommand. In this case, the changes are saved to a file you specify.
- Exit the ADMINISTER_SCHEDULING utility by issuing a QUIT subcommand with the SAVE_CHANGE parameter set to TRUE. In this case, the changes are saved on the file specified by the RESULT parameter of the USE_PROFILE or CREATE_DEFAULT_PROFILE subcommand. The default result file is $USER.SCHEDULING_PROFILE.

Under no circumstances are the scheduling tables updated by the ADMINISTER_SCHEDULING utility. To cause the saved changes to take effect, you need to activate the scheduling profile by entering the MANAGE_ACTIVE_SCHEDULING utility and issuing an ACTIVATE_PROFILE subcommand. This subcommand also updates the job scheduling tables.

ADMINISTER_SCHEDULING Command

Purpose Calls the ADMINISTER_SCHEDULING utility.

Format **ADMINISTER_SCHEDULING** or
ADMS
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.

Remarks

- To use this utility, you must be validated for the SCHEDULING_ADMINISTRATOR capability as described in the NOS/VE User Validation manual.
- When you enter this utility, the following prompt appears:

```
AS/
```

- The ADMINISTER_APPLICATION, ADMINISTER_CONTROLS, ADMINISTER_JOB_CLASS, and ADMINISTER_SERVICE_CLASS subcommands are described later in this chapter in the corresponding subutility sections.

CHANGE_JOB_CATEGORY Subcommand

Purpose Changes the definition of an existing job category.

Format **CHANGE_JOB_CATEGORY** or
CHAJC
  CATEGORY_NAME=name
  CPU_TIME_LIMIT=range of integer or *keyword*
  JOB_MODE=keyword
  JOB_QUALIFIER=list of name
  LOGIN_ACCOUNT=list of name
  LOGIN_FAMILY=list of name
  LOGIN_PROJECT=list of name
  LOGIN_USER=list of name
  MAGNETIC_TAPE_LIMIT=range of integer or *keyword*
  MAXIMUM_WORKING_SET=range of integer or *keyword*
  ORIGINATING_APPLICATION_NAME=list of name
  SRU_LIMIT=range of integer or *keyword*
  USER_JOB_NAME=list of name
  STATUS=status variable

Parameters CATEGORY_NAME or CN

Specifies the name to be assigned to this category. This parameter is required.

CPU_TIME_LIMIT or *CTL*

Specifies a CPU time limit range in seconds. The job's CPU time limit must fall within this range before the job can be assigned this category. Values you can specify for this parameter range from 1 to $MAX_INTEGER. You can also specify the keyword UNLIMITED.

JOB_MODE or *JM*

Specifies the job mode (INTERACTIVE or BATCH) required for a job if it is to be assigned this job category.

JOB_QUALIFIER or *JOB_QUALIFIERS* or *JQ*

Specifies a list of job qualifier names defined by your site. A job qualifier can restrict the mainframe on which a job can execute. The job's job qualifier name must match one of the names in this list before the job can be assigned this job category. You can set a job's default set of job qualifiers using the CHANGE_JOB_ATTRIBUTE_DEFAULTS command, described in the NOS/VE Operations manual.

LOGIN_ACCOUNT or *LOGIN_ACCOUNTS* or *LA*

Specifies a list of one or more account names. The job's account name must match one of the names in this list before the job can be assigned this job category. You may need to specify the LOGIN_FAMILY parameter to make this parameter meaningful.

LOGIN_FAMILY or *LOGIN_FAMILIES* or *LF*

Specifies a list of one or more validation family names. The job's login family name must match one of the names in this list before the job can be assigned this job category.

LOGIN_PROJECT or *LOGIN_PROJECTS* or *LP*

Specifies a list of one or more project names. The job's login project name must match one of the names in this list before the job can be assigned this job category. You may need to specify the LOGIN_FAMILY and LOGIN_ACCOUNT parameters to make this parameter meaningful.

LOGIN_USER or *LOGIN_USERS* or *LU*

Specifies a list of one or more user names. The job's login user name must match one of the names in this list before the job can be assigned this job category. You may need to specify the LOGIN_FAMILY parameter to make this parameter meaningful.

MAGNETIC_TAPE_LIMIT or *MTL*

Specifies the magnetic tape limit range. The job must have a magnetic tape limit within this range before it can be assigned this job category. Values for this parameter range from 0 to 100. The keywords UNLIMITED or UNSPECIFIED are also valid. UNSPECIFIED means that jobs whose tape requirements have not been specified either explicitly or implicitly are assigned this category.

MAXIMUM_WORKING_SET or *MAXWS*

Specifies the working set limit range. The job must have a working set limit within this range before it can be assigned this job category. Values you can specify for this parameter range from 20 to 65,000. You can also specify the keyword UNLIMITED.

ORIGINATING_APPLICATION_NAME or *ORIGINATING_APPLICATION_NAMES* or *OAN*

Specifies a list of one or more originating application names. The job's originating application name must match one of the names in this list before the job can be assigned this job category. See the Remarks section for details.

SRU_LIMIT or *SL*

Specifies the SRU limit range. The job's SRU limit must fall within this range before the job can be assigned this job category. Values you can specify for this parameter range from 1 to $MAX_INTEGER. You can also specify the keyword UNLIMITED.

USER_JOB_NAME or *USER_JOB_NAMES* or *UJN*

Specifies a list of one or more user job names. The job's user job name must match one of the names in this list before the job can be assigned this job category. You may need to specify the LOGIN_USER and LOGIN_FAMILY parameters to make this parameter meaningful.

STATUS

Returns the completion status for this subcommand.

Remarks

- This subcommand replaces the definition of the specified job category with the new definition.
- Changing the definitions of job categories causes changes in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.
- Two job category names, INTERACTIVE and BATCH, are defined for the predefined scheduling profile. These categories are defined as the job mode attributes INTERACTIVE and BATCH, respectively.
- If you specify more than one attribute on this subcommand, a job is assigned this category only if all of those attributes are held by the job. If you specify more than one value for the same attribute, this attribute is satisfied by a job if the job satisfies any of the values.
- Only system jobs (those executing under the $SYSTEM user name) can assign an originating application name to submitted jobs. The following names can be assigned:

  OSA$TIMESHARING

  Associated with interactive jobs that have entered the system through the timesharing executive. These are nondual-state jobs.

  OSA$DUAL_STATE_INTERACTIVE

  Associated with dual-state interactive jobs that have entered the system through the dual-state interactive application.

  OSA$FILE_TRANSFER_SERVER

  Associated with jobs that have been initiated through the Permanent File Transfer Facility. This represents server jobs that support the transfer of permanent files to or from a remote mainframe.

FTPD

Associated with jobs that have been initiated through the File Transfer Protocol (FTP) application. This represents server jobs that support the transfer of files using FTP to or from a remote mainframe.

OSA$SUBMIT_JOB

Associated with all jobs that have been entered into the system through the SUBMIT_JOB command, the JOB command, or the JMP$SUBMIT_JOB program interface and were not assigned an explicit originating application name.

OSA$DUAL_STATE_BATCH

Associated with all batch jobs submitted to NOS/VE from a dual-state partner.

OSA$DEADSTART

Associated with the system job only.

OSA$QUEUE_TRANSFER_SERVER

Associated with jobs that have been initiated through the Queue File Transfer Facility. This represents server jobs that support the transfer of queued files to or from a remote mainframe.

CHANGE_LIST_OPTION Subcommand

Purpose

Changes the default name used for the OUTPUT parameter on all the ADMINISTER_SCHEDULING utility subcommands.

Format

CHANGE_LIST_OPTION or
CHALO
OUTPUT=file
STATUS=status variable

Parameters

OUTPUT or *O*

Specifies the name of the file to be used as the default output file for the remaining subcommands of this utility.

STATUS

Returns the completion status for this subcommand.

Remarks

- The default output file name is $OUTPUT unless changed explicitly with this subcommand.
- The default file name created by this subcommand remains in effect until the end of the ADMINISTER_SCHEDULING utility session or until a subsequent CHANGE_LIST_OPTION subcommand is entered.

CREATE_DEFAULT_PROFILE Subcommand

Purpose Sets the working profile to the standard scheduling profile (the profile containing the default job categories, job classes, and service classes) and establishes the result file for the ADMINISTER_SCHEDULING utility.

Format **CREATE_DEFAULT_PROFILE** or
CREDP
  RESULT=file
  STATUS=status variable

Parameters *RESULT* or *R*

Specifies the name of the utility result file. This is the file to which the profile is written at the completion of the utility. The default is $USER.SCHEDULING_PROFILE.$NEXT.

STATUS

Returns the completion status for this subcommand.

Remarks The standard scheduling profile contains definitions for the BATCH and INTERACTIVE job categories and for the BATCH, INTERACTIVE, SYSTEM, MAINTENANCE, and UNASSIGNED job classes and service classes.

Examples See the example under the GENERATE_PROFILE_DEFINITION subcommand description.

CREATE_JOB_CATEGORY Subcommand

Purpose Creates a job category.

Format **CREATE_JOB_CATEGORY** or
CREJC
  CATEGORY_NAME=name
  CPU_TIME_LIMIT=range of integer or *keyword*
  JOB_MODE=keyword
  JOB_QUALIFIER=list of name
  LOGIN_ACCOUNT=list of name
  LOGIN_FAMILY=list of name
  LOGIN_PROJECT=list of name
  LOGIN_USER=list of name
  MAGNETIC_TAPE_LIMIT=range of integer or *keyword*
  MAXIMUM_WORKING_SET=range of integer or *keyword*
  ORIGINATING_APPLICATION_NAME=list of name
  SRU_LIMIT=range of integer or *keyword*
  USER_JOB_NAME=list of name
  STATUS=status variable

Parameters **CATEGORY_NAME** or **CN**

Specifies the name to be assigned to this job category. This parameter is required.

CPU_TIME_LIMIT or *CTL*

Specifies a CPU time limit range in seconds. The job must have a CPU time limit within this range before the job can be assigned this job category. Values you can specify for this parameter range from 1 to $MAX_INTEGER. You can also specify the keyword UNLIMITED.

JOB_MODE or *JM*

Specifies the job mode (INTERACTIVE or BATCH) required for the job if it is to be assigned this job category. Valid job modes are INTERACTIVE and BATCH.

JOB_QUALIFIER or *JOB_QUALIFIERS* or *JQ*

Specifies a list of job qualifier names defined by your site. A job qualifier can restrict the mainframe on which a job can execute. The job's job qualifier name must match one of the names in this list before the job can be assigned this job category. You can set a job's default set of job qualifiers using the CHANGE_JOB_ATTRIBUTE_DEFAULTS command, described in the NOS/VE Operations manual.

LOGIN_ACCOUNT or *LOGIN_ACCOUNTS* or *LA*

Specifies a list of one or more account names. The job's account name must match one of the names in this list before the job can be assigned this job category. You may need to specify the LOGIN_FAMILY parameter to make this parameter meaningful.

LOGIN_FAMILY or *LOGIN_FAMILIES* or *LF*

Specifies a list of one or more validation family names. The job's login family name must match one of the names in this list before the job can be assigned this job category.

LOGIN_PROJECT or *LOGIN_PROJECTS* or *LP*

Specifies a list of one or more project names. The job's login project name must match one of the names in this list before the job can be assigned this job category. You may need to specify the LOGIN_FAMILY and LOGIN_ACCOUNT parameters to make this parameter meaningful.

LOGIN_USER or *LOGIN_USERS* or *LU*

Specifies a list of one or more user names. The job's login user name must match one of the names in this list before the job can be assigned this job category. You may need to specify the LOGIN_FAMILY parameter to make this parameter meaningful.

MAGNETIC_TAPE_LIMIT or *MTL*

Specifies the magnetic tape limit range. The job must have a magnetic tape limit within this range before it can be assigned this job category. Values for this parameter range from 0 to 100. The keywords UNLIMITED or UNSPECIFIED are also valid. UNSPECIFIED means that jobs whose tape requirements have not been specified either explicitly or implicitly are assigned this category.

MAXIMUM_WORKING_SET or *MAXWS*

Specifies the working set limit range. The job must have a working set limit within this range before it can be assigned this job category. Values you can specify for this parameter range from 20 to 65,000. You can also specify the keyword UNLIMITED.

ORIGINATING_APPLICATION_NAME or *ORIGINATING_APPLICATION_NAMES* or *OAN*

Specifies a list of one or more originating application names. The job's originating application name must match one of the names in this list before the job can be assigned this job category. See the Remarks section for details.

SRU_LIMIT or *SL*

Specifies the SRU limit range. The job's SRU limit must fall within this range before the job can be assigned this job category. Values you can specify for this parameter range from 1 to $MAX_INTEGER. You can also specify the keyword UNLIMITED.

USER_JOB_NAME or *USER_JOB_NAMES* or *UJN*

Specifies a list of one or more user job names. The job's user job name must match one of the names in this list before the job can be assigned this job category. You may need to specify the LOGIN_FAMILY and LOGIN_USER parameters to make this parameter meaningful.

STATUS

Returns the completion status for this subcommand.

Remarks

- You can define a maximum of 64 job categories.
- Two job category names, INTERACTIVE and BATCH, are defined for the predefined scheduling profile. These categories are defined as the job mode attributes INTERACTIVE and BATCH, respectively. You can delete these job mode categories if you do not want to use them.
- Creating job categories causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.
- If you specify more than one attribute on this subcommand, a job is assigned this job category only if all of those attributes are held by the job. If you specify more than one value for the same attribute, this attribute is satisfied by a job if the job satisfies any of the values. If you specify the same job category name on two or more CREATE_JOB_CATEGORY subcommands, a job is assigned this job category if it satisfies any one of the job category definitions.
- Only system jobs (those executing under the $SYSTEM user name) can assign an originating application name to submitted jobs. The following names can be assigned:

  OSA$TIMESHARING

  Associated with interactive jobs that have entered the system through the timesharing executive. These are nondual-state jobs.

  OSA$DUAL_STATE_INTERACTIVE

  Associated with dual-state interactive jobs that have entered the system through the dual-state interactive application.

  OSA$FILE_TRANSFER_SERVER

  Associated with jobs that have been initiated through the Permanent File Transfer Facility. This represents server jobs that support the transfer of permanent files to or from a remote mainframe.

FTPD

Associated with jobs that have been initiated through the File Transfer Protocol (FTP) application. This represents server jobs that support the transfer of files using FTP to or from a remote mainframe.

OSA$SUBMIT_JOB

Associated with all jobs that have been entered into the system through the SUBMIT_JOB command, the JOB command, or the JMP$SUBMIT_JOB program interface and were not assigned an explicit originating application name.

OSA$DUAL_STATE_BATCH

Associated with all batch jobs submitted to NOS/VE from a dual-state partner.

OSA$DEADSTART

Associated with the system job only.

OSA$QUEUE_TRANSFER_SERVER

Associated with jobs that have been initiated through the Queue File Transfer Facility. This represents server jobs that support the transfer of queued files to or from a remote mainframe.

Examples In the following example, the LONG_JOB job category is created and its scheduling attributes are displayed:

```
AS/create_job_category category_name=long_job sru_limit=300..unlimited
AS/display_job_category category_name=long_job

  long_job
    Sru_Limit                         : 300..unlimited
```

DELETE_JOB_CATEGORY Subcommand

Purpose Removes a job category from the scheduling profile.

Format **DELETE_JOB_CATEGORY** or
DELJC
  CATEGORY_NAME=keyword or **list of name**
  STATUS=status variable

Parameters **CATEGORY_NAME** or **CN**

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

STATUS

Returns the completion status for this subcommand.

Remarks

- You can remove the default categories INTERACTIVE and BATCH using this subcommand. Do this by setting the EXCLUDED_CATEGORIES attribute to NONE for all job classes in the scheduling profile and then deleting the job categories.
- When a job category is deleted, all category definitions with the specified job category name are removed from the scheduling profile.
- You cannot delete a job category until references to that category are removed from the profile. You can find references to a job category in the scheduling controls and job classes within the profile.
- Deleting job categories causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

DISPLAY_JOB_CATEGORY Subcommand

Purpose Displays the current job category definitions and their attributes.

Format **DISPLAY_JOB_CATEGORY** or
DISPLAY_JOB_CATEGORIES or
DISJC
CATEGORY_NAME=keyword or *list of name*
OUTPUT=file
STATUS=status variable

Parameters *CATEGORY_NAME* or *CATEGORY_NAMES* or *CN*

Specifies the name or names of the job categories to be displayed. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_JOB_CATEGORY function.

OUTPUT or *O*

Specifies the name of the file to which the display is to be written.

STATUS

Returns the completion status for this subcommand.

Examples The following example displays information about the INTERACTIVE job category:

```
AS/display_job_category category_name=interactive

  interactive
    Job_Mode                             : interactive
```

DISPLAY_PROFILE_SUMMARY Subcommand

Purpose Displays the names of the job categories, scheduling controls, job classes, service classes, and applications in the scheduling profile.

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

Parameters *DISPLAY_OPTIONS* or *DO*

Specifies which type of information is to be displayed. You can specify the following keywords; the default is ALL:

ALL

Displays the names of all job categories, scheduling controls, job classes, service classes, and applications in the scheduling profile.

APPLICATION or A

Displays all applications in the profile.

CONTROLS or C

Displays all scheduling controls in the profile.

JOB_CATEGORY or JCA

Displays all job categories in the profile.

JOB_CLASS or JCL

Displays all job classes in the profile.

SERVICE_CLASS or SC

Displays all service classes in the profile.

OUTPUT or *O*

Specifies the name of the file to which the display information is to be written.

STATUS

Returns the completion status for this subcommand.

Remarks

- This display is identical to the display described under the DISPLAY_ PROFILE_SUMMARY subcommand of the MANAGE_ACTIVE_ SCHEDULING utility.
- The number of references to a particular part of the scheduling profile is also displayed. This count is the column of numbers shown in the following display.

Examples The following example displays a summary of a job scheduling profile:

```
AS/display_profile_summary

Summary of Job Categories
   batch                        2
   interactive                  7

Summary of Controls
   $system_0855_0109            0

Summary of Job Classes
   batch                        0
   dm_kernel                    0
   express_batch                0
   express_interactive          0
   file_transfer                0
   interactive                  0
   large_working_set_batch      0
     :
   system                       0
   maintenance                  0
   unassigned                   0

Summary of Service Classes
   system                       2
   maintenance                  1
   unassigned                   1
   batch                        1
   express_batch                3
   interactive                  1
   large_working_set            1
```

GENERATE_PROFILE_DEFINITION Subcommand

Purpose Creates a source file of subcommands that describe the current scheduling profile. You can use the source file to recreate the scheduling profile.

Format **GENERATE_PROFILE_DEFINITION** or
GENPD
GENERATE_OPTION=list of key
OUTPUT=file
STATUS=status variable

Parameters *GENERATE_OPTION* or *GO*

Specifies a keyword indicating which parameters are to be written to the output file. You can specify the following keywords; the default is BRIEF:

BRIEF or B

Writes only the changed parameters.

FULL or F

Writes all parameters.

OUTPUT or *O*

Specifies the name of the file to which the source directives are to be written. This is an SCL command file.

STATUS

Returns the completion status for this subcommand.

Remarks The source file created by this subcommand can be used as input to the ADMINISTER_SCHEDULING utility in case you need to recreate the scheduling profile. You can do this by specifying the name of the source file on an SCL INCLUDE_FILE command.

Examples The following example creates a default scheduling profile and establishes $LOCAL.PROFILE as the result file. It then creates a source file ($LOCAL.BRIEF) that can serve as input to the ADMINISTER_SCHEDULING utility for recreating the profile. Since the default value for the GENERATE_OPTION parameter is BRIEF, the file contains only the changed parameters.

```
AS/create_default_profile result=$local.profile
AS/generate_profile_definition output=$local.brief
AS/copy_file $local.brief

administer_scheduling

  create_default_profile
  administer_job_class
    change_attributes all ec=none rc=none
  quit
  delete_job_category all
```

```
create_job_category interactive ..
  Job_Mode                        = interactive

create_job_category batch ..
  Job_Mode                        = batch

administer_controls

  delete_controls $name($mainframe(identifier))
  create_controls $system_0855_0109

  change_attribute $system_0855_0109
quit

administer_service_class

  change_attribute interactive ..
    Abbreviation                  = i ..
    Guaranteed_Service_Quantum    = 9000 ..
    Maximum_Active_Jobs           = 100 ..
    Scheduling_Priority           = (8000 17000 1000 1000)

  change_attribute batch ..
    Abbreviation                  = b ..
    Guaranteed_Service_Quantum    = 20000 ..
    Maximum_Active_Jobs           = 50

  change_attribute system ..
    Abbreviation                  = s ..
    Dispatching_Control           = ((P6 unlimited 1 1)) ..
    Guaranteed_Service_Quantum    = 2000 ..
    Scheduling_Priority           = (19000 22000 1000 0)

  change_attribute maintenance ..
    Abbreviation                  = m ..
    Dispatching_Control           = ((P6 unlimited 1 1)) ..
    Guaranteed_Service_Quantum    = 9400 ..
    Maximum_Active_Jobs           = 100 ..
    Scheduling_Priority           = (8000 17000 1000 1000)

  change_attribute unassigned ..
    Abbreviation                  = u ..
    Guaranteed_Service_Quantum    = 20000 ..
    Maximum_Active_Jobs           = 0
quit
```

```
  administer_job_class

    change_attribute interactive ..
      Abbreviation                   = i ..
      Excluded_Categories            = batch ..
      Immediate_Initiation_Candidate = true ..
      Initiation_Level               = unlimited ..
      Selection_Priority             = (14000 17000 1000)

    change_attribute batch ..
      Abbreviation                   = b ..
      Excluded_Categories            = interactive ..
      Initial_Working_Set            = 200 ..
      Initiation_Age_Interval        = 60 ..
      Initiation_Level               = 10

    change_attribute system ..
      Abbreviation                   = s ..
      Excluded_Categories            = interactive ..
      Initial_Working_Set            = 80 ..
      Initiation_Level               = unlimited ..
      Selection_Priority             = (19000 22000 100)

    change_attribute maintenance ..
      Abbreviation                   = m ..
      Immediate_Initiation_Candidate = true ..
      Initiation_Age_Interval        = 60 ..
      Initiation_Level               = 10 ..
      Selection_Priority             = (14000 17000 1000)

    change_attribute unassigned ..
      Abbreviation                   = u ..
      Enable_Class_Initiation        = false ..
      Initial_Working_Set            = 200 ..
      Initiation_Age_Interval        = 60 ..
      Initiation_Level               = 10
  quit

quit
```

QUIT Subcommand

Purpose Terminates the ADMINISTER_SCHEDULING utility.

Format

QUIT or
QUI
  SAVE_CHANGE=boolean
  STATUS=status variable

Parameters *SAVE_CHANGE* or *SC*

Specifies whether all changes made under the utility are to be written to the result file. The default is TRUE.

TRUE or YES

Changes are to be saved.

FALSE or NO

Changes are not to be saved.

STATUS

Returns the completion status for this subcommand.

USE_PROFILE Subcommand

Purpose Specifies the name of a base (existing) scheduling profile to be used as the working profile for this ADMINISTER_SCHEDULING utility session. This subcommand also specifies the name of the result file to be written at the end of the utility session.

Format **USE_PROFILE** or
USEP
BASE=file
RESULT=file
STATUS=status variable

Parameters *BASE* or *B*

Specifies the name of the file containing the scheduling profile to be modified. The subcommand reads this profile and makes it the working profile to be modified by the ADMINISTER_SCHEDULING utility. If the file you specify does not exist, an error status is returned. However, the working profile is not affected. Specifying $NULL also does not change the working profile.

The default value for this parameter is $USER.SCHEDULING_PROFILE. If you omit this parameter and the $USER.SCHEDULING_PROFILE file does not exist, the standard profile is assumed.

RESULT or *R*

Specifies the name of the file to which the new scheduling profile is to be written at completion of the ADMINISTER_SCHEDULING utility. The default is the $NEXT cycle of the file specified through the BASE parameter. If you do not specify the BASE parameter, the default for the RESULT parameter is $USER.SCHEDULING_PROFILE.$NEXT.

STATUS

Returns the completion status for this subcommand.

Remarks You need to enter a USE_PROFILE subcommand before entering any subcommands that create or change job categories, job classes, or service classes.

WRITE_PROFILE Subcommand

Purpose Writes the current scheduling profile and its updates to an output file.

Format **WRITE_PROFILE** or
WRIP
RESULT=file
STATUS=status variable

Parameters *RESULT* or *R*

Specifies the name of the file to which the current scheduling profile is to be written. The default is the value of the RESULT parameter of the USE_PROFILE or CREATE_DEFAULT_PROFILE subcommand.

STATUS

Returns the completion status for this subcommand.

ADMINISTER_CONTROLS Subutility

The ADMINISTER_CONTROLS subutility of the ADMINISTER_SCHEDULING utility is used to assign values to the scheduling control attributes. These attributes control the global functions of the job scheduler; that is, they apply to all jobs regardless of job class, service class, or job category.

You can define a unique set of values for the scheduling control attributes for each mainframe in a configuration. When the scheduling profile is activated, only the scheduling control attributes defined for a particular mainframe take effect.

When you initiate this subutility (using the ADMINISTER_CONTROLS subcommand), the value of the $CURRENT_MAINFRAME function is set to the name of the mainframe on which the utility is currently executing. For a description of the $CURRENT_MAINFRAME function, see table 4-2, Functions Used by the ADMINISTER_SCHEDULING Utility.

The ADMINISTER_CONTROLS subutility includes the following subcommands (table 4-4) for managing the scheduling controls.

Table 4-4. ADMINISTER_CONTROLS Subutility Subcommands

Subcommand	Description
ADD_JOB_CATEGORY_ENTRY	Adds one or more job categories to a mainframe's list of initiation or validation categories.
CHANGE_ATTRIBUTE	Changes the value of one or more scheduling control attributes for a mainframe.
CREATE_CONTROLS	Creates a set of scheduling control attributes for a mainframe.
DELETE_CONTROLS	Deletes the scheduling control attribute definitions for a mainframe.
DELETE_JOB_CATEGORY_ENTRY	Removes one or more job categories from a mainframe's list of initiation or validation categories.
DISPLAY_ATTRIBUTE	Displays the scheduling control attribute values for a mainframe.
QUIT	Terminates the ADMINISTER_CONTROLS subutility.

ADMINISTER_CONTROLS Subcommand

Purpose Initiates the ADMINISTER_CONTROLS subutility.

Format **ADMINISTER_CONTROLS** or
ADMC
STATUS=status variable

Parameters *STATUS*

Returns the completion status for the ADMINISTER_CONTROLS subutility.

Remarks When you enter this subutility, the following prompt appears:

```
AC/
```

ADD_JOB_CATEGORY_ENTRY Subcommand

Purpose Adds one or more job categories to the list of initiation or validation categories for a mainframe.

Format **ADD_JOB_CATEGORY_ENTRY** or
ADD_JOB_CATEGORY_ENTRIES or
ADDJCE
MAINFRAME_NAME=keyword or *list of name*
INITIATION_EXCLUDED_CATEGORIES=keyword or *list of name*
INITIATION_REQUIRED_CATEGORIES=keyword or *list of name*
VALIDATION_EXCLUDED_CATEGORIES=keyword or *list of name*
VALIDATION_REQUIRED_CATEGORIES=keyword or *list of name*
STATUS=status variable

Parameters *MAINFRAME_NAME* or *MAINFRAME_NAMES* or *MN* or *CONTROLS_NAME* or *CONTROLS_NAMES* or *CN*

Specifies the 17-character name or names ($system_mmmm_ssss or its abbreviation) of the mainframe affected by the job category change. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_MAINFRAME function. If no controls subcommands have yet been issued during this utility session, this value is the name of the mainframe on which the utility is currently executing.

INITIATION_EXCLUDED_CATEGORIES or *IEC*

Specifies the job category name or names to be added to the list specified by the INITIATION_EXCLUDED_CATEGORIES scheduling attribute. You can also specify the keyword ALL. If the name already exists, an error status is returned.

INITIATION_REQUIRED_CATEGORIES or *IRC*

Specifies the job category name or names to be added to the list specified by the INITIATION_REQUIRED_CATEGORIES scheduling attribute. You can also specify the keyword ALL. If the name already exists, an error status is returned.

VALIDATION_EXCLUDED_CATEGORIES or *VEC*

Specifies the job category name or names to be added to the list specified by the VALIDATION_EXCLUDED_CATEGORIES scheduling attribute. You can also specify the keyword ALL. If the name already exists, an error status is returned.

VALIDATION_REQUIRED_CATEGORIES or *VRC*

Specifies the job category name or names to be added to the list specified by the VALIDATION_REQUIRED_CATEGORIES scheduling attribute. You can also specify the keyword ALL. If the name already exists, an error status is returned.

STATUS

Returns the completion status for this subcommand.

Remarks

- The same job category name cannot reside in both the required list (INITIATION_REQUIRED_CATEGORIES or VALIDATION_REQUIRED_CATEGORIES) and its corresponding excluded list (INITIATION_EXCLUDED_CATEGORIES or VALIDATION_EXCLUDED_CATEGORIES). If you enter the same name in both lists, the subcommand aborts and an error status is returned.
- Adding validation categories causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

CHANGE_ATTRIBUTE Subcommand

Purpose Changes the value of one or more scheduling control attributes.

Format **CHANGE_ATTRIBUTE** or
CHANGE_ATTRIBUTES or
CHAA
MAINFRAME_NAME=keyword or *list of name*
attribute=keyword or *list of name*
STATUS=status variable

Parameters *MAINFRAME_NAME* or *MAINFRAME_NAMES* or *MN* or *CONTROLS_NAME* or *CONTROLS_NAMES* or *CN*

Specifies the name or names ($system_mmmm_ssss or the abbreviation) of the mainframe to be affected by the change. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_MAINFRAME function. If no controls subcommands have yet been issued during this utility session, this value is the name of the mainframe on which the utility is currently executing.

attribute

Specifies the name or names of the scheduling control attributes that are to change, followed by the new values. Valid abbreviations can also be used. If you specify an attribute value as DEFAULT, the default value is assigned to this attribute. The scheduling control attributes are:

ABBREVIATION or A
CPU_DISPATCHING_ALLOCATION or CDA
CPU_DISPATCHING_INTERVAL or CDI
CPU_QUANTUM_TIME or CQT
DUAL_STATE_PRIORITY_CONTROL or DSPC
ENABLE_JOB_LEVELING or EJL
IDLE_DISPATCHING_QUEUE_TIME or IDQT
INITIATION_EXCLUDED_CATEGORIES or IEC
INITIATION_REQUIRED_CATEGORIES or IRC
JOB_LEVELING_INTERVAL or JLI
JOB_LEVELING_PRIORITY_BIAS or JLPB
SCHEDULING_MEMORY_LEVELS or SML
SERVICE_CALCULATION_INTERVAL or SCI
VALIDATION_REQUIRED_CATEGORIES or VRC
VALIDATION_EXCLUDED_CATEGORIES or VEC

STATUS

Returns the completion status for this subcommand.

Remarks Changing validation categories causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

CREATE_CONTROLS Subcommand

Purpose Creates a set of scheduling control attribute values.

Format
CREATE_CONTROLS or
CREC
  MAINFRAME_NAME=name
  DEFAULT_VALUES=name
  STATUS=status variable

Parameters **MAINFRAME_NAME** or **MN** or **CONTROLS_NAME** or **CN**

Specifies the name ($system_mmmm_ssss or the abbreviation) of the mainframe for which scheduling control attributes are to be established. This parameter is required.

DEFAULT_VALUES or *DV*

Specifies the name ($system_mmmm_ssss or the abbreviation) of the mainframe whose scheduling control attribute values are to be used as the initial values for this subcommand. If you do not specify this parameter, the scheduling control attribute defaults described in table 4-13, Scheduling Attributes, are used.

STATUS

Returns the completion status for this subcommand.

Remarks Creating controls causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

DELETE_CONTROLS Subcommand

Purpose Deletes the scheduling control attribute definitions for a mainframe.

Format

DELETE_CONTROLS or
DELC
  MAINFRAME_NAME=list of name
  STATUS=status variable

Parameters **MAINFRAME_NAME** or **MN** or **CONTROLS_NAME** or **CN**

Specifies the name ($system_mmmm_ssss or the abbreviation) of the mainframe whose scheduling control attributes are to be removed from the profile. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks Deleting controls causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

DELETE_JOB_CATEGORY_ENTRY Subcommand

Purpose Removes one or more job categories from a mainframe's list of initiation or validation categories.

Format **DELETE_JOB_CATEGORY_ENTRY** or
DELETE_JOB_CATEGORY_ENTRIES or
DELJCE
MAINFRAME_NAME=keyword or *list of name*
INITIATION_EXCLUDED_CATEGORIES=keyword or *list of name*
INITIATION_REQUIRED_CATEGORIES=keyword or *list of name*
VALIDATION_EXCLUDED_CATEGORIES=keyword or *list of name*
VALIDATION_REQUIRED_CATEGORIES=keyword or *list of name*
STATUS=status variable

Parameters *MAINFRAME_NAME* or *MAINFRAME_NAMES* or *CONTROLS_NAME* or *CONTROLS_NAMES* or *MN* or *CN*

Specifies the name or names ($system_mmmm_ssss or the abbreviation) of the mainframe affected by the job category change. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_MAINFRAME function. If no controls subcommands have yet been issued during this utility session, this value is the name of the mainframe on which the utility is currently executing.

INITIATION_EXCLUDED_CATEGORIES or *IEC*

Specifies the job category name or names to be removed from the list specified by the INITIATION_EXCLUDED_CATEGORIES scheduling attribute. You can also specify the keyword ALL. If the name does not exist in the list, an error status is returned.

INITIATION_REQUIRED_CATEGORIES or *IRC*

Specifies the job category name or names to be removed from the list specified by the INITIATION_REQUIRED_CATEGORIES scheduling attribute. You can also specify the keyword ALL. If the name does not exist in the list, an error status is returned.

VALIDATION_EXCLUDED_CATEGORIES or *VEC*

Specifies the job category name or names to be removed from the list specified by the VALIDATION_EXCLUDED_CATEGORIES scheduling attribute. You can also specify the keyword ALL. If the name does not exist in the list, an error status is returned.

VALIDATION_REQUIRED_CATEGORIES or *VRC*

Specifies the job category name or names to be removed from the list specified by the VALIDATION_REQUIRED_CATEGORIES scheduling attribute. You can also specify the keyword ALL. If the name does not exist in the list, an error status is returned.

STATUS

Returns the completion status for this subcommand.

Remarks

- The same job category name cannot reside in both the required list (INITIATION_REQUIRED_CATEGORIES or VALIDATION_REQUIRED_CATEGORIES) and its corresponding excluded list (INITIATION_EXCLUDED_CATEGORIES or VALIDATION_EXCLUDED_CATEGORIES). If you enter the same name in both lists, the subcommand aborts and an error status is returned.
- Deleting validation categories causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

DISPLAY_ATTRIBUTE Subcommand

Purpose Displays the scheduling control attribute values for a mainframe.

Format **DISPLAY_ATTRIBUTE** or
DISPLAY_ATTRIBUTES or
DISA
MAINFRAME_NAME=keyword or *list of name*
DISPLAY_OPTION=keyword or *list of name*
GROUP_OPTION=list of keyword
OUTPUT=file
STATUS=status variable

Parameters *MAINFRAME_NAME* or *MAINFRAME_NAMES* or *MN* or *CONTROLS_NAME* or *CONTROLS_NAMES* or *CN*

Specifies the name or names ($system_mmmm_ssss or the abbreviation) of the mainframe whose scheduling control attributes are to be displayed. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_MAINFRAME function. If no controls subcommands have yet been issued during this utility session, this value is the name of the mainframe on which the utility is currently executing.

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Specifies the attribute name or names to be displayed. You can also specify the keyword ALL. If you specify neither this parameter nor the GROUP_OPTION parameter, the display appears as if you had specified DISPLAY_OPTION=ALL. If you specify this parameter, do not specify the GROUP_OPTION parameter. You can specify the following scheduling control attributes:

ABBREVIATION or A
CPU_DISPATCHING_ALLOCATION or CDA
CPU_DISPATCHING_INTERVAL or CDI
CPU_QUANTUM_TIME or CQT
DUAL_STATE_PRIORITY_CONTROL or DSPC
ENABLE_JOB_LEVELING or EJL
IDLE_DISPATCHING_QUEUE_TIME or IDQT
INITIATION_EXCLUDED_CATEGORIES or IEC
INITIATION_REQUIRED_CATEGORIES or IRC
JOB_LEVELING_INTERVAL or JLI
JOB_LEVELING_PRIORITY_BIAS or JLPB
SCHEDULING_MEMORY_LEVELS or SML
SERVICE_CALCULATION_INTERVAL or SCI
VALIDATION_REQUIRED_CATEGORIES or VRC
VALIDATION_EXCLUDED_CATEGORIES or VEC

GROUP_OPTION or *GROUP_OPTIONS* or *GO*

Specifies that the attributes are displayed in groups. If you specify this parameter, do not specify the DISPLAY_OPTION parameter. If neither this parameter nor the DISPLAY_OPTION parameter is specified, the display appears as if you had specified DISPLAY_OPTION=ALL. You can specify the following keywords; see table 4-11, Scheduling Attribute Subsets, for more information about the groups indicated:

ALL

Displays all scheduling control attributes for a mainframe.

CONTROL or C

Displays the group control attribute subset within the set of scheduling control attributes for a mainframe.

DEFINITION or D

Displays the group definition attribute subset within the set of scheduling control attributes for a mainframe.

LIMIT or L

Displays the group limit attribute subset within the set of scheduling control attributes for a mainframe.

MEMBERSHIP or M

Displays the group membership attribute subset within the set of scheduling control attributes for a mainframe.

PRIORITY or P

Displays the group priority attribute subset within the set of scheduling control attributes for a mainframe.

STATISTIC or S

Displays the group statistic attribute subset within the set of scheduling control attributes for a mainframe.

OUTPUT or *O*

Specifies the name of the file to which the display is to be written.

STATUS

Returns the completion status for this subcommand.

Examples The following example displays the scheduling control attributes for the current mainframe ($SYSTEM_0855_0109):

```
AC/display_attribute


$system_0855_0109
  Abbreviation                        : none
  CPU_Dispatching_Allocation
    Priority                          : P1..P8
    Minimum_Percent                   : 0
    Maximum_Percent                   : 100
    Enforce_Maximum                   : false
  CPU_Dispatching_Interval            : 1
  CPU_Quantum_Time                    : 30000
  Dual_State_Priority_Control
    Dispatching_Priority              : p1..p2
    Dual_State_Priority               : 1
    Subpriority                       : 8
      -
    Dispatching_Priority              : p3..p4
    Dual_State_Priority               : 2
    Subpriority                       : 8
      -
    Dispatching_Priority              : p5..p6
    Dual_State_Priority               : 3
    Subpriority                       : 8
      -
    Dispatching_Priority              : p7..p8
    Dual_State_Priority               : 4
    Subpriority                       : 8
      -
    Dispatching_Priority              : p9..p10
    Dual_State_Priority               : 5
    Subpriority                       : 8
  Enable_Job_Leveling                 : true
  Idle_Dispatching_Queue_Time         : 360
  Initiation_Excluded_Categories      : none
  Initiation_Required_Categories      : none
  Job_Leveling_Interval               : 30
  Job_Leveling_Priority_Bias          : 0
  Scheduling_Memory_Levels
    Target                            : 60
    Thrashing                         : 20
  Service_Calculation_Interval        : 10
  Validation_Excluded_Categories      : none
  Validation_Required_Categories      : none
```

QUIT Subcommand

Purpose Terminates the ADMINISTER_CONTROLS utility.

Format

```
QUIT or
QUI
   STATUS=status variable
```

Parameters *STATUS*

Returns the completion status for this subcommand.

ADMINISTER_JOB_CLASS Subutility

The ADMINISTER_JOB_CLASS subutility of the ADMINISTER_SCHEDULING utility is used to create or delete job classes and assign values to the attributes within job classes. The following considerations affect the handling of job classes:

- The job class controls a job's initiation phase.
- The system automatically defines the following job classes and places them in the standard scheduling profile: SYSTEM, MAINTENANCE, INTERACTIVE, BATCH, and UNASSIGNED. Of these five classes, SYSTEM, MAINTENANCE, and UNASSIGNED cannot be deleted.
- When activated, jobs assigned to a job class that does not exist on the new scheduling profile are assigned to the UNASSIGNED job class.
- You cannot activate a new scheduling profile that removes a job class containing active jobs.
- There are various ways of defining default job classes for users, as well as defining default job characteristics such as CPU_TIME_LIMIT and SRU_LIMIT. Defaults can be defined using the ADMINSTER_SCHEDULING utility, the ADMINISTER_VALIDATIONS utility, and the CHANGE_JOB_ATTRIBUTE_DEFAULTS operator command. For information on how these default specifications interact with each other, see the Site Tailoring chapter of the NOS/VE System Performance and Maintenance manual, Volume 2.

The ADMINISTER_JOB_CLASS subutility includes the following subcommands (table 4-5) for managing the job classes.

Table 4-5. ADMINISTER_JOB_CLASS Subutility Subcommands

Subcommand	Description
ADD_JOB_CATEGORY_ENTRY	Adds one or more job categories to the list of required or excluded categories for a job class.
CHANGE_ATTRIBUTE	Changes the value of one or more job class attributes.
CREATE_CLASS	Creates a new job class definition within the scheduling profile.
DELETE_CLASS	Removes one or more job class definitions from the scheduling profile.
DELETE_JOB_CATEGORY_ENTRY	Removes one or more job categories from the list of required or excluded categories for a job class.
DISPLAY_ATTRIBUTE	Displays job class attribute values.
QUIT	Terminates the ADMINISTER_JOB_CLASS subutility.

ADMINISTER_JOB_CLASS Subcommand

Purpose Initiates the ADMINISTER_JOB_CLASS subutility.

Format **ADMINISTER_JOB_CLASS** or
ADMJC
STATUS=status variable

Parameters *STATUS*

Returns the completion status for the ADMINISTER_JOB_CLASS subutility.

Remarks When you enter this subutility, the following prompt appears:

```
AJC/
```

ADD_JOB_CATEGORY_ENTRY Subcommand

Purpose Adds one or more job categories to the list of required or excluded categories for a job class.

Format **ADD_JOB_CATEGORY_ENTRY** or
ADD_JOB_CATEGORY_ENTRIES or
ADDJCE
CLASS_NAME=keyword or *list of name*
EXCLUDED_CATEGORIES=keyword or *list of name*
REQUIRED_CATEGORIES=keyword or *list of name*
STATUS=status variable

Parameters *CLASS_NAME* or *CLASS_NAMES* or *CN*

Specifies the name or names of the job classes to be affected. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_JOB_CLASS function.

EXCLUDED_CATEGORIES or *EC*

Specifies the job category name or names to be added to the list specified by the EXCLUDED_CATEGORIES scheduling attribute. You can also specify the keyword ALL. If the name already exists in the EXCLUDED_CATEGORIES list, an error status is returned.

REQUIRED_CATEGORIES or *RC*

Specifies the job category name or names to be added to the list specified by the REQUIRED_CATEGORIES scheduling attribute. You can also specify the keyword ALL. If the name already exists in the REQUIRED_CATEGORIES list, an error status is returned.

STATUS

Returns the completion status for this subcommand.

Remarks

- The EXCLUDED_CATEGORIES and REQUIRED_CATEGORIES scheduling attributes control what jobs can be assigned to the specified job class.
- The same job category name cannot reside in both the EXCLUDED_CATEGORIES list and the REQUIRED_CATEGORIES list. If you enter the same name in both lists, the subcommand aborts and an error status is returned.
- Changing the categories of a job class causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

CHANGE_ATTRIBUTE Subcommand

Purpose Changes the value of one or more job class attributes.

Format **CHANGE_ATTRIBUTE** or
CHANGE_ATTRIBUTES or
CHAA
CLASS_NAME=keyword or *list of name*
attribute=keyword or *list of name*
STATUS=status variable

Parameters *CLASS_NAME* or *CLASS_NAMES* or *CN*

Specifies the name or names of the job classes to be modified. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_JOB_CLASS function.

attribute

Specifies the name or names of the job class attributes that are to change, followed by the new values. Valid abbreviations can also be used. If you specify an attribute value as DEFAULT, the default value is assigned to this attribute. You can specify the following job class attributes:

ABBREVIATION or A
AUTOMATIC_CLASS_SELECTION or ACS
CPU_TIME_LIMIT or CTL
CYCLIC_AGING_INTERVAL or CAI
DETACHED_JOB_WAIT_TIME or DJWT
ENABLE_CLASS_INITIATION or ECI
EPILOG or E
EXCLUDED_CATEGORIES or EC
IMMEDIATE_INITIATION_CANDIDATE or IIC
INITIAL_SERVICE_CLASS or ISC
INITIAL_WORKING_SET or IWS
INITIATION_AGE_INTERVAL or IAI
INITIATION_LEVEL or IL
JOB_LEVELING_PRIORITY_BIAS or JLPB
MAGNETIC_TAPE_LIMIT or MTL
MAXIMUM_WORKING_SET or MAXWS
MINIMUM_WORKING_SET or MINWS
PAGE_AGING_INTERVAL or PAI
PROLOG or P
REQUIRED_CATEGORIES or RC
SELECTION_PRIORITY or SP
SELECTION_RANK or SR
SRU_LIMIT or SL

STATUS

Returns the completion status for this subcommand.

Remarks

- If you specify more than one job class for the change, do not specify the ABBREVIATION attribute. Doing so causes an error status to be returned.
- Changing the AUTOMATIC_CLASS_SELECTION, EXCLUDED_CATEGORIES, REQUIRED_CATEGORIES, or SELECTION_RANK attributes of a job class causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

CREATE_CLASS Subcommand

Purpose Creates a new job class definition within the scheduling profile.

Format

CREATE_CLASS or
CREC
  CLASS_NAME=name
  DEFAULT_VALUES=name
  STATUS=status variable

Parameters

CLASS_NAME or CN

Specifies the name of the job class to be created. The name must be unique. This parameter is required.

DEFAULT_VALUES or DV

Specifies the name (or its abbreviation) of a previously defined job class whose attributes are to be used as the initial values for the created class. If this parameter is not specified, the job class attribute defaults defined in table 4-13, Scheduling Attributes, are used.

STATUS

Returns the completion status for this subcommand.

Remarks

- An error results if the job class name already exists in the scheduling profile.
- Do not use reserved names as names for job classes you create.
- Creating a job class causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

DELETE_CLASS Subcommand

Purpose Removes one or more previously defined job class definitions from the scheduling profile.

Format **DELETE_CLASS** or
DELC
  CLASS_NAME=list of name
  STATUS=status variable

Parameters **CLASS_NAME** or **CLASS_NAMES** or **CN**

Specifies the name or names of the job classes to be deleted. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks Deleting a job class causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

DELETE_JOB_CATEGORY_ENTRY Subcommand

Purpose Removes one or more job categories from the list of required or excluded categories for a job class.

Format **DELETE_JOB_CATEGORY_ENTRY** or
DELETE_JOB_CATEGORY_ENTRIES or
DELJCE
CLASS_NAME=keyword or *list of name*
EXCLUDED_CATEGORIES=keyword or *list of name*
REQUIRED_CATEGORIES=keyword or *list of name*
STATUS=status variable

Parameters *CLASS_NAME* or *CLASS_NAMES* or *CN*

Specifies the name or names of the job classes to be affected. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_JOB_CLASS function.

EXCLUDED_CATEGORIES or *EC*

Specifies the job category name or names to be removed from the list specified by the EXCLUDED_CATEGORY scheduling attribute. You can also specify the keyword ALL. If the name does not exist in the list, an error status is returned.

REQUIRED_CATEGORIES or *RC*

Specifies the job category name or names to be removed from the list specified by the REQUIRED_CATEGORY scheduling attribute. You can also specify the keyword ALL. If the name does not exist in the list, an error status is returned.

STATUS

Returns the completion status for this subcommand.

Remarks

- The EXCLUDED_CATEGORIES and REQUIRED_CATEGORIES scheduling attributes control what jobs can be assigned to the specified job class.
- The same job category name cannot reside in both the EXCLUDED_CATEGORIES list and the REQUIRED_CATEGORIES list. If you enter the same name in both lists, the subcommand aborts and an error status is returned.
- Changing the categories of a job class causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

DISPLAY_ATTRIBUTE Subcommand

Purpose Displays job class attribute values.

Format **DISPLAY_ATTRIBUTE** or
DISPLAY_ATTRIBUTES or
DISA
CLASS_NAME=keyword or *list of name*
DISPLAY_OPTION=keyword or *list of name*
GROUP_OPTION=list of keyword
OUTPUT=file
STATUS=status variable

Parameters *CLASS_NAME* or *CLASS_NAMES* or *CN*

Specifies the name or names of the job classes to be affected. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_JOB_CLASS function.

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Specifies the attribute name or names to be displayed. You can also specify the keyword ALL. If you specify neither this parameter nor the GROUP_ OPTION parameter, the display appears as if you had specified DISPLAY_ OPTION=ALL. If you specify this parameter, do not specify the GROUP_ OPTION parameter. You can specify the following job class attributes:

ABBREVIATION or A
AUTOMATIC_CLASS_SELECTION or ACS
CPU_TIME_LIMIT or CTL
CYCLIC_AGING_INTERVAL or CAI
DETACHED_JOB_WAIT_TIME or DJWT
ENABLE_CLASS_INITIATION or ECI
EPILOG or E
EXCLUDED_CATEGORIES or EC
IMMEDIATE_INITIATION_CANDIDATE or IIC
INITIAL_SERVICE_CLASS or ISC
INITIAL_WORKING SET or IWS
INITIATION_AGE_INTERVAL or IAI
INITIATION_LEVEL or IL
JOB_LEVELING_PRIORITY_BIAS or JLPB
MAGNETIC_TAPE_LIMIT or MTL
MAXIMUM_WORKING_SET or MAXWS
MINIMUM_WORKING_SET or MINWS
PAGE_AGING_INTERVAL or PAI
PROLOG or P
REQUIRED_CATEGORIES or RC
SELECTION_PRIORITY or SP
SELECTION_RANK or SR
SRU_LIMIT or SL

GROUP_OPTION or *GROUP_OPTIONS* or *GO*

Specifies that the attributes are displayed in groups. If you specify this parameter, do not specify the DISPLAY_OPTION parameter. If neither this parameter nor the DISPLAY_OPTION parameter is specified, the display appears as if you had specified DISPLAY_OPTION=ALL. You can specify the following keywords; see table 4-11, Scheduling Attribute Subsets, for more information about the groups indicated:

ALL

Displays all job class attributes for a job class.

CONTROL or C

Displays the group control attributes for a job class.

DEFINITION or D

Displays the group definition attributes for a job class.

LIMIT or L

Displays the group limit attributes for a job class.

MEMBERSHIP or M

Displays the group membership attributes for a job class.

PRIORITY or P

Displays the group priority attributes for a job class.

STATISTIC or S

Displays the group statistics attributes for a job class.

OUTPUT or *O*

Specifies the name of the file to which the display is to be written.

STATUS

Returns the completion status for this subcommand.

Remarks The information displayed by this subcommand consists of the attribute name or names followed by their data values.

Examples The following example displays the scheduling attributes of a hypothetical FILE_TRANSFER job class:

```
AJC/display_attributes class_name=file_transfer

file_transfer
  Abbreviation                          : ft
  Automatic_Class_Selection             : false
  CPU_Time_Limit                        : unlimited
  Cyclic_Aging_Interval
    Default                             : 1000000000
    Minimum                             : 10000
    Maximum                             : 1000000000
  Detached_Job_Wait_Time
    Default                             : 3600
    Minimum                             : 0
    Maximum                             : 18000
  Enable_Class_Initiation               : true
  Epilog                                : none
  Excluded_Categories                   : interactive
  Immediate_Initiation_Candidate        : false
  Initial_Service_Class                 : file_transfer
  Initial_Working_Set                   : 65
  Initiation_Age_Interval               : 1
  Initiation_Level
    Preferred                           : 16
  Job_Leveling_Priority_Bias            : 0
  Magnetic_Tape_Limit                   : unlimited
  Maximum_Working_Set
    Default                             : 1000
    Minimum                             : 20
    Maximum                             : 1000
  Minimum_Working_Set
    Default                             : 20
    Minimum                             : 20
    Maximum                             : 1000
  Page_Aging_Interval
    Default                             : 10000
    Minimum                             : 10000
    Maximum                             : 1000000000
  Prolog                                : none
  Required_Categories                   : file_transfer
  Selection_Priority
    Initial                             : 1000
    Maximum                             : 10000
    Increment                           : 1
  SRU_Limit                             : unlimited
```

QUIT Subcommand

Purpose Terminates the ADMINISTER_JOB_CLASS subutility.

Format

```
QUIT or
QUI
   STATUS=status variable
```

Parameters *STATUS*

Returns the completion status for this subcommand.

ADMINISTER_SERVICE_CLASS Subutility

The ADMINISTER_SERVICE_CLASS subutility of the ADMINISTER_SCHEDULING utility enables you to manage service classes by creating and deleting new service classes and assigning values to the attributes within those service classes. The following considerations affect the handling of service classes:

- The service class controls the execution phase of job processing.
- The system automatically defines the following service classes and places them in the standard scheduling profile: SYSTEM, MAINTENANCE, INTERACTIVE, BATCH, and UNASSIGNED. Of these five classes, SYSTEM, MAINTENANCE, and UNASSIGNED cannot be deleted.
- You cannot activate a new scheduling profile that removes a service class containing active jobs.

The ADMINISTER_SERVICE_CLASS subutility includes the following subcommands (table 4-6) for managing the service classes.

Table 4-6. ADMINISTER_SERVICE_CLASS Subutility Subcommands

Subcommand	Description
CHANGE_ATTRIBUTE	Changes the value of one or more service class attributes.
CREATE_CLASS	Creates a new service class definition within the scheduling profile.
DELETE_CLASS	Removes one or more service class definitions from the scheduling profile.
DISPLAY_ATTRIBUTE	Displays service class attribute values.
QUIT	Terminates the ADMINISTER_SERVICE_CLASS subutility.

ADMINISTER_SERVICE_CLASS Subcommand

Purpose Initiates the ADMINISTER_SERVICE_CLASS subutility.

Format **ADMINISTER_SERVICE_CLASS** or
ADMSC
STATUS=status variable

Parameters *STATUS*

Returns the completion status for the ADMINISTER_SERVICE_CLASS subutility.

Remarks When you enter this subutility, the following prompt appears:

```
ASC/
```

CHANGE_ATTRIBUTE Subcommand

Purpose Changes the value of one or more service class attributes.

Format **CHANGE_ATTRIBUTE** or
CHANGE_ATTRIBUTES or
CHAA
CLASS_NAME=keyword or *list of name*
attribute=keyword or *list of name*
STATUS=status variable

Parameters *CLASS_NAME* or *CLASS_NAMES* or *CN*

Specifies the name or names of the service classes to be modified. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_SERVICE_CLASS function.

attribute

Specifies the name or names of the service class attributes that are to change, followed by the new values. Valid abbreviations can also be used. If you specify an attribute value as DEFAULT, the default value is assigned to this attribute. You can specify the following service class attributes:

ABBREVIATION or A
CLASS_SERVICE_THRESHOLD or CST
DISPATCHING_CONTROL or DC
GUARANTEED_SERVICE_QUANTUM or GSQ
MAXIMUM_ACTIVE_JOBS or MAXAJ
NEXT_SERVICE_CLASS or NSC
SCHEDULING_PRIORITY or SP
SERVICE_FACTORS or SF
SWAP_AGE_INTERVAL or SAI

STATUS

Returns the completion status for this subcommand.

Remarks If you specify more than one service class for the change, do not specify the ABBREVIATION attribute. Doing so causes an error status to be returned.

CREATE_CLASS Subcommand

Purpose Creates a new service class definition within the scheduling profile.

Format **CREATE_CLASS** or
CREC
  CLASS_NAME=name
  DEFAULT_VALUES=name
  STATUS=status variable

Parameters **CLASS_NAME or CN**

Specifies the name of the service class to be created. The name must be unique. This parameter is required.

DEFAULT_VALUES or *DV*

Specifies the name (or its abbreviation) of a previously defined service class whose attributes are to be used as the initial values for the created service class. If you do not specify this parameter, the service class attribute defaults defined in table 4-13, Scheduling Attributes, are used.

STATUS

Returns the completion status for this subcommand.

Remarks

- An error results if the service class name already exists in the scheduling profile.
- Do not use reserved names as names of service classes you want to create.
- Creating a service class causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

DELETE_CLASS Subcommand

Purpose Removes one or more previously defined service class definitions from the scheduling profile.

Format **DELETE_CLASS** or
DELC
  CLASS_NAME=list of name
  STATUS=status variable

Parameters **CLASS_NAME or CLASS_NAMES or CN**

Specifies the name or names of the service classes to be deleted. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks

- You cannot delete a service class if it is referenced by another service class via the NEXT_SERVICE_CLASS attribute or by a job class via the INITIAL_SERVICE_CLASS attribute.
- Deleting a service class causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

DISPLAY_ATTRIBUTE Subcommand

Purpose Displays service class attribute values.

Format **DISPLAY_ATTRIBUTE** or
DISPLAY_ATTRIBUTES or
DISA
CLASS_NAME=keyword or *list of name*
DISPLAY_OPTION=keyword or *list of name*
GROUP_OPTION=list of keyword
OUTPUT=file
STATUS=status variable

Parameters *CLASS_NAME* or *CLASS_NAMES* or *CN*

Specifies the name or names of the service classes to be affected. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_SERVICE_CLASS function.

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Specifies the attribute name or names to be displayed. You can also specify the keyword ALL. If you specify neither this parameter nor the GROUP_OPTION parameter, the display appears as if you had specified DISPLAY_OPTION=ALL. If you specify this parameter, do not specify the GROUP_OPTION parameter. You can specify the following service class attributes:

ABBREVIATION or A
CLASS_SERVICE_THRESHOLD or CST
DISPATCHING_CONTROL or DC
GUARANTEED_SERVICE_QUANTUM or GSQ
MAXIMUM_ACTIVE_JOBS or MAXAJ
NEXT_SERVICE_CLASS or NSC
SCHEDULING_PRIORITY or SP
SERVICE_FACTORS or SF
SWAP_AGE_INTERVAL or SAI

GROUP_OPTION or *GROUP_OPTIONS* or *GO*

Specifies that the attributes are displayed in groups. If you specify this parameter, do not specify the DISPLAY_OPTION parameter. If neither this parameter nor the DISPLAY_OPTION parameter is specified, the display appears as if you had specified DISPLAY_OPTION=ALL. You can specify the following keywords; see table 4-11, Scheduling Attribute Subsets, for more information about the groups indicated:

ALL

Displays all service class attributes for a service class.

CONTROL or C

Displays the group control attributes for a service class.

DEFINITION or D

Displays the group definition attributes for a service class.

LIMIT or L

Displays the group limit attributes for a service class.

MEMBERSHIP or M

Displays the group membership attributes for a service class.

PRIORITY or P

Displays the group priority attributes for a service class.

STATISTIC or S

Displays the group statistics attributes for a service class.

OUTPUT or *O*

Specifies the name of the file to which the display is to be written.

STATUS

Returns the completion status for this subcommand.

Remarks The information displayed by this subcommand consists of the attribute name or names followed by their data values.

Examples The following example displays the scheduling attributes of a hypothetical QUICK_JOB service class:

```
ASC/display_attributes class_name=quick_job

quick_job
  Abbreviation                        : qj
  Class_Service_Threshold             : unlimited
  Dispatching_Control
    [ 1]
      Dispatching_Priority            : P6
      Service_Time                    : 5000
      Minor_Timeslice                 : 1
      Major_Timeslice                 : 1
    [ 2]
      Dispatching_Priority            : P5
      Service_Time                    : unlimited
      Minor_Timeslice                 : 1
      Major_Timeslice                 : 1
  Guaranteed_Service_Quantum          : 9400
  Maximum_Active_Jobs                 : 2
  Next_Service_Class                  : none
  Scheduling_Priority
    Minimum                           : 8000
    Maximum                           : 17000
    Swap_Age_Increment                : 1000
    Ready_Task_Increment              : 1000
  Service_Factors
    CPU                               : 1
    Memory                            : 1
    Residence                         : 1
    IO                                : 1
  Swap_Age_Interval                   : 1
```

QUIT Subcommand

Purpose Terminates the ADMINISTER_SERVICE_CLASS subutility.

Format

QUIT or
QUI
  STATUS=status variable

Parameters *STATUS*

Returns the completion status for this subcommand.

ADMINISTER_APPLICATION Subutility

The ADMINISTER_APPLICATION subutility of the ADMINISTER_SCHEDULING utility is used to define special job processing characteristics for specific application programs. When a job calls an application program for which special processing has been defined, the job begins executing under the scheduling characteristics defined for the application program. When execution of the application has been completed, the job reverts to its normal scheduling characteristics. The following considerations apply to application scheduling:

- If you define application scheduling for several applications that execute in a nested manner, the scheduling is nested as follows: as each application completes, it returns to the scheduling attributes of the previous application.
- If the application begins executing from within another specially scheduled application, the attributes that are overridden by the previous application and not to be changed for the new application are reset to the values applicable to the job.
- When a job is processing under application scheduling, modifications specified by a CHANGE_JOB_ATTRIBUTE command take effect only when one of the following conditions is encountered:
  - The application completes.
  - A nested application is initiated with different application scheduling attribute values that do not include the specified attributes.
- A DISPLAY_JOB_ATTRIBUTES command displays the attribute values that are set within the job, not those values changed by the ADMINISTER_APPLICATION subutility.

The ADMINISTER_APPLICATION subutility includes the following subcommands (table 4-7) for managing the applications.

Table 4-7. ADMINISTER_APPLICATION Subutility Subcommands

Subcommand	Description
CHANGE_ATTRIBUTE	Changes the value of one or more scheduling attributes for an application.
CREATE_APPLICATION	Creates a new application scheduling definition within the scheduling profile.
DELETE_APPLICATION	Removes one or more application scheduling definitions from the scheduling profile.
DISPLAY_ATTRIBUTE	Displays application scheduling attribute values for an application.
QUIT	Terminates the ADMINISTER_APPLICATION subutility.

For information about how to improve the performance of your application programs in NOS/VE, see the NOS/VE Object Code Management manual.

ADMINISTER_APPLICATION Subcommand

Purpose Initiates the ADMINISTER_APPLICATION subutility.

Format **ADMINISTER_APPLICATION** or
ADMA
STATUS=status variable

Parameters *STATUS*

Returns the completion status for the ADMINISTER_APPLICATION subutility.

Remarks When you enter this subutility, the following prompt appears:

```
AA/
```

CHANGE_ATTRIBUTE Subcommand

Purpose Changes the value of one or more scheduling attributes for a particular application.

Format **CHANGE_ATTRIBUTE** or
CHANGE_ATTRIBUTES or
CHAA
APPLICATION_NAME=keyword or *list of name*
attribute=keyword or *list of name*
STATUS=status variable

Parameters *APPLICATION_NAME* or *APPLICATION_NAMES* or *AN*

Specifies the name or names of the applications to be modified. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_APPLICATION function.

attribute

Specifies the name or names of the application scheduling attributes that are to change, followed by the new values. Valid abbreviations can also be used. If you specify an attribute value as DEFAULT, the default value is assigned to this attribute. You can specify the following application scheduling attributes:

ENABLE_APPLICATION_SCHEDULING or EAS
SERVICE_CLASS or SC
MAXIMUM_WORKING_SET or MAXWS
MINIMUM_WORKING_SET or MINWS
PAGE_AGING_INTERVAL or PAI
CYCLIC_AGING_INTERVAL or CAI

STATUS

Returns the completion status for this subcommand.

CREATE_APPLICATION Subcommand

Purpose Creates a new application scheduling definition within the scheduling profile.

Format **CREATE_APPLICATION** or
CREA
  APPLICATION_NAME=name
  DEFAULT_VALUES=name
  STATUS=status variable

Parameters **APPLICATION_NAME** or **AN**

Specifies the name of the application for which a new scheduling definition is to be created. The name must be unique. This parameter is required. For more information about application names, see the Remarks section of this subcommand description.

DEFAULT_VALUES or *DV*

Specifies the name of a previously defined application whose scheduling definition will be used to obtain values for the scheduling attributes assigned to the new application definition. If you do not specify this parameter, the default value UNSPECIFIED is used for the application's scheduling attributes.

STATUS

Returns the completion status for this subcommand.

Remarks

- The application name should correspond to an application name defined in the module header for a program description, command procedure, or load module in an object library. Application scheduling is supported only for program descriptions, command procedures, and load modules defined as applications in an object library. See the NOS/VE Object Code Management manual for details.
- An error results if the application name already exists in the scheduling profile.
- Creating an application causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

DELETE_APPLICATION Subcommand

Purpose Removes one or more previously defined application scheduling definitions from the scheduling profile.

Format **DELETE_APPLICATION** or
DELA
  APPLICATION_NAME=list of name
  STATUS=status variable

Parameters **APPLICATION_NAME** or **APPLICATION_NAMES** or **AN**

Specifies the name or names of the applications to be deleted. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks Deleting an application causes a change in the structure of the scheduling profile. For more information about the impact of structural changes on a scheduling profile, see the description of the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility later in this chapter.

DISPLAY_ATTRIBUTE Subcommand

Purpose Displays the scheduling attributes for a particular application.

Format **DISPLAY_ATTRIBUTE** or
DISPLAY_ATTRIBUTES or
DISA
APPLICATION_NAME=keyword or *list of name*
DISPLAY_OPTION=keyword or *list of name*
GROUP_OPTION=list of keyword
OUTPUT=file
STATUS=status variable

Parameters *APPLICATION_NAME* or *APPLICATION_NAMES* or *AN*

Specifies the name or names of the applications whose scheduling attributes are to be displayed. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_APPLICATION function.

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Specifies the attribute name or names to be displayed. You can also specify the keyword ALL. If you specify neither this parameter nor the GROUP_OPTION parameter, the display appears as if you had specified DISPLAY_OPTION=ALL. If you specify this parameter, do not specify the GROUP_OPTION parameter. You can specify the following application scheduling attributes:

ENABLE_APPLICATION_SCHEDULING or EAS
SERVICE_CLASS or SC
MAXIMUM_WORKING_SET or MAXWS
MINIMUM_WORKING_SET or MINWS
PAGE_AGING_INTERVAL or PAI
CYCLIC_AGING_INTERVAL or CAI

GROUP_OPTION or *GROUP_OPTIONS* or *GO*

Specifies that the attributes are to be displayed in groups. If you specify this parameter, do not specify the DISPLAY_OPTION parameter. If you specify neither this parameter nor the DISPLAY_OPTION parameter, the display appears as if you had specified DISPLAY_OPTION=ALL. You can specify the following keywords; see table 4-11, Scheduling Attribute Subsets, for more information about the groups indicated:

ALL

Displays all scheduling attributes for an application.

CONTROL or C

Displays the group control attributes for an application.

DEFINITION or D

Displays the group definition attributes for an application.

LIMIT or L

Displays the group limit attributes for an application.

MEMBERSHIP or M

Displays the group membership attributes for an application.

STATISTIC or S

Displays the group statistics attributes for an application.

OUTPUT or *O*

Specifies the name of the file to which the display is to be written.

STATUS

Returns the completion status for this subcommand.

Remarks The information displayed by this subcommand consists of the attribute name or names followed by their data values.

Examples The following example displays the MAXIMUM_WORKING_SET and SERVICE_CLASS scheduling attributes for the TEST_PD_CREOL application:

```
AA/display_attribute application_name=test_pd_creol ..
AA../display_options=(maximum_working_set service_class)

 test_pd_creol
  Maximum_Working_Set              : 4000
  Service_Class                    : batch
```

QUIT Subcommand

Purpose Terminates the ADMINISTER_APPLICATION subutility.

Format **QUIT** or
QUI
  STATUS=status variable

Parameters *STATUS*

Returns the completion status for this subcommand.

MANAGE_ACTIVE_SCHEDULING Utility

The MANAGE_ACTIVE_SCHEDULING utility is used to make local or temporary nonstructural changes to the active scheduling profile. This scheduling profile is then used to install or update the job scheduling tables. Call this utility to display the active scheduling profile and to modify the system scheduling either directly or by activating a profile created with the ADMINISTER_SCHEDULING utility. To initiate the MANAGE_ACTIVE_SCHEDULING utility, you must be validated for the SCHEDULING_ADMINISTRATION user capability as described in the NOS/VE User Validation manual.

The MANAGE_ACTIVE_SCHEDULING utility provides commands that let you perform most system tuning and operational scheduling changes directly. Using subcommands, you can change most attributes of existing job classes, service classes, and applications. You can use subcommands to display these attributes and various statistics. Except for activating a profile, any changes you make under the MANAGE_ACTIVE_SCHEDULING utility do not take effect until you exit the utility using a QUIT subcommand with the SAVE_CHANGE parameter set to TRUE.

To perform more complicated changes (such as adding or removing job classes or service classes, scheduling control, or job categories), first use the ADMINISTER_SCHEDULING utility to modify a local version of the profile. Then activate the modified profile using the ACTIVATE_PROFILE subcommand of the MANAGE_ACTIVE_SCHEDULING utility.

The active scheduling profile is kept on the following file:

:$SYSTEM.$SYSTEM.SCHEDULING.OSF$SYSTEM_PROFILE

This file is kept on a system-critical device. At each system startup, the MANAGE_ACTIVE_SCHEDULING utility is called to build the scheduling tables from this profile. Thus, the scheduling changes are retained across deadstarts. You should never delete this file or reference it by any commands.

NOTE

At the current level of NOS/VE, the SET_JOB_CLASS_LIMITS command changes the scheduler tables directly and does not update the active scheduling profile. Consequently, these changes are not reflected in the displays under the MANAGE_ACTIVE_SCHEDULING utility and are not recovered at the next deadstart. You can correct this and any other discrepancies between the scheduler tables and the MANAGE_ACTIVE_SCHEDULING displays by entering the following subcommand:

```
activate_profile $system.scheduling.osf$system_profile
```

Because of this property, we recommend no use of the SET_JOB_CLASS_LIMITS command.

Table 4-8 describes the subcommands of the MANAGE_ACTIVE_SCHEDULING utility.

Table 4-8. MANAGE_ACTIVE_SCHEDULING Utility Subcommands

Subcommand	Description
ACTIVATE_PROFILE	Uses the specified scheduling profile to update the active scheduler tables.
ADD_JOB_CATEGORY_ ENTRY	Adds one or more job categories to the list of initiation categories for a mainframe.
CHANGE_APPLICATION	Changes the value of one or more application scheduling attributes in the active scheduling profile.
CHANGE_CONTROLS	Changes the values of one or more scheduling control attributes in the active scheduling profile.
CHANGE_JOB_CLASS	Changes the value of one or more job class attributes in the active scheduling profile.
CHANGE_LIST_OPTION	Changes the default value of the OUTPUT parameter for all MANAGE_ACTIVE_ SCHEDULING utility subcommands.
CHANGE_SERVICE_CLASS	Changes the value of one or more service class attributes in the active scheduling profile.
DELETE_JOB_CATEGORY_ ENTRY	Removes one or more job categories from the list of initiation categories for a mainframe.
DISPLAY_APPLICATION	Displays the current application scheduling attributes in the active scheduling profile.
DISPLAY_CONTROLS	Displays the scheduling control attribute values for a mainframe.
DISPLAY_JOB_CATEGORY	Displays selected job categories and their attributes.
DISPLAY_JOB_CLASS	Displays the job class attributes in the active scheduling profile.
DISPLAY_PROFILE_ SUMMARY	Displays the job categories, scheduling control attributes, job class attributes, service class attributes, and application scheduling attributes in the active scheduling profile.
DISPLAY_SERVICE_CLASS	Displays the service class attributes in the active scheduling profile.
QUIT	Terminates the MANAGE_ACTIVE_SCHEDULING utility.
WRITE_PROFILE	Writes the active scheduling profile to a specified file.

Functions Used by the Utility

Within this utility, variables exist that provide defaults for many of the subcommands. For example, you can specify a particular job class and then use it in other commands without needing to specify that name again. The utility accesses these default variables by examining a set of functions, which are described in table 4-9. None of the functions have parameters.

Table 4-9. Functions Used by the MANAGE_ACTIVE_SCHEDULING Utility

Function Name	Description
$CURRENT_APPLICATION	Returns the name of the application that was last specified within the utility. This function remains undefined within the utility until you enter a CHANGE_APPLICATION or DISPLAY_APPLICATION subcommand.
$CURRENT_JOB_CATEGORY	Returns the job category name that was last specified within the utility. This function remains undefined within the utility until you enter a DISPLAY_JOB_CATEGORY subcommand.
$CURRENT_JOB_CLASS	Returns the name of the job class that was last specified within the utility. This function remains undefined within the utility until you enter a CHANGE_JOB_CLASS or DISPLAY_JOB_CLASS subcommand.
$CURRENT_MAINFRAME	Returns the mainframe identifier of the MAINFRAME_NAME parameter that was last specified within the utility. If no controls subcommands have been issued, this function returns the name of the mainframe on which the utility is currently processing.
$CURRENT_PROFILE_LEVEL	Returns the type of the scheduling profile object (job category, controls, job class, service class, application) which you last referenced. For example, if you enter the CHANGE_APPLICATION subcommand, the value returned by the $CURRENT_PROFILE_LEVEL function is APPLICATION.
$CURRENT_SERVICE_CLASS	Returns the name of the service class that was last specified within the utility. This function remains undefined within the utility until you enter a CHANGE_SERVICE_CLASS or DISPLAY_SERVICE_CLASS subcommand.

Reserved Words

As with the ADMINISTER_SCHEDULING utility, several reserved words are defined for use with with the MANAGE_ACTIVE_SCHEDULING utility subcommands. Table 4-10 lists these words. Avoid using them when defining job classes, service classes, job qualifiers, and job category names.

Table 4-10. Reserved Words

Reserved Word	Description
ALL	Indicates that all defined names or values are to be used.
AUTOMATIC	Indicates, when specified as a job's job class name either explicitly or through the validation default, that the actual job class assigned is determined by the attributes associated with the job category. Do not use this name when defining classes, applications, or scheduling attributes. For information about establishing validation defaults, see the NOS/VE System Usage manual and the description of the LOGIN command in the NOS/VE Commands and Functions manual.
DEFAULT	Assigns the default value to an attribute. The default values for each attribute are described in table 4-13, Scheduling Attributes. If an attribute is defined as a list of one or more values, the DEFAULT keyword assigns the default value to the entire list. If one or more of the values in the list are to be assigned the default value, specifying DEFAULT for that value causes the default value to be used.
NONE	Specifies that no name or value is defined for the specific attribute.
SYSTEM_ DEFAULT	Specifies that the currently-defined system default value should be used. The system default values are defined or changed using the CHANGE_JOB_ATTRIBUTE_DEFAULTS operator command. Changing the system default value dynamically changes the value currently used for this parameter.
UNLIMITED	Indicates, for selected numeric parameters, that no limit exists for this value. In many cases, an overall system limit exists (for example, the MAXIMUM_ACTIVE_JOBS parameter is constrained by a table size of 250). In this case, the UNLIMITED keyword implies the value associated with the system limit.
UNSPECIFIED	Indicates that the specific parameter has no definition. For example, if the MAGNETIC_TAPE_LIMIT attribute is assigned a value of UNSPECIFIED, this indicates that a magnetic tape limit does not exist and knowledge of a job's tape requirements is not required.
$nnnn	Defines system names and functions. Do not use names with a dollar sign ($). Those names are reserved for system names and functions.

Optional Parameter Handling

Most subcommands contain optional parameters. If you omit an optional parameter, it is handled in one of the following ways:

- The value set by the scheduling attribute that corresponds to the subcommand parameter remains in effect.
- The most recent value specified for this parameter during the current utility session is used. Assume, for example, that you specified QUICK_BATCH for the CLASS_NAME parameter of the CHANGE_JOB_CLASS subcommand earlier in a utility session. The next time you enter a CHANGE_JOB_CLASS subcommand during the same utility session, the CLASS_NAME parameter defaults to QUICK_BATCH. In this example, the most recently specified value for the CLASS_NAME parameter is returned by the $CURRENT_JOB_CLASS function. Table 4-9, Functions Used by the MANAGE_ACTIVE_SCHEDULING Utility, lists the functions that return such variables.
- The MANAGE_ACTIVE_SCHEDULING utility includes a number of display subcommands. Each display subcommand has an OUTPUT parameter that specifies the output file for the subcommand. You can use the CHANGE_LIST_OPTION subcommand to specify a default output file name to be used on all subsequent display commands entered during the utility session.

Saving Changes and Activating the Updated Profile

Scheduling profile changes specified by MANAGE_ACTIVE_SCHEDULING utility subcommands are saved after you do one of the following:

- Issue a WRITE_PROFILE subcommand. In this case, the changes are written to a specified file (the default file is $USER.SCHEDULING_PROFILE). The job scheduling tables are not updated.
- Exit the MANAGE_ACTIVE_SCHEDULING utility by issuing a QUIT subcommand with the SAVE_CHANGE parameter set to TRUE. In this case, the changes are saved on the :$SYSTEM.$SYSTEM.SCHEDULING.OSF$SYSTEM_PROFILE file. In addition, the job scheduling tables are automatically updated.

MANAGE_ACTIVE_SCHEDULING Command

Purpose

Calls the MANAGE_ACTIVE_SCHEDULING utility.

Format

MANAGE_ACTIVE_SCHEDULING or
MANAS
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.

Remarks

- Only one copy of this utility may be in use at any given time.
- At the current level of NOS/VE, the SET_JOB_CLASS_LIMITS command changes the job scheduling tables directly and does not update the active scheduling profile. Consequently, these changes are not reflected in the displays under the MANAGE_ACTIVE_SCHEDULING utility and are not recovered at the next deadstart. You can correct this and any other discrepancies between the scheduler tables and the MANAGE_ACTIVE_SCHEDULING utility displays by entering the following subcommand:

  ```
  activate_profile $system.scheduling.osf$system_profile
  ```

  Because of this property, we recommend either limited or no use of the SET_JOB_CLASS_LIMITS command.
- When you enter this utility, the following prompt appears:

  ```
  MAS/
  ```

ACTIVATE_PROFILE Subcommand

Purpose Activates the specified scheduling profile created with the ADMINISTER_SCHEDULING utility. The new scheduling profile updates the active scheduler tables within the system.

Format

ACTIVATE_PROFILE or
ACTP
  PROFILE=file
  ENABLE_JOB_RECLASSIFICATION=boolean
  OUTPUT=file
  STATUS=status variable

Parameters

PROFILE or **P**

Specifies the name of the file containing the scheduling profile to be activated. This file must have been created through the ADMINISTER_SCHEDULING utility. This parameter is required.

ENABLE_JOB_RECLASSIFICATION or *EJR*

Specifies whether resubmission of jobs is allowed. The default is FALSE.

TRUE

Allows job resubmission.

FALSE

Does not allow job resubmission.

If the new profile is structurally different from the active profile, it may be necessary to resubmit (reclassify) some or all of the jobs. If this is true and if this parameter is FALSE, an error status is returned and the ACTIVATE_PROFILE subcommand terminates without activating the new profile.

OUTPUT or *O*

Specifies the name of the output file for this subcommand. Information about the differences between the new profile and active profile is written to this file. This file also contains information about the state of all jobs reclassified (resubmitted) because of activating the profile.

STATUS

Returns the completion status for this subcommand.

Remarks

- If job load leveling is active, load leveling activities are terminated and cannot be reestablished until all mainframes in the load leveling environment activate structurally identical scheduling profiles. The VEDISPLAY File Server Display indicates a profile mismatch if load leveling is still enabled.

- The maximum number of job classes and service classes that can be created in the job scheduling tables is limited by values specified during system deadstart for the MAXIMUM_JOB_CLASSES and MAXIMUM_SERVICE_CLASSES system attributes. Refer to chapter 2, Adjusting System Attributes, for more information about these attributes.

- If the specified scheduling profile has structurally changed from the active scheduling profile, the ACTIVATE_PROFILE subcommand cannot execute unless the ENABLE_JOB_RECLASSIFICATION=TRUE parameter is specified. Structural changes include additions, deletions, or changes to the job affiliation of job categories, job classes, or service classes. This is done because these jobs may be assigned membership in a different job class after the resubmission is complete. The following factors cause jobs to be resubmitted:

  - New or deleted job classes.

  - New, changed, or deleted job categories.

  - Changes made to the REQUIRED_CATEGORIES, EXCLUDED_CATEGORIES, or SELECTION_RANK attributes of one or more job classes.

  If, under the ADMINISTER_SCHEDULING utility, any of these changes are made to a scheduling profile and are subsequently recreated or changed back, the profile is still considered structurally different. Jobs can be resubmitted if the profile is activated.

- While this subcommand is executing, the submission of new jobs is delayed until the new job class structure is in place. Login of interactive jobs is also inhibited during this time. Batch jobs queued for initiation in classes that are no longer defined in the new profile are assigned to the UNASSIGNED job class. You determine what action, if any, is taken for those jobs or files that remain in that job class. You do this by issuing commands to change the scheduling attributes for the UNASSIGNED job class. The MANAGE_ACTIVE_SCHEDULING utility sets the ENABLE_CLASS_INITIATION attribute to FALSE for the UNASSIGNED job class after the resubmission process has taken place. If you want to execute jobs from the UNASSIGNED job class, you must change the ENABLE_CLASS_INITIATION attribute with the CHANGE_JOB_CLASS subcommand.

- This subcommand aborts without changing the scheduling profile when jobs are active in job classes or service classes that are no longer defined in the new profile.

- The following special consideration applies to jobs that are active under application scheduling: if application scheduling switched the service class and the service class of the job that was active before the application started no longer exists when the application completes, the job reverts to the initial service class of the job class to which the job was originally assigned.

- Jobs that are currently assigned to classes defined in the new profile and that reside in the input queue are affected as follows:

  - All jobs that remain assigned to the same job classes inherit the priority values from the new profile.

  - If a structural difference exists between the new and the active profile, jobs may be resubmitted. A job that specifies a nonexistent job class or that otherwise cannot be assigned another job class is assigned to the UNASSIGNED job class. This also occurs when a job class assignment of AUTOMATIC is specified and no class is found for the job in the new scheduling profile.

  - Jobs initiated into execution inherit the service class priority and service controls from the new profile. All other limits and controls established within the job remain unchanged.

- In a file server configuration, the same profile should be activated on the other configured mainframes. This ensures that jobs submitted on any of the mainframes to any of the served families receive the correct values.

- The active scheduling profile created by this subcommand is kept in the following file:

  :$SYSTEM.$SYSTEM.SCHEDULING.OSF$SYSTEM_PROFILE

  If this file is lost or damaged, you can restore it from a backup permanent file dump. Specify this file name on the ACTIVATE_PROFILE subcommand to refresh the scheduler tables if necessary.

  When a profile is activated, new job and service classes from the profile are assigned a class index in the system scheduler tables. A class retains the same index until it is deleted by activating a profile that does not contain this class. Deleting classes may create gaps in the sequence of class indexes, causing the size of the job class or service class tables allocated at deadstart to be larger than the site has requested. During system deadstart, an informative message is issued to the SYSTEM log. This message displays the maximum indexes for the job classes and service classes defined in the active scheduling profile.

If these maximum indexes become larger than the maximum number of classes declared by your site during system deadstart, we recommend you recreate the active scheduling permanent profile as follows:

1. Create a profile that contains only the SYSTEM, MAINTENANCE, and UNASSIGNED job classes and service classes:

```
/administer_scheduling
AS/create_default_profile result=basic_profile
AS/administer_job_class
AJC/delete_class class_name=(interactive,batch)
AJC/quit
AS/administer_service_class
ASC/delete_class class_name=(interactive,batch)
ASC/quit
AS/quit save_change=true
```

2. When job activity has completed for the day, recreate the active scheduling profile to resequence the job class and service class indexes as follows:

```
/manage_active_scheduling
MAS/write_profile result=current_profile
MAS/activate_profile profile=basic_profile ..
MAS../enable_job_reclassification=true
MAS/activate_profile profile=current_profile
MAS/quit
```

ADD_JOB_CATEGORY_ENTRY Subcommand

Purpose Adds one or more job categories to the list of initiation categories for a mainframe.

Format **ADD_JOB_CATEGORY_ENTRY** or
ADD_JOB_CATEGORY_ENTRIES or
ADDJCE
INITIATION_EXCLUDED_CATEGORIES=keyword or *list of name*
INITIATION_REQUIRED_CATEGORIES=keyword or *list of name*
STATUS=status variable

Parameters *INITIATION_EXCLUDED_CATEGORIES* or *IEC*

Specifies the job category name or names to be added to the list specified by the INITIATION_EXCLUDED_CATEGORIES scheduling attribute. You can also specify the keyword ALL. If the name already exists, an error status is returned.

INITIATION_REQUIRED_CATEGORIES or *IRC*

Specifies the job category name or names to be added to the list specified by the INITIATION_REQUIRED_CATEGORIES scheduling attribute. You can also specify the keyword ALL. If the name already exists, an error status is returned.

STATUS

Returns the completion status for this subcommand.

Remarks The same job category name cannot reside in both the INITIATION_EXCLUDED_CATEGORIES list and the INITIATION_REQUIRED_CATEGORIES list. If you enter the same name in both lists, the subcommand aborts and an error status is returned.

CHANGE_APPLICATION Subcommand

Purpose Changes the values of one or more of the application scheduling attributes in the active scheduling profile.

Format **CHANGE_APPLICATION** or
CHANGE_APPLICATIONS or
CHAA
APPLICATION_NAME=keyword or *list of name*
attribute=keyword or *list of name*
STATUS=status variable

Parameters *APPLICATION_NAME* or *APPLICATION_NAMES* or *AN*

Specifies the name or names of the applications whose attributes are to be modified. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_APPLICATION function.

attribute

Specifies the name or names of the application scheduling attributes that are to change, followed by the new values. Valid abbreviations can also be used. You can specify the following application scheduling attributes:

CYCLIC_AGING_INTERVAL or CAI
ENABLE_APPLICATION_SCHEDULING or EAS
MAXIMUM_WORKING_SET or MAXWS
MINIMUM_WORKING_SET or MINWS
PAGE_AGING_INTERVAL or PAI
SERVICE_CLASS or SC

STATUS

Returns the completion status for this subcommand.

CHANGE_CONTROLS Subcommand

Purpose Changes the value of one or more of the scheduling control attributes in the active scheduling profile.

Format **CHANGE_CONTROLS** or
CHAC
attribute=keyword or *list of name*
STATUS=status variable

Parameters *attribute*

Specifies the name or names of the scheduling control attributes that are to change, followed by the new values. Valid abbreviations can also be used. You can specify the following scheduling control attributes:

ABBREVIATION or A
CPU_DISPATCHING_ALLOCATION or CDA
CPU_DISPATCHING_INTERVAL or CDI
CPU_QUANTUM_TIME or CQT
DUAL_STATE_PRIORITY_CONTROL or DSPC
ENABLE_JOB_LEVELING or EJL
IDLE_DISPATCHING_QUEUE_TIME or IDQT
INITIATION_EXCLUDED_CATEGORIES or IEC
INITIATION_REQUIRED_CATEGORIES or IRC
JOB_LEVELING_INTERVAL or JLI
JOB_LEVELING_PRIORITY_BIAS or JLPB
SCHEDULING_MEMORY_LEVELS or SML
SERVICE_CALCULATION_INTERVAL or SCI

STATUS

Returns the completion status for this subcommand.

CHANGE_JOB_CLASS Subcommand

Purpose Changes the value of one or more of the job class attributes in the active scheduling profile.

Format **CHANGE_JOB_CLASS** or
CHANGE_JOB_CLASSES or
CHAJC
CLASS_NAME=keyword or *list of name*
attribute=keyword or *list of name*
STATUS=status variable

Parameters *CLASS_NAME* or *CLASS_NAMES* or *CN*

Specifies the name or names of the job classes to be modified. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_JOB_CLASS function.

attribute

Specifies the name or names of the job class attributes that are to change, followed by the new values. Valid abbreviations can also be used. You can specify the following job class attributes:

CPU_TIME_LIMIT or CTL
CYCLIC_AGING_INTERVAL or CAI
DETACHED_JOB_WAIT_TIME or DJWT
ENABLE_CLASS_INITIATION or ECI
EPILOG or E
IMMEDIATE_INITIATION_CANDIDATES or IIC
INITIAL_SERVICE_CLASS or ISC
INITIAL_WORKING_SET or IWS
INITIATION_AGE_INTERVAL or IAI
INITIATION_LEVEL or IL
JOB_LEVELING_PRIORITY_BIAS or JLPB
MAGNETIC_TAPE_LIMIT or MTL
MAXIMUM_WORKING_SET or MAXWS
MINIMUM_WORKING_SET or MINWS
PAGE_AGING_INTERVAL or PAI
PROLOG or P
SELECTION_PRIORITY or SP
SRU_LIMIT or SL

STATUS

Returns the completion status for this subcommand.

CHANGE_LIST_OPTION Subcommand

Purpose Changes the default name for the OUTPUT parameter for all MANAGE_ACTIVE_SCHEDULING utility subcommands.

Format **CHANGE_LIST_OPTION** or
CHANGE_LIST_OPTIONS or
CHALO
OUTPUT=file
STATUS=status variable

Parameters *OUTPUT* or *O*

Specifies the name of the file to be used as the default output file on the remaining subcommands for the utility.

STATUS

Returns the completion status for this subcommand.

Remarks The default output file name is $OUTPUT unless changed explicitly with this subcommand.

CHANGE_SERVICE_CLASS Subcommand

Purpose Changes the value of one or more of the service class attributes in the active scheduling profile.

Format **CHANGE_SERVICE_CLASS** or
CHANGE_SERVICE_CLASSES or
CHASC
CLASS_NAME=keyword or *list of name*
attribute=list of value
STATUS=status variable

Parameters *CLASS_NAME* or *CLASS_NAMES* or *CN*

Specifies the name or names of the service classes to be modified. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_SERVICE_CLASS function.

attribute

Specifies the name or names of the service class attributes that are to change, followed by the new values. Valid abbreviations can also be used. You can specify the following service class attributes:

CLASS_SERVICE_THRESHOLD or CST
DISPATCHING_CONTROL or DC
GUARANTEED_SERVICE_QUANTUM or GSQ
MAXIMUM_ACTIVE_JOBS or MAXAJ
NEXT_SERVICE_CLASS or NSC
SCHEDULING_PRIORITY or SP
SERVICE_FACTORS or SF
SWAP_AGE_INTERVAL or SAI

STATUS

Returns the completion status for this subcommand.

DELETE_JOB_CATEGORY_ENTRY Subcommand

Purpose Removes one or more job categories from the list of initiation categories for a mainframe.

Format

DELETE_JOB_CATEGORY_ENTRY or
DELETE_JOB_CATEGORY_ENTRIES or
DELJCE
INITIATION_EXCLUDED_CATEGORIES=keyword or *list of name*
INITIATION_REQUIRED_CATEGORIES=keyword or *list of name*
STATUS=status variable

Parameters

INITIATION_EXCLUDED_CATEGORIES or *IEC*

Specifies the job category name or names to be deleted from the list specified by the INITIATION_EXCLUDED_CATEGORIES scheduling attribute. You can also specify the keyword ALL. If the name does not exist, an error status is returned.

INITIATION_REQUIRED_CATEGORIES or *IRC*

Specifies the job category name or names to be deleted from the list specified by the INITIATION_REQUIRED_CATEGORIES scheduling attribute. You can also specify the keyword ALL. If the name does not exist, an error status is returned.

STATUS

Returns the completion status for this subcommand.

Remarks The same job category name cannot reside in both the INITIATION_EXCLUDED_CATEGORIES list and the INITIATION_REQUIRED_CATEGORIES list. If you enter the same name in both lists, the subcommand aborts and an error status is returned.

DISPLAY_APPLICATION Subcommand

Purpose Displays the current application scheduling attributes in the active scheduling profile.

Format

DISPLAY_APPLICATION or
DISPLAY_APPLICATIONS or
DISA
  APPLICATION_NAME=keyword or *list of name*
  DISPLAY_OPTION=keyword or *list of name*
  GROUP_OPTION=list of keyword
  OUTPUT=file
  STATUS=status variable

Parameters

APPLICATION_NAME or *APPLICATION_NAMES* or *AN*

Specifies the name or names of the applications whose scheduling attributes are to be displayed. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_APPLICATION function.

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Specifies the name or names of the application scheduling attributes to be displayed. You can also specify the keyword ALL. If you specify neither this parameter nor the GROUP_OPTION parameter, the display appears as if you had specified DISPLAY_OPTION=ALL. If you specify this parameter, do not specify the GROUP_OPTION parameter. You can specify the following application scheduling attributes:

CYCLIC_AGING_INTERVAL or CAI
ENABLE_APPLICATION_SCHEDULING or EAS
MAXIMUM_WORKING_SET or MAXWS
MINIMUM_WORKING_SET or MINWS
PAGE_AGING_INTERVAL or PAI
SERVICE_CLASS or SC

GROUP_OPTION or *GROUP_OPTIONS* or *GO*

Specifies that the application scheduling attributes are to be displayed in groups. If you specify this parameter, do not specify the DISPLAY_OPTION parameter. If you specify neither this parameter nor the DISPLAY_OPTION parameter, the display appears as if you had specified DISPLAY_OPTION=ALL. You can specify the following keywords; see table 4-11, Scheduling Attribute Subsets, for more information about the groups indicated:

ALL

Displays all application scheduling attributes for an application.

CONTROL or C

Displays the group control attributes for an application.

DEFINITION or D

Displays the group definition attributes for an application.

LIMIT or L

Displays the group limit attributes for an application.

MEMBERSHIP or M

Displays the group membership attributes for an application.

STATISTIC or S

Displays the group statistics attributes for an application.

OUTPUT or *O*

Specifies the name of the file to which the display is to be written.

STATUS

Returns the completion status for this subcommand.

Remarks The information displayed by this subcommand consists of the attribute name or names followed by their data values.

Examples The following example displays the group definition and group control attributes for the TEST_PD_CREOL application:

```
MAS/display_application test_pd_creol ..
MAS../group_option=(definition control)

 test_pd_creol
  Definition Group
   Enable_Application_Scheduling     : true
  Control Group
   Cyclic_Aging_Interval              : unspecified
   Maximum_Working_Set                : 4000
   Minimum_Working_Set                : 50
   Page_Aging_Interval                : unspecified
   Service_Class                      : batch
```

DISPLAY_CONTROLS Subcommand

Purpose Displays the scheduling control attribute values for a mainframe.

Format **DISPLAY_CONTROLS** or
DISC
MAINFRAME_NAME=keyword or *list of name*
DISPLAY_OPTION=keyword or *list of name*
GROUP_OPTION=list of keyword
OUTPUT=file
STATUS=status variable

Parameters *MAINFRAME_NAME* or *MAINFRAME_NAMES* or *MN* or *CONTROL_NAME* or *CONTROL_NAMES* or *CN*

Specifies the name or names ($system_mmmm_ssss or the abbreviation) of the mainframe whose scheduling control attributes are to be displayed. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_MAINFRAME function. If no subcommands containing this parameter have yet been issued during this utility session, this value is the name of the mainframe on which the utility is currently executing.

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Specifies the name or names of the scheduling control attributes to be displayed. You can also specify the keyword ALL. If you specify neither this parameter nor the GROUP_OPTION parameter, the display appears as if you had specified DISPLAY_OPTION=ALL. If you specify this parameter, do not specify the GROUP_OPTION parameter. You can specify the following scheduling control attributes:

ABBREVIATION or A
CPU_DISPATCHING_ALLOCATION or CDA
CPU_DISPATCHING_INTERVAL or CDI
CPU_QUANTUM_TIME or CQT
DUAL_STATE_PRIORITY_CONTROL or DSPC
ENABLE_JOB_LEVELING or EJL
IDLE_DISPATCHING_QUEUE_TIME or IDQT
INITIATION_EXCLUDED_CATEGORIES or IEC
INITIATION_REQUIRED_CATEGORIES or IRC
JOB LEVELING_INTERVAL or JLI
JOB_LEVELING_PRIORITY_BIAS or JLPB
SCHEDULING_MEMORY_LEVELS or SML
SERVICE_CALCULATION_INTERVAL or SCI
VALIDATION_EXCLUDED_CATEGORIES or VEC
VALIDATION_REQUIRED_CATEGORIES or VRC

GROUP_OPTION or *GROUP_OPTIONS* or *GO*

Specifies that the scheduling control attributes are to be displayed in groups. If you specify this parameter, do not specify the DISPLAY_OPTION parameter. If you specify neither this parameter nor the DISPLAY_OPTION parameter, the display appears as if you had specified DISPLAY_OPTION=ALL. You can specify the following keywords; see table 4-11, Scheduling Attribute Subsets, for more information about the groups indicated:

ALL

Displays all scheduling control attributes for a mainframe.

CONTROL or C

Displays the group control attributes for a mainframe.

DEFINITION or D

Displays the group definition attributes for a mainframe.

LIMIT or L

Displays the group limit attributes for a mainframe.

MEMBERSHIP or M

Displays the group membership attributes for a mainframe.

PRIORITY or P

Displays the group priority attributes for a mainframe.

STATISTIC or S

Displays the group statistics attributes for a mainframe.

OUTPUT or *O*

Specifies the name of the file to which the display is to be written.

STATUS

Returns the completion status for this subcommand.

Remarks The information displayed by this subcommand consists of the attribute name or names followed by their data values.

Examples The following example displays the entire set of scheduling control attributes for a CYBER 830 with the mainframe identifier $SYSTEM_0830_0631:

```
MAS/display_controls do=all

 $system_0830_0631
  Abbreviation                          : none
  CPU_Dispatching_Allocation
    Priority                            : P1..P8
    Minimum_Percent                     : 0
    Maximum_Percent                     : 100
    Enforce_Maximum                     : false
  CPU_Dispatching_Interval              : 1
  CPU_Quantum_Time                      : 50000
  Dual_State_Priority_Control
    Dispatching_Priority                : p1..p2
    Dual_State_Priority                 : 1
    Subpriority                         : 8
      -
    Dispatching_Priority                : p3..p4
    Dual_State_Priority                 : 2
    Subpriority                         : 8
      -
    Dispatching_Priority                : p5..p6
    Dual_State_Priority                 : 3
    Subpriority                         : 8
      -
    Dispatching_Priority                : p7..p8
    Dual_State_Priority                 : 4
    Subpriority                         : 8
      -
    Dispatching_Priority                : p9..p10
    Dual_State_Priority                 : 5
    Subpriority                         : 8
  Enable_Job_Leveling                   : false
  Idle_Dispatching_Queue_Time           : 360
  Initiation_Excluded_Categories        : none
  Initiation_Required_Categories        : none
  Job_Leveling_Interval                 : 60
  Job_Leveling_Priority_Bias            : 0
  Scheduling_Memory_Levels
    Target                              : 60
    Thrashing                           : 20
  Service_Calculation_Interval          : 10
  Validation_Excluded_Categories        : none
  Validation_Required_Categories        : none
```

DISPLAY_JOB_CATEGORY Subcommand

Purpose Displays selected job categories and their attributes.

Format **DISPLAY_JOB_CATEGORY** or
DISPLAY_JOB_CATEGORIES or
DISJC
CATEGORY_NAME=name
OUTPUT=file
STATUS=status variable

Parameters *CATEGORY_NAME* or *CATEGORY_NAMES* or *CN*

Specifies the name or names of the job categories to be displayed. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_JOB_CATEGORY function.

OUTPUT or *O*

Specifies the name of the file to which the display is to be written.

STATUS

Returns the completion status for this subcommand.

Examples The following example displays attributes for the BATCH and FILE_TRANSFER job categories:

```
MAS/display_job_category category_name=(batch file_transfer)

 batch
   Job_Mode                          : batch

 file_transfer
   Originating_Application_Name      : osa$file_transfer_server
```

DISPLAY_JOB_CLASS Subcommand

Purpose Displays the job class attributes in the active scheduling profile.

Format **DISPLAY_JOB_CLASS** or
DISPLAY_JOB_CLASSES or
DISJCL
CLASS_NAME=keyword or *list of name*
DISPLAY_OPTION=keyword or *list of name*
GROUP_OPTION=list of keyword
OUTPUT=file
STATUS=status variable

Parameters *CLASS_NAME* or *CLASS_NAMES* or *CN*

Specifies the name or names of the job classes to be displayed. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_JOB_CLASS function.

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Specifies the name or names of the job class attributes to be displayed. You can also specify the keyword ALL. If you specify neither this parameter nor the GROUP_OPTION parameter, the display appears as if you had specified DISPLAY_OPTION=ALL. If you specify this parameter, do not specify the GROUP_OPTION parameter. You can specify the following job class attributes:

ABBREVIATION or A
AUTOMATIC_CLASS_SELECTION or ACS
CPU_TIME_LIMIT or CTL
CYCLIC_AGING_INTERVAL or CAI
DETACHED_JOB_WAIT_TIME or DJWT
ENABLE_CLASS_INITIATION or ECI
EPILOG or E
EXCLUDED_CATEGORIES or EC
IMMEDIATE_INITIATION_CANDIDATE or IIC
INITIAL_SERVICE_CLASS or ISC
INITIAL_WORKING_SET or IWS
INITIATED_JOBS or IJ
INITIATION_AGE_INTERVAL or IAI
INITIATION_LEVEL or IL
JOB_LEVELING_PRIORITY_BIAS or JLPB
MAGNETIC_TAPE_LIMIT or MTL
MAXIMUM_WORKING_SET or MAXWS
MINIMUM_WORKING_SET or MINWS
PAGE_AGING_INTERVAL or PAI
PROLOG or P
QUEUED_JOBS or QJ
REQUIRED_CATEGORIES or RC
SELECTION_PRIORITY or SP
SRU_LIMIT or SL

GROUP_OPTION or *GROUP_OPTIONS* or *GO*

Specifies that the job class attributes are to be displayed in groups and that the attributes of specified groups are to be displayed. If you specify this parameter, do not specify the DISPLAY_OPTION parameter. If you specify neither this parameter nor the DISPLAY_OPTION parameter, the display appears as if you had specified DISPLAY_OPTION=ALL. You can specify the following keywords; see table 4-11, Scheduling Attribute Subsets, for more information about the groups indicated:

ALL

Displays all job class attributes for a job class.

CONTROL or C

Displays the group control attributes for a job class.

DEFINITION or D

Displays the group definition attributes for a job class.

LIMIT or L

Displays the group limit attributes for a job class.

MEMBERSHIP or M

Displays the group membership attributes for a job class.

PRIORITY or P

Displays the group priority attributes for a job class.

STATISTIC or S

Displays the group statistic attributes for a job class.

OUTPUT or *O*

Specifies the name of the file to which the display is to be written.

STATUS

Returns the completion status for this subcommand.

Remarks The information displayed by this subcommand consists of the attribute name or names followed by their data values.

Examples The following example displays the group statistics attributes for the INTERACTIVE and BATCH job classes:

```
MAS/display_job_classes class_names=(interactive batch) ..
MAS../group_option=statistic

 interactive
  Statistic Group
   Initiated_Jobs                 : 11
   Queued_Jobs                    : 0

 batch
  Statistic Group
   Initiated_Jobs                 : 2
   Queued_Jobs                    : 0
```

DISPLAY_PROFILE_SUMMARY Subcommand

Purpose Displays the names of the job categories, scheduling controls, job classes, service classes, and applications in the scheduling profile.

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

Parameters *DISPLAY_OPTIONS* or *DO*

Specifies which type of information is to be displayed. You can specify the following keywords; the default is ALL:

ALL

Displays the names of all job categories, scheduling controls, job classes, service classes, and applications in the scheduling profile.

APPLICATION or A

Displays all applications in the profile.

CONTROLS or C

Displays all scheduling controls in the profile.

JOB_CATEGORY or JCA

Displays all job categories in the profile.

JOB_CLASS or JCL

Displays all job classes in the profile.

SERVICE_CLASS or SC

Displays all service classes in the profile.

OUTPUT or *O*

Specifies the name of the file to which the display information is to be written.

STATUS

Returns the completion status for this subcommand.

Remarks

- This display is identical to the display described under the DISPLAY_PROFILE_SUMMARY subcommand of the ADMINISTER_SCHEDULING utility.
- The number of references to a particular part of the scheduling profile is also displayed.

Examples The following example displays the service classes in the current scheduling profile:

```
MAS/display_profile_summary display_options=service_class

 Summary of Service Classes
    interactive                1
    batch                      2
    system                     1
    maintenance                1
    unassigned                 1
    file_transfer              1
```

DISPLAY_SERVICE_CLASS Subcommand

Purpose Displays the service class attributes in the active scheduling profile.

Format **DISPLAY_SERVICE_CLASS** or
DISPLAY_SERVICE_CLASSES or
DISA
CLASS_NAME=keyword or *list of name*
DISPLAY_OPTION=keyword or *list of name*
GROUP_OPTION=list of keyword
OUTPUT=file
STATUS=status variable

Parameters *CLASS_NAME* or *CLASS_NAMES* or *CN*

Specifies the name or names of the service classes to be displayed. You can also specify the keyword ALL. The default is the name you last specified for this parameter during the current utility session. The default value is provided by the $CURRENT_SERVICE_CLASS function.

DISPLAY_OPTION or *DISPLAY_OPTIONS* or *DO*

Specifies the name or names of the service class attributes to be displayed. You can also specify the keyword ALL. If you specify neither this parameter nor the GROUP_OPTION parameter, the display appears as if DISPLAY_OPTION=ALL had been specified. If you specify this parameter, do not specify the GROUP_OPTION parameter. You can specify the following service class attributes:

ABBREVIATION or A
ACTIVE_JOBS or AJ
CLASS_SERVICE_THRESHOLD or CST
DISPATCHING_CONTROL or DC
GUARANTEED_SERVICE_QUANTUM or GSQ
MAXIMUM_ACTIVE_JOBS or MAXAJ
NEXT_SERVICE_CLASS or NSC
QUEUED_JOBS or QJ
SCHEDULING_PRIORITY or SP
SERVICE_FACTORS or SF
SWAP_AGE_INTERVAL or SAI
SWAPPED_JOBS or SJ

GROUP_OPTION or *GROUP_OPTIONS* or *GO*

Specifies that the attributes will be displayed in groups. If you specify this parameter, do not specify the DISPLAY_OPTION parameter. If you specify neither this parameter nor the DISPLAY_OPTION parameter, the display appears as if you had specified DISPLAY_OPTION=ALL. You can specify the following keywords; see table 4-11, Scheduling Attribute Subsets, for more information about the groups indicated:

ALL

Displays all service class attributes for a service class.

CONTROL or C

Displays the group control attributes for a service class.

DEFINITION or D

Displays the group definition attributes for a service class.

LIMIT or L

Displays the group limit attributes for a service class.

MEMBERSHIP or M

Displays the group membership attributes for a service class.

PRIORITY or P

Displays the group priority attributes for a service class.

STATISTIC or S

Displays the group statistic attributes for a service class.

OUTPUT or *O*

Specifies the name of the file to which the display is written.

STATUS

Returns the completion status for this subcommand.

Remarks The information displayed by this subcommand consists of the attribute name or names followed by their data values.

Examples

- The following example displays the MAXIMUM_ACTIVE_JOB scheduling attribute for the FILE_TRANSFER service class:

```
MAS/display_service_class class_name=file_transfer ..
MAS../display_option=maximum_active_job

 file_transfer
   Maximum_Active_Jobs              : 40
```

- The following example displays the group statistics attributes for the FILE_TRANSFER service class:

```
MAS/display_service_class class_name=file_transfer ..
MAS../group_option=statistic

 file_transfer
  Statistic Group
   Active_Jobs                      : 3
   Queued_Jobs                      : 0
   Swapped_Jobs                     : 0
```

QUIT Subcommand

Purpose Terminates the MANAGE_ACTIVE_SCHEDULING utility.

Format **QUIT** or **QUI**
SAVE_CHANGE=boolean
STATUS=status variable

Parameters *SAVE_CHANGE* or *SC*

Specifies whether all changes made under the utility are to be included in the active scheduling profile. The default is TRUE.

TRUE or YES

Changes are to be saved.

FALSE or NO

Changes are not to be saved.

STATUS

Returns the completion status for this subcommand.

Remarks Issuing this subcommand with the SAVE_CHANGE=TRUE parameter in effect causes the job scheduling tables to be updated.

WRITE_PROFILE Subcommand

Purpose Writes the active scheduling profile to a specified file.

Format **WRITE_PROFILE** or
WRIP
RESULT=file
STATUS=status variable

Parameters *RESULT* or *R*

Specifies the name of the file to which the active scheduling profile is to be written. The default result file is $USER.SCHEDULING_PROFILE.

STATUS

Returns the completion status for this subcommand.

Scheduling Attributes

As defined earlier under Job Scheduling Categories and Attributes, scheduling attributes consist of the following: control attributes, job class attributes, service class attributes, and application scheduling attributes. These attribute sets can be further divided into the subsets shown in table 4-11. These subdivisions are primarily of significance when you display attribute values using any of the display subcommands provided by the MANAGE_ACTIVE_SCHEDULING utility or the ADMINISTER_SCHEDULING utility and its subutilities. When using the display subcommands, you have the choice of displaying all attributes of a particular type or of displaying one or more of the following subsets:

Table 4-11. Scheduling Attribute Subsets

Attribute Subset	Description
Group definition attributes	Establish names or abbreviations for such profile entities as job classes or service classes; accept boolean values that control such variables as membership in a job class or job initiation.
Group control attributes	Establish boundaries that control how jobs are run on the specified mainframe or in the specified class. Examples are the INITIATION_LEVEL job class attribute and the INITIATION_EXCLUDED_CATEGORIES control attribute.
Group limit attributes	Provide maximum limits for jobs assigned to that job class or service class. Job-specified limits, if lower, are observed. An example is the CPU_TIME_LIMIT job class attribute.
Group membership attributes	Control assignment of a job to a mainframe and a job class. An example is the EXCLUDED_CATEGORIES job class attribute.
Group priority attributes	Provide priority controls for the initiation and execution of jobs. Examples are the SELECTION_PRIORITY job class attribute and the SCHEDULING_PRIORITY service class attribute.
Group statistic attributes	Provide access to statistical information about the scheduling entity. An example is the INITIATED_JOBS job class attribute. Statistic attributes are used for output only, as on the DISPLAY_SERVICE_CLASS subcommand of the MANAGE_ACTIVE_SCHEDULING utility.

Not every attribute subset is represented in each set of scheduling attributes. Table 4-12 illustrates how they interrelate.

Table 4-12. Scheduling Attribute Matrix

	Scheduling Control Attributes	Job Class Attributes	Service Class Attributes	Application Scheduling Attributes
Group definition attributes	X	X	X	X
Group control attributes	X	X	X	X
Group limit attributes		X		
Group membership attributes	X	X		
Group priority attributes	X	X	X	
Group statistic attributes		X	X	

Table 4-13 lists the scheduling attributes according to the sets and subsets described in tables 4-11 and 4-12. Default values are also supplied. Detailed descriptions of the scheduling attributes are presented in alphabetical order at the end of this chapter.

Table 4-13. Scheduling Attributes

Attribute Set	Attributes	Default Value
Scheduling Control Attributes	Group Definition Attributes	
	ABBREVIATION	NONE
	CPU_QUANTUM_TIME	Model-dependent
	ENABLE_JOB_LEVELING	FALSE
	JOB_LEVELING_INTERVAL	60
	SERVICE_CALCULATION_INTERVAL	1
	Group Control Attributes	
	CPU_DISPATCHING_ALLOCATION	See footnote 1.
	CPU_DISPATCHING_INTERVAL	10
	DUAL_STATE_PRIORITY_CONTROL	See footnote 2.
	IDLE_DISPATCHING_QUEUE_TIME	360
	INITIATION_EXCLUDED_CATEGORIES	NONE
	INITIATION_REQUIRED_CATEGORIES	NONE
	SCHEDULING_MEMORY_LEVELS	(60,20)
	Group Membership Attributes	
	VALIDATION_EXCLUDED_CATEGORIES	NONE
	VALIDATION_REQUIRED_CATEGORIES	NONE
	Group Priority Attributes	
	JOB_LEVELING_PRIORITY_BIAS	0
Job Class Attributes	Group Definition Attributes	
	ABBREVIATION	NONE
	ENABLE_CLASS_INITIATION	TRUE
	EPILOG	NONE
	IMMEDIATE_INITIATION_CANDIDATE	FALSE
	INITIAL_SERVICE_CLASS	See footnote 3.
	INITIAL_WORKING_SET	65
	PROLOG	NONE

1. The default values for the CPU_DISPATCHING_ALLOCATION attribute are (P1..P8,0,100,FALSE). This represents a dispatching priority, minimum percent, maximum percent, and whether to enforce maximum percent.

2. The default values for the DUAL_STATE_PRIORITY_CONTROL attribute are shown in a table following the attribute description.

3. The default initial service class is the same as the initial service class for the DEFAULT_VALUES service class. If no DEFAULT_VALUES service class is selected, the initial service class is UNASSIGNED. In the the default scheduling profile, default initial service class has the same name as the job class.

(Continued)

Table 4-13. Scheduling Attributes *(Continued)*

Attribute Set	Attributes	Default Value
Job Class Attributes (continued)	Group Control Attributes	
	CYCLIC_AGING_INTERVAL	See footnote 1.
	INITIATION_LEVEL	20
	MAXIMUM_WORKING_SET	(1000,20,1000)[2]
	MINIMUM_WORKING_SET	(20,20,1000)[2]
	PAGE_AGING_INTERVAL	See footnote 3.
	Group Limit Attributes	
	CPU_TIME_LIMIT	UNLIMITED
	DETACHED_JOB_WAIT_TIME	(3600,0,18000)[4]
	MAGNETIC_TAPE_LIMIT	UNLIMITED
	SRU_LIMIT	UNLIMITED
	Group Membership Attributes	
	AUTOMATIC_CLASS_SELECTION	FALSE
	EXCLUDED_CATEGORIES	NONE
	REQUIRED_CATEGORIES	NONE
	SELECTION_RANK	(No default)
	Group Priority Attributes	
	INITIATION_AGE_INTERVAL	1
	JOB_LEVELING_PRIORITY_BIAS	0
	SELECTION_PRIORITY	(5000,10000,10)[5]
	Group Statistic Attributes	
	INITIATED_JOBS	Not applicable.
	QUEUED_JOBS	Not applicable.

1. The default values for the CYCLIC_AGING_INTERVAL attribute are (1000000000,10000,1000000000). This represents a default value, a minimum value, and a maximum value.

2. These default values are specified as a default value, a minimum value, and a maximum value.

3. The set of default values for the PAGE_AGING_INTERVAL attribute is (model,10000,1000000000). The first value is the actual default value and is model-dependent. The second and third values are the minimum and maximum values, respectively.

4. These default values are specified as a default value, a minimum value, and maximum value.

5. The default for the SELECTION_PRIORITY attribute is specified as an initial value, a maximum value, and an increment.

(Continued)

Table 4-13. Scheduling Attributes *(Continued)*

Attribute Set	Attributes	Default Value
Service Class Attributes	Group Definition Attributes	
	ABBREVIATION	NONE
	Group Control Attributes	
	CLASS_SERVICE_THRESHOLD	UNLIMITED
	GUARANTEED_SERVICE_QUANTUM	100
	MAXIMUM_ACTIVE_JOBS	20
	NEXT_SERVICE_CLASS	NONE
	SERVICE_FACTORS	(1,1,1,1)[1]
	Group Priority Attributes	
	DISPATCHING_CONTROL	(P5,UNLIMITED,1,1)[2]
	SCHEDULING_PRIORITY	(1000,10000,1000,0)[3]
	SWAP_AGE_INTERVAL	1
	Group Statistic Attributes	
	ACTIVE_JOBS	Not applicable.
	QUEUED_JOBS	Not applicable.
	SWAPPED_JOBS	Not applicable.
Application Scheduling Attributes	Group Definition Attributes	
	ENABLE_APPLICATION_SCHEDULING	TRUE
	Group Control Attributes	
	CYCLIC_AGING_INTERVAL	UNSPECIFIED
	MAXIMUM_WORKING_SET	UNSPECIFIED
	MINIMUM_WORKING_SET	UNSPECIFIED
	PAGE_AGING_INTERVAL	UNSPECIFIED
	SERVICE_CLASS	UNSPECIFIED

1. The default values for the SERVICE_FACTORS attribute are specified as multipliers for the CPU, memory, residence, and I/O (page faults).

2. The DISPATCHING_CONTROL attribute is specified as one to five sets of lists of four order-dependent values denoting the dispatching priority, service time, minor time slice, and major time slice. Each time slice is expressed as a multiplier of the CPU_QUANTUM_TIME control attribute.

3. The SCHEDULING_PRIORITY attribute is specified as a list of four integers denoting the minimum priority, maximum priority, swap age increment, and ready task increment.

Following are detailed descriptions of the job scheduling attributes listed earlier in table 4-13, Scheduling Attributes. The attributes are described in alphabetical order.

ABBREVIATION or A
Scheduling Control, Job Class, and Service Class Attribute

Specifies the name that can be used as an abbreviation for a mainframe name, a job class, or a service class. The abbreviation must be unique among each group of abbreviations or names used to identify mainframes, job classes, or service classes. You can use the abbreviation to specify this mainframe or class on all commands.

The first two characters of the service class abbreviation are used on the VED INITIATED_JOBS operator display. If you do not specify an abbreviation when defining a service class, no information about the service class appears on the operator display.

The names NONE, ALL, and SYSTEM_DEFAULT are reserved and cannot be used as abbreviations.

The following are the values for this attribute as defined for the five predefined job classes and service classes:

SYSTEM	S
MAINTENANCE	M
INTERACTIVE	I
BATCH	B
UNASSIGNED	U

The default value for this attribute is NONE (no abbreviation).

ACTIVE_JOBS or AJ
Service Class Attribute

Shows the number of jobs currently active in memory under a specified service class.

AUTOMATIC_CLASS_SELECTION or ACS
Job Class Attribute

Specifies a boolean value that controls the automatic assignment of jobs to the job class specified. If you specify FALSE, a job can become a member of the indicated job class only if either of the following conditions is satisfied:

- The job class name is explicitly specified in the JOB_CLASS parameter of the LOGIN, SUBMIT_JOB, or JOB command.
- The JOB_CLASS parameter is omitted and the default class for the job is the indicated class.

If you specify TRUE, a job can become a member of the job class by satisfying one of the previous conditions or by automatic selection under either of the following conditions:

- The JOB_CLASS parameter of the LOGIN, SUBMIT_JOB, or JOB commands specifies AUTOMATIC.
- The JOB_CLASS parameter is omitted and the default job class for the job is AUTOMATIC.

The job is then assigned this class if it is the first class checked for which the user is validated and the job's categories are compatible with the REQUIRED_CATEGORIES and EXCLUDED_CATEGORIES lists for the class.

For the predefined job classes, only INTERACTIVE and BATCH can be selected automatically. The other predefined job classes: SYSTEM, MAINTENANCE, and UNASSIGNED cannot be selected automatically. The default value for this attribute is FALSE.

CLASS_SERVICE_THRESHOLD or CST
Service Class Attribute

Specifies the amount of service a job must accumulate while it is a member of a specified service class before being switched to the service class specified by the NEXT_SERVICE_CLASS attribute. The amount of service is calculated according to the formula described under the SERVICE_FACTORS scheduling attribute. Values for this attribute range from 0 to 100,000,000,000. You can also specify the keyword UNLIMITED. The default value for this attribute is UNLIMITED.

CPU_DISPATCHING_ALLOCATION or CDA
Scheduling Control Attribute

Defines the minimum and maximum percent of the time specified by the CPU_DISPATCHING_INTERVAL attribute to be devoted to executing tasks of given dispatching priorities. This allows you to ensure that low-priority tasks get some CPU execution time even if higher-priority tasks are CPU-bound.

The CPU_DISPATCHING_ALLOCATION attribute consists of a list of one to eight sets, in the following form:

```
cpu_dispatching_allocation=((set1),(set2),...,(set8))
```

The variable *set* represents the following order-dependent fields:

```
set=(dispatching priority,minimum percent,maximum percent,enforce maximum
    percent)
```

The following paragraphs describe each field in more detail.

Dispatching Priority

Specifies the CPU dispatching priority or range of priorities to which the set applies. Only user dispatching priorities (P1 to P8) can be specified. System and subsystem dispatching priorities cannot be specified. All ready tasks with system or subsystem dispatching priorities continue to be dispatched using a highest-dispatching-priority-first algorithm.

Minimum Percent

Specifies the minimum percent of the time specified by the CPU_DISPATCHING_INTERVAL attribute to be devoted to tasks of the specified dispatching priority or priorities. The minimum percentage defined for the set must be less than or equal to the maximum percentage defined for the set. To ensure that lower dispatching priorities get their minimum percentage of the CPU, the sum of the minimum percentages for all user dispatching priorities can not be greater than 100. Even if a system has more than one CPU, there is still only 100 percent of dispatchable CPU execution time. Values that can be specified for this field range from 0 to 100. The default value is 0.

Maximum Percent

Specifies the maximum percent of the time specified by the CPU_DISPATCHING_INTERVAL attribute that tasks of the specified dispatching priority or priorities will be allowed to use. The maximum percentage defined for each set must be greater than or equal to the minimum percentage defined for each set. The sum of the maximum percentages for all user dispatching priorities can be greater than 100. Specifying each dispatching priority with a maximum of 100 indicates that there are no maximums. Values that can be specified for this field range from 0 to 100. The default value is 100.

Enforce Maximum Percent

Specifies whether the maximum percentage should be enforced. If TRUE, no tasks of the specified dispatching priority or priorities are selected for the rest of the CPU dispatching interval after the maximum percentage is reached. If there are no other ready tasks, the CPU goes idle. If FALSE, tasks are selected for execution after the maximum percentage has been reached if there are no ready tasks in other dispatching control table (DCT) queues that have not exceeded their maximum percentage. The default value is FALSE.

How Dispatching Allocation Works

When a site defines its own CPU dispatching allocation, the system will try to devote the minimum percentage of the CPU defined for each dispatching priority to ready tasks of that priority. Once the maximum percentage defined for a dispatching priority has been reached, tasks in that DCT queue will not be selected while there are tasks in any other DCT queue that have not reached the maximum percentage.

Defining CPU dispatching allocation is like giving a timeslice to each user dispatching priority DCT queue. The dispatcher resets the timeslice based on the time interval set by the CPU_DISPATCHING_INTERVAL attribute. If the interval is 2 seconds and a dispatching priority has been defined to have 10 as a minimum percentage of the CPU, the dispatcher will try to select tasks of that priority to execute for 200,000 microseconds out of every 2,000,000 microseconds that user dispatching priority tasks are selected to execute.

Unlike other scheduling attributes, specifying a set for a dispatching priority changes the setting for that priority only. Any dispatching priority not specified is not changed. The default values for the CPU_DISPATCHING_ALLOCATION attribute are (P1..P8,0,100,FALSE).

For additional information about selecting ready tasks within given dispatching priorities, refer to Task Selection in chapter 5, CPU Scheduling.

Dispatching Allocation on Dual-State Systems

Because the CPU_DISPATCHING_ALLOCATION attribute may allow the system to override the highest-priority-first task selection algorithm, the NOS/VE task dispatcher may select a low priority task to execute even though a higher priority task is ready. When this happens, NOS/VE's dual-state priority is determined by the dispatching priority of the highest priority task that is ready to execute. (This is the task that would have been selected if dispatching allocation was not defined.)

For example, given the following DUAL_STATE_PRIORITY_CONTROL mappings:

NOS CPU Priority	NOS/VE Dispatching Priority	Dual-State Priority	Subpriority
10–17 octal	P1–P2	1	8
20–27 octal	P3–P4	2	8
30–37 octal	P5–P6	3	8
40–47 octal	P7–P8	4	8
50–57 octal	P9–P10	5	8
60–67 octal	P11–P12	6	8
70–77 octal	P13–P14	7	8

assume the following CPU_DISPATCHING_ALLOCATION:

```
cpu_dispatching_allocation=((p2,10,100,false),(p4,20,100,false),(p6,30,100,false))
```

For this example, the CPU executes the following tasks:

NOS Task Priority	NOS/VE Task Priority	Other NOS/VE Ready Tasks	NOS - NOS/VE Dual-State Priority	System Assigned CPU
30	P6	P4, P2	3 - 3	SHARE
30	P4	P6, P2	3 - 3	SHARE
30	P2	P6, P4	3 - 3	SHARE
20	P6	P4, P2	2 - 3	NOS/VE
20	P4	P6	2 - 3	NOS/VE
20	P4	P2	2 - 2	SHARE
20	P2	NONE	2 - 1	NOS

CPU_DISPATCHING_INTERVAL or CDI Scheduling Control Attribute

Specifies the CPU execution time interval, in seconds, that is allocated for user tasks of given dispatching priorities. The time allocation and dispatching priorities are set by the CPU_DISPATCHING_ALLOCATION attribute. Values for the CPU_DISPATCHING_INTERVAL attribute range from 1 to 100. The default value is 1.

Be careful when setting the CPU_DISPATCHING_INTERVAL in a mixed interactive and batch environment. If the interval is too large, the time spent satisfying the allocation for the batch dispatching priorities will cause a noticeable delay for interactive users. A small interval requires more overhead for the system because the allocations are reset each interval.

The percentage of time allocated to batch dispatching priorities should be considered when determining the CPU_DISPATCHING_INTERVAL. For example, if the interval is 10 seconds and batch priorities are allocated a minimum of 5 percent of the interval, the maximum delay in dispatching a higher priority interactive task is 0.5 seconds. If batch priorities are allocated a minimum of 50 percent of the interval, the delay in dispatching an interactive task is as much as 5 seconds.

CPU_QUANTUM_TIME or CQT Scheduling Control Attribute

Defines a mainframe-dependent time interval in microseconds. Since this interval is used to help define the time slices described under the DISPATCHING_CONTROL attribute, changing its value affects the time slice values. Values for this attribute range from 1,000 to 100,000. The default values are mainframe-dependent; they are determined by the mainframe identifier specified by the CREATE_CONTROLS subcommand of the ADMINISTER_CONTROLS subutility. These values are as follows:

Mainframe Model	CPU_QUANTUM_TIME Default
810, 815	60,000 microseconds
825, 830, 930-11, 930A, 930C, 932-11	50,000 microseconds
835, 840, 930-31, 930B, 932-31, 932A, 932B	40,000 microseconds
840S, 845, 845S, 850, 855, 855S, 860, 865, 960-11, 960-31, 990, 992-31, 992-32, 994-31, 994-32, 995	30,000 microseconds

CPU_TIME_LIMIT or CTL Job Class Attribute

Specifies a default time, in seconds, that controls the maximum CPU time a job may accumulate (this time is also known as the job abort limit). If a job is assigned explicitly to a job class and does not specify a CPU time limit on the LOGIN, SUBMIT_JOB, or JOB command, the job is assigned the user's validated time limit or the value associated with the specified job class, whichever is smaller. If the job is assigned implicitly to a job class (that is, to job class AUTOMATIC), the job is assigned the user's validated time limit or the job attribute default, whichever is smaller. (Job attributes are characteristics that affect how the system processes a job and provide default attribute values for output files generated by the job. Job attributes are described in the NOS/VE System Usage manual.) Values for this attribute range from 1 to $MAX_INTEGER. You can also specify the names UNLIMITED and SYSTEM_DEFAULT. If you specify SYSTEM_DEFAULT, the system default CPU time limit defined by the CHANGE_JOB_ATTRIBUTE_DEFAULTS command is used. The default value for this attribute is UNLIMITED.

NOTE

This value is a default only and does not restrict a job with a CPU_TIME_LIMIT value that is greater than this default value from being a member of the job class. In order to restrict membership to a job class, use job categories.

CYCLIC_AGING_INTERVAL or CAI Job Class Attribute

Specifies a default real-time interval, in microseconds, used for aging the job's working set. If no aging has occurred before this time interval has elapsed, the job's working set is aged. The value specified can be in the range of 1,000 to 3,600,000,000.

For average jobs (that is, jobs that do not go into extended waits and jobs that do not have many asynchronous, unrelated tasks), the CYCLIC_AGING_INTERVAL attribute should be set high and the PAGE_AGING_INTERVAL attribute should be set low enough to cause aging based on the CPU time used. The PAGE_AGING_INTERVAL attribute is described later in this chapter.

For jobs that go into extended waits, such as a database manager, or for jobs that have many asynchronous, unrelated tasks, such as the system job, cyclic aging controlled by the CYCLIC_AGING_INTERVAL attribute should be used. Set the PAGE_AGING_INTERVAL attribute high and the CYCLIC_AGING_INTERVAL attribute low (for example, to 10 or 20 seconds).

This value is specified as a list of three integers, in the form:

```
(default,minimum,maximum)
```

For example:

```
CYCLIC_AGING_INTERVAL=(10000000,10000,1000000000)
```

In this example, 10 seconds is the default, 10 milliseconds is the minimum, and 1000 seconds is the maximum.

Within the range specified by the CYCLIC_AGING_INTERVAL attribute, the value can later be changed by the CHANGE_JOB_ATTRIBUTE command. The CHANGE_JOB_ATTRIBUTE command is described in the NOS/VE Commands and Functions manual.

The set of default values for this attribute is as follows:

(1000000000,10000,1000000000)

CYCLIC_AGING_INTERVAL or CAI Application Scheduling Attribute

Specifies a default real-time interval, in microseconds, used for aging a job's working set. If no aging has occurred before this time interval has elapsed, the job's working set is aged. The value specified can be in the range of 1,000 to 3,600,000,000.

If you specify this attribute while defining application scheduling under the ADMINISTER_APPLICATION subutility, the value specified remains in effect as long as the application is executing. When the application completes, the cyclic aging interval reverts to its normal value.

The default value for this attribute is UNSPECIFIED. This means that the cyclic aging interval for the job does not change while the application is executing.

DETACHED_JOB_WAIT_TIME or DJWT Job Class Attribute

Specifies the amount of time, in seconds, a detached job can remain suspended before it is terminated. The value specified can be in the range of 0 to 36,000. You can also use the keyword UNLIMITED.

This value is specified as a list of three integers, in the form:

```
(default,minimum,maximum)
```

For example:

```
DETACHED_JOB_WAIT_TIME=(600,60,3600)
```

In this example, 600 is the default, 60 is the minimum, and 3,600 is the maximum.

Within the range specified by the DETACHED_JOB_WAIT_TIME attribute, the value can later be changed by the CHANGE_JOB_ATTRIBUTE command, described in the NOS/VE Commands and Functions manual.

The set of default values for this attribute is as follows:

(3600,0,18000)

DISPATCHING_CONTROL or DC Service Class Attribute

Defines the CPU dispatching priority and time slice for a service class. All tasks of all jobs executing in the service class are assigned a dispatching priority and a time slice from the dispatching controls. You can establish dynamic dispatching for the service class by defining several sets of values for the DISPATCHING_CONTROL attribute. Under dynamic dispatching, the system periodically adjusts a job's dispatching priority according to the CPU service the job has already received. Alternatively, you can establish nondynamic dispatching by defining only one set of values for the DISPATCHING_CONTROL attribute, with the service time variable set to UNLIMITED.

The DISPATCHING_CONTROL attribute consists of a list of one to five sets, in the following form:

```
dispatching_control=((set1),(set2),...,(set5))
```

The variable *set* represents the following order-dependent fields:

```
set=(dispatching priority,service time,minor time slice,major time slice)
```

The following paragraphs describe each field in more detail.

Dispatching Priority

The dispatching priority value specifies the dispatching priority for a task. Dispatching priorities are values ranging from P1 to P10. P10 is the highest priority. The CPU dispatcher maintains a queue of ready tasks for each dispatching priority. When a task is ready, it is linked into the appropriate dispatching control table (DCT) queue. By default, a highest-priority-first task selection algorithm is used. A task is selected for CPU execution from the head of the highest priority DCT queue. A task in a lower-priority DCT queue is selected for CPU execution only when all higher-priority DCT queues are empty. See the description of the CPU_DISPATCHING_ALLOCATION attribute for information about how to set up a site-defined dispatching allocation.

For information about the relationship between NOS/VE dispatching priorities and NOS (or NOS/BE) CPU priorities, refer to the description of the DUAL_STATE_PRIORITY_CONTROL attribute.

The following are the values for the dispatching priority as defined for the five predefined service classes:

SYSTEM	P6
MAINTENANCE	P6
INTERACTIVE	P5
BATCH	P5
UNASSIGNED	P5

Service Time

The service time value specifies the amount of CPU time, in milliseconds, that a job can use while the current DISPATCHING_CONTROL attribute set is in effect. You can also specify the keyword UNLIMITED, which indicates that the dispatching controls defined by the set have no time limit. You can specify this keyword only for the last set defined.

Each set of values specified remains in effect for the duration of time specified by the service time field. When the service time value has expired for a set, the next set of values becomes effective. When the service time value for the last set has expired, the first set of values becomes effective again. This cycle can be repeated indefinitely. When the service time field is set to UNLIMITED, the set never expires.

For interactive mode job classes, any input from the terminal causes the first set of dispatching control values to take effect. Consequently, the specified sets of dispatching controls control the job's behavior on an input-by-input basis. For batch mode job classes, the dispatching control values are not reset. The specified sets of dispatching controls control the job's behavior on a job-wide basis.

Entering a pause break causes the first set of dispatching controls to take effect for the job. A RESUME_COMMAND command restores the dispatching control values that were in effect for the job at the time the pause break occurred.

Minor Time Slice

The minor time slice value specifies, as a multiplier of the CPU_QUANTUM_TIME scheduling attribute, the maximum amount of time, in microseconds, that a task can use the CPU before a call is made to the CPU dispatcher to select a different task of equal or higher priority.

In using the minor time slice, the system computes a value called the minimum slice. A minimum slice is defined as one eighth of the minor time slice. When a task's remaining minor time slice is less than the minimum slice, the system considers the task's minor time slice to have expired. In this way, the system avoids dispatching tasks that have essentially no remaining time slice.

It is important to distinguish between a service class's time slice and a task's remaining time slice. The service class minor time slice is the initial allotment of minor time slice CPU time assigned to a task. A task's remaining time slice is its current unused portion of that allotment. When a task uses up its minor time slice, it is given a new allotment of the minor time slice equal to the value of the service class minor time slice. The same is true of the major time slice.

The following describes how a task is asssigned a priority bias of high, medium, or low:

- Remaining minor time slice > minimum slice

  If the value of a task's remaining time slice is greater than its minimum slice, the task is considered to have a high priority bias. It is inserted into its priority list at the end of the tasks having a high priority bias.

- Remaining minor time slice < minimum slice < remaining major time slice

  If the value of a task's remaining minor time slice is less than its minimum slice, but its remaining major time slice is greater than the minimum slice, the task is considered to have a medium priority bias. Its remaining minor time slice is reset to its service class's minor time slice. The task is inserted at the end of the tasks having a medium priority bias.

- Remaining minor time slice and major time slice < minimum slice

  If the value of a task's remaining minor time slice and its remaining major time slice are both less than its minimum slice, the task is considered to have a low priority bias. Its remaining minor time slice and remaining major time slice values will be reset to the corresponding time slices for its service class. The task is inserted at the end of the tasks having a low priority bias.

Values for the minor time slice range from 1 to 100. The default value is 1.

Major Time Slice

The major time slice value specifies, as a multiplier of the CPU_QUANTUM_TIME scheduling attribute, the amount of time, in microseconds, a task can execute before losing its priority. At that time, it is placed at the end of the task's dispatching priority queue.

The value of the major time slice should be set such that the resulting value (major time slice * CPU quantum time) is 5,000 microseconds or more. If the resulting value is below 5,000, the increase in task switching may cause a decline in system throughput.

Values for the major time slice range from 1 to 100. The default value is 1.

NOTE

The major and minor time slice values have a significant effect on system performance. Although most tasks will page fault and give up the CPU before the minor time slice expires, increasing this value allows CPU-intensive tasks to keep the CPU for longer periods of time.

The five predefined service classes are set up to have nondynamic dispatching priorities. That is, the system does not dynamically adjust the job's dispatching priority according to the CPU service the job has already received. The dispatching control values for the five predefined service classes are as follows (the minor time slice and major time slice values are dependent on the value of the CPU_QUANTUM_TIME scheduling attribute):

SYSTEM	(P6,UNLIMITED,1,1)
MAINTENANCE	(P6,UNLIMITED,1,1)
INTERACTIVE	(P5,UNLIMITED,1,1)
BATCH	(P5,UNLIMITED,1,1)
UNASSIGNED	(P5,UNLIMITED,1,1)

The default set of values for the DISPATCHING_CONTROL attribute is as follows:

(P5,UNLIMITED,1,1)

The following example shows how you can establish dynamic dispatching controls for the INTERACTIVE service class for a local system:

```
/manage_active_scheduling
MAS/change_service_class class_name=interactive ..
MAS../dispatching_control=((p6,1000,1,1), ..
MAS../(p5,unlimited,1,1))
MAS/quit save_change=true
```

A job executing under the INTERACTIVE service class will execute at a dispatching priority of P6 for 1,000 milliseconds. If the command the job is executing has not completed within 1,000 milliseconds, the job's dispatching priority drops to P5. Each additional item of input from the terminal causes the job's dispatching priority to be reset to P6.

See chapter 5, CPU Scheduling, for further information about how the DISPATCHING_CONTROL attribute is used.

DUAL_STATE_PRIORITY_CONTROL or DSPC Scheduling Control Attribute

Defines a mapping of NOS/VE dispatching priorities to the dual-state priorities and subpriorities. This attribute applies to dual-state systems only.

The DUAL_STATE_PRIORITY_CONTROL attribute consists of a list of one to ten sets, in the following form:

```
   dual_state_priority_control=((set1),(set2),...,(set10))
```

The variable *set* represents the following order-dependent fields:

```
   set=(nos/ve priority,dual-state priority,subpriority)
```

Each NOS/VE priority (P1 to P10) can have a separate mapping. The NOS/VE priority field can also be specified as a range.

NOS/VE assigns priorities to its tasks and NOS or NOS/BE assigns priorities to its jobs. The task or job with the highest priority gets the CPU. If the priorities are equal, the subpriority value (1 to 15) determines which task or job gets the CPU.

A site can map NOS/VE dispatching priorities P1 to P10 to any of the dual-state priorities (1 to 7) to which NOS or NOS/BE map their priorities.

The following table shows the relationship between NOS or NOS/BE CPU priorities and the dual-state priorities.

NOS CPU Priority	NOS/BE CPU Priority	Dual-State Priority
10–17 octal		1
20–27 octal		2
30–37 octal	Batch/Interactive	3
40–47 octal		4
50–57 octal		5
60–67 octal		6
70–77 octal	System	7

The following table shows the default relationship between NOS/VE Dispatching priorities and the dual-state priorities and subpriorities.

NOS/VE Dispatching Priority	Dual-State Priority	Subpriority
P1–P2	1	8
P3–P4	2	8
P5–P6	3	8
P7–P8	4	8
P9–P10	5	8
P11–P12	6	8
P13–P14	7	8

NOS/VE divides the integer you specify for the subpriority by 16 to determine what proportion of the time a NOS/VE task gets the CPU first when a NOS or NOS/BE job has an equal priority. For example, a subpriority of 10 means that, when other priorities are equal, the NOS/VE task gets the CPU before the NOS or NOS/BE job in 10 cases out of 16.

Site-defined DUAL_STATE_PRIORITY_CONTROL attributes can only be used to modify the priority mappings of NOS/VE dispatching priorities P1 to P10. Any modifications involving NOS/VE dispatching priority P10 should be done with extreme caution. Misuse of this priority could cause a serious performance degradation, or in the worst case, a system hang.

ENABLE_APPLICATION_SCHEDULING or EAS Application Scheduling Attribute

Specifies a boolean value that enables or disables the application scheduling for the specified application. The default for this attribute is TRUE (application scheduling for the specified application is enabled).

ENABLE_CLASS_INITIATION or ECI Job Class Attribute

Specifies a boolean value that controls the initiation of jobs from the input queue. The default value is TRUE (class initiation is enabled).

For the UNASSIGNED job class, this attribute is changed to FALSE when a scheduling profile is activated. If you want the value to be set to TRUE for this class, you must change it explicitly by issuing the CHANGE_JOB_CLASS subcommand of the MANAGE_ACTIVE_SCHEDULING utility once the scheduling profile is activated.

ENABLE_JOB_LEVELING or EJL Scheduling Control Attribute

Specifies a boolean value that controls whether the mainframe participates in load leveling when the scheduling profile is activated. This attribute has meaning only for installations using multiple mainframes connected by the NOS/VE File Server and supporting load leveling. The default for this attribute is FALSE (the mainframe does not participate in load leveling).

EPILOG or E Job Class Attribute

Specifies the file path of a file containing SCL commands that are executed directly before the system epilog. When creating a file that is to contain a site epilog, make the file available for public access and specify ACCESS_MODE=(READ,EXECUTE) and SHARE_MODE=(READ,EXECUTE) on the CREATE_FILE_PERMIT command described in the NOS/VE Commands and Functions manual.

The default value for all job classes is NONE (no epilog is used).

EXCLUDED_CATEGORIES or EC Job Class Attribute

Defines a list of job categories that are excluded from the specified job class. All jobs assigned any of the excluded categories are prevented from running in this job class. Any job category that is in the REQUIRED_CATEGORIES list cannot be in the EXCLUDED_CATEGORIES list. You can also specify the keywords NONE or ALL. The value ALL specifies that all defined categories are excluded and that only jobs to which no categories are assigned can become members of this class. The following are the values for this attribute as defined for the five predefined job classes:

SYSTEM	INTERACTIVE
MAINTENANCE	NONE
INTERACTIVE	BATCH
BATCH	INTERACTIVE
UNASSIGNED	NONE

The default value for this attribute is NONE.

GUARANTEED_SERVICE_QUANTUM or GSQ Service Class Attribute

Specifies the integer number of service units a job must accumulate after it is swapped in before it can be considered a candidate for preemption (that is, being swapped out to allow another job to be swapped in). A job can be preempted before receiving this amount of service under the following conditions:

- The system is low on memory (thrashing is occurring). Jobs are swapped out to free up the target number of pages (specified by the target memory value of the SCHEDULING_MEMORY_LEVELS attribute).
- The limit specified by the services class's MAXIMUM_ACTIVE_JOBS scheduling attribute has been reached and another job with a higher dispatching priority (within the same service class) is waiting to be swapped in. This situation occurs if dynamic dispatching controls are defined for a service class.
- Memory is needed to swap in a job with a higher dispatching priority (regardless of service class).
- The job has a lower dispatching priority than other CPU-bound jobs and the time period specified by the IDLE_DISPATCHING_QUEUE_TIME scheduling attribute has elapsed.

The value specified by this attribute ensures that a job is given a defined amount of service with jobs of equal dispatching priority before it can be preempted by other jobs of equal dispatching priority. Values for this attribute range from 0 to 100,000,000,000. You can also specify the keyword UNLIMITED.

The following are the values for this attribute as defined for the five predefined service classes:

SYSTEM	2,000
MAINTENANCE	9,400
INTERACTIVE	9,000
BATCH	20,000
UNASSIGNED	20,000

The default value for this attribute is 100.

NOTE

Service is guaranteed only when jobs of equal dispatching priority are being considered. A job with a higher dispatching priority preempts a job with a lower dispatching priority even if the latter has not reached its guaranteed amount of service.

IDLE_DISPATCHING_QUEUE_TIME or IDQT Scheduling Control Attribute

Specifies the number of seconds the system waits before classifying a dispatching priority queue as idle (nondispatchable). All jobs in an idle dispatching priority queue are swapped out. These jobs are not considered for swapin until the dispatcher begins service on their dispatching queue. The value specified by this attribute is checked at intervals defined by the CHECK_IDLE_DISPATCHING_INTERVAL system attribute. The value specified by this attribute ranges from 10 to 36,000 seconds. You can also specify the keyword UMLIMITED. The system default is 360.

IMMEDIATE_INITIATION_CANDIDATE or IIC Job Class Attribute

Specifies a boolean value that controls the number and type of members allowed into the job class specified. A value of TRUE disallows membership into the class if the number of initiated jobs plus the number of queued jobs for the class exceed the value of the INITIATION_LEVEL scheduling attribute. In this case, a LOGIN, SUBMIT_JOB or JOB command returns an error status. A value of FALSE allows such membership.

The following are values for this attribute as defined for the five predefined job classes:

SYSTEM	FALSE
MAINTENANCE	TRUE
INTERACTIVE	TRUE
BATCH	FALSE
UNASSIGNED	FALSE

The default value for this attribute is FALSE.

INITIAL_SERVICE_CLASS or ISC Job Class Attribute

Specifies the service class name under which jobs begin processing. User validation is not required for the initial service class. This service class must have been defined with the CREATE_CLASS subcommand of the ADMINISTER_SERVICE_CLASS subutility before you can specify it in a job class definition. The default value for this attribute is the service class of the same name as the job class (if existent). If the service class does not exist, the default is the service class UNASSIGNED.

INITIAL_WORKING_SET or IWS Job Class Attribute

Specifies the amount of memory, in pages, a job requires upon its initiation. The value specified can be in the range of 20 to 65,000 pages. Set this attribute to a value that approximates the size of the working set of a job when the job begins execution. After initiation, the size of the job's actual working set is determined through class controls, the memory manager, and individual job processing characteristics.

The following are the values for this attribute as defined for the five predefined job classes:

SYSTEM	80
MAINTENANCE	65
INTERACTIVE	65
BATCH	200
UNASSIGNED	200

The default value for this attribute is 65.

INITIATED_JOBS or IJ Job Class Attribute

Shows the number of jobs currently in the execution phase under the job class specified when the class is displayed under the MANAGE_ACTIVE_SCHEDULING utility.

INITIATION_AGE_INTERVAL or IAI Job Class Attribute

Specifies the time interval, in seconds, used for aging jobs in the input queue. Values for this attribute range from 1 to 36,000 seconds. You can also specify the keyword UNLIMITED. The UNLIMITED value means that the job's initiation priority remains static; no aging takes place.

The following are the values for this attribute as defined for the five predefined job classes:

SYSTEM	1
MAINTENANCE	60
INTERACTIVE	1
BATCH	60
UNASSIGNED	60

The default value for this attribute is 1.

INITIATION_EXCLUDED_CATEGORIES or IEC Scheduling Control Attribute

Defines a list of job categories that are excluded from initiating on a specified mainframe. No job assigned a category in this list will be initiated on the mainframe. Initiation refers to the point at which the system takes a job from the input queue and begins executing the job. Typically, all job categories specified for the VALIDATION_EXCLUDED_CATEGORIES control attribute should also be specified for this control attribute to prevent jobs with those job categories from initiating. Likewise, those job categories that are specified for this control attribute should be specified for the VALIDATION_EXCLUDED_CATEGORIES control attribute to prevent jobs that never initiate from being submitted.

Any job category that is in the INITIATION_REQUIRED_CATEGORIES list cannot be in the INITIATION_EXCLUDED_CATEGORIES list. You can also specify the keywords NONE or ALL. The keyword ALL specifies that all defined job categories are excluded and that only jobs to which no categories are assigned can be initiated on the specified mainframe. The default is NONE.

INITIATION_LEVEL or IL Job Class Attribute

Specifies the number of jobs that can be initiated from a specified job class before job class initiation is closed. Jobs must then terminate from this class before additional job class members can be initiated from the input queue.

If a specified job class is defined as an immediate initiation candidate (IMMEDIATE_INITIATION_CANDIDATE=TRUE), it cannot accept additional members as long as job class initiation is closed.

Values for this attribute range from 0 to 10,000. You can also specify the keyword UNLIMITED.

The following are the values for this attribute as defined for the five predefined job classes:

SYSTEM	UNLIMITED
MAINTENANCE	10
INTERACTIVE	UNLIMITED
BATCH	10
UNASSIGNED	10

The default value for this attribute is 20.

INITIATION_REQUIRED_CATEGORIES or IRC Scheduling Control Attribute

Defines a list of job categories that a job must be assigned before it can be initiated on a specified mainframe. Initiation refers to the point at which the system takes a job from the input queue and begins executing the job. Typically, all job categories specified for the VALIDATION_REQUIRED_CATEGORIES control attribute should also be specified for this control attribute to prevent jobs without those job categories from initiating. Likewise, those job categories that are specified for this control attribute should be specified for the VALIDATION_REQUIRED_CATEGORIES control attribute to prevent jobs that never initiate from being submitted.

Any job category that is in the INITIATION_EXCLUDED_CATEGORIES list cannot be in the INITIATION_REQUIRED_CATEGORIES list. You can also specify the keywords NONE or ALL. The keyword ALL specifies that all defined job categories are required and that only jobs to which all job categories are assigned can be initiated on the specified mainframe. The default is NONE.

JOB_LEVELING_INTERVAL or LSI Scheduling Control Attribute

Defines the interval, in seconds, that the job leveler task uses between communications with the server mainframes to obtain or return jobs. This attribute has meaning only for installations using multiple mainframes connected by the NOS/VE File Server and supporting load leveling. Values that can be specified for this attribute range from 1 to 3,600 seconds. The default value is 60.

JOB_LEVELING_PRIORITY_BIAS or JLPB Job Class Attribute

Defines an integer value that is applied as a bias to the selection priority of all jobs in this job class on this mainframe. This attribute is used to bias jobs that are resident (cataloged on a local family device) on a server mainframe such that these jobs tend to be selected for initiation on the server mainframe rather than on the client mainframe. Generally, it is better to execute these jobs on the server mainframe, when possible, since family access from jobs load leveled to other mainframes requires access through the server to perform I/O on the family devices. This attribute has meaning only for installations using multiple mainframes connected by the NOS/VE File Server and supporting load leveling. Values that can be specified for this attribute range from -8,000,000 to 8,000,000. The default value is 0.

JOB_LEVELING_PRIORITY_BIAS or JLPB Scheduling Control Attribute

Defines an integer value that is applied as a bias to the selection priority of all jobs on this mainframe. This attribute is used to bias jobs that are resident (cataloged on a local family device) on a server mainframe such that these jobs tend to be selected for initiation on the server mainframe rather than on the client mainframe. Generally, it is better to execute these jobs on the server mainframe, when possible, since family access from jobs load leveled to other mainframes requires access through the server to perform I/O on the family devices. This attribute has meaning only for installations using multiple mainframes connected by the NOS/VE File Server and supporting load leveling. Values that can be specified for this attribute range from -8,000,000 to 8,000,000. The default value is 0.

MAGNETIC_TAPE_LIMIT or MTL Job Class Attribute

Specifies the default maximum number of magnetic tape drives that can be assigned to a job concurrently. If a job has unspecified magnetic tape requirements, it is assigned the job attribute default as defined by the CHANGE_JOB_ATTRIBUTE_DEFAULTS command. If this value is unspecified, it is assigned the magnetic tape limit of the specified job class. If you specify zero, the job is considered a nontape job and is not allowed to use magnetic tapes. Values for this attribute range from 0 to 100 tape drives. You can also specify the keyword UNLIMITED. The default value is UNLIMITED.

NOTE

This value is a default only and does not restrict a job with a MAGNETIC_TAPE_LIMIT value that is greater than this default value from being a member of the job class. In order to restrict membership to a job class, use job categories.

MAXIMUM_ACTIVE_JOBS or MAXAJ Service Class Attribute

Defines how many jobs can be active at one time in a specified service class. Excess jobs processing under a specified service class will be in a swapped state. The value specified by this attribute can be in the range of 0 to 250 jobs. You can also specify the keyword UNLIMITED.

The following are the values for this attribute as defined for the five predefined service classes:

SYSTEM	20
MAINTENANCE	100
INTERACTIVE	100
BATCH	50
UNASSIGNED	0

The default value for this attribute is 20.

MAXIMUM_WORKING_SET or MAXWS Job Class Attribute

Specifies the maximum number of pages allowed in a job's working set. The value specified can be in the range of 20 to 65,000 pages.

This value is specified as a list of three integers, in the form:

```
(default,minimum,maximum)
```

For example:

```
MAXIMUM_WORKING_SET=(1000,200,2000)
```

In this example, 1,000 is the default, 200 is the smallest maximum working set, and 2,000 is the largest maximum working set.

The default value is assigned if the job does not specify a maximum working set on a LOGIN, JOB, or SUBMIT_JOB command and is explicitly assigned to a job class. If a job is automatically assigned to a job class (by specifying JOB_CLASS=AUTOMATIC on a LOGIN, JOB, or SUBMIT_JOB command), the maximum working set is defined through the job attribute defaults (see the description of the DISPLAY_JOB_ATTRIBUTE_DEFAULTS command in the NOS/VE Operations manual). Within the range specified by the MAXIMUM_WORKING_SET attribute, the value can later be changed by the CHANGE_JOB_ATTRIBUTE command.

Note that the system enforces a system-wide maximum working set that may be less than that allowed by the MAXIMUM_WORKING_SET attribute. The system limit on a job's working set is equal to the total number of pages on the mainframe minus the total of all pages in the wired queue, the shared queue, the system job's working set, and the target memory value (first integer) of the SCHEDULING_MEMORY_LEVELS attribute.

To determine reasonable values for the MAXIMUM_WORKING_SET attribute, examine the JM3, PM3, and PM6 statistics after running a representative workload. These statistics are described in chapter 9, Statistics Facility. The default maximum working set should be large enough to minimize the number of tasks slowed down due to excessive paging, thus generating PM6 statistics. If only a few applications cause excessive paging, consider improving the application according to guidelines described in the NOS/VE Object Code Management manual.

The default set of values for this attribute is as follows:

(1000,20,1000)

NOTE

This value is a default only and does not restrict a job with a MAXIMUM_WORKING_SET value that is greater than this default value from being a member of the job class. In order to restrict membership to a job class, use job categories.

If a job's supplied MAXIMUM_WORKING_SET value is greater than the default value, the job's MAXIMUM_WORKING_SET value is lowered to the default value when the job is initiated.

MAXIMUM_WORKING_SET or MAXWS Application Scheduling Attribute

Specifies the maximum number of pages allowed in a job's working set. The value specified can be in the range of 20 to 65,000 pages.

The default value for this attribute is UNSPECIFIED. This means that the maximum working set for the job does not change while the application is executing.

MINIMUM_WORKING_SET or MINWS Job Class Attribute

Specifies the minimum number of pages to which a job's working set may be reduced by the page aging process. The value specified can be in the range of 20 to 65,000.

NOTE

The minimum number of pages specified by this attribute only applies to the page aging process. Other ways in which the number of pages in a job's working set might drop below the value specified by the MINIMUM_WORKING_SET attribute are: deleting files, detaching files, closing files, reading/writing sequential files, terminating tasks, and having too many pages in a segment.

This value is specified as a list of three integers, in the form:

```
(default,minimum,maximum)
```

For example:

```
MINIMUM_WORKING_SET=(50,20,5000)
```

In this example, 50 is the default, 20 is the smallest minimum working set size, and 5,000 is the largest minimum working set size.

Within the range specified by the MINIMUM_WORKING_SET attribute, you can later change the default using the CHANGE_JOB_ATTRIBUTE command (described in the NOS/VE Commands and Functions manual).

The default set of values for this attribute is as follows:

(20,20,1000)

MINIMUM_WORKING_SET or MINWS Application Scheduling Attribute

Specifies the minimum number of pages allowed in a job's working set. The value specified can be in the range of 20 to 65,000 pages.

If you specify this attribute while defining application scheduling under the ADMINISTER_APPLICATION subutility, the value specified remains in effect as long as the application is executing. When the application completes, the minimum working set reverts to its normal value.

The default value for this attribute is UNSPECIFIED. This means that the minimum working set for the job does not change while the application is executing.

NEXT_SERVICE_CLASS or NSC Service Class Attribute

Specifies the service class to which a job is switched when the value specified by the CLASS_SERVICE_THRESHOLD attribute is exceeded. Any defined service class can be specified, as well as the keywords UNSPECIFIED or NONE. The keywords UNSPECIFIED and NONE mean that no service class switch occurs. User validation is not required for the specified service class.

Using NEXT_SERVICE_CLASS, you can create several service classes and link them together.

You cannot delete a service class that the NEXT_SERVICE_CLASS attribute references by another name unless you change this reference.

The default value for this attribute is NONE.

PAGE_AGING_INTERVAL or PAI Job Class Attribute

Specifies a default CPU time interval, in microseconds, used for aging the job's working set. A page fault occurring after this time interval causes the job's working set to be aged. The value specified can be in the range of 1,000 to 3,600,000,000.

This value is specified as three integers, in the form:

```
(default,minimum,maximum)
```

For example:

```
PAGE_AGING_INTERVAL=(100000,50000,2000000)
```

In this example, 100,000 is the default, 50,000 is the minimum interval, and 2,000,000 is the maximum interval. The default value must be within the minimum and maximum range. If you specify only the default, the minimum and maximum equal the default and therefore cannot be changed by the job.

Within the range specified by the PAGE_AGING_INTERVAL attribute, the value can later be changed by the CHANGE_JOB_ATTRIBUTE command (described in the NOS/VE Commands and Functions manual).

The PAGE_AGING_INTERVAL attribute works in conjunction with the JOB_WORKING_SET_AGE_INTERVAL memory attribute (described in chapter 3, Managing Memory). Like the PAGE_AGING_INTERVAL attribute, JOB_WORKING_SET_AGE_INTERVAL sets an interval of CPU time at which the job's working set is aged, but the interval it sets is longer. If the interval set by JOB_WORKING_SET_AGE_INTERVAL elapses before a page fault occurs, the job's working set is aged at the end of that interval. However, if a page fault occurs before the interval set by the JOB_WORKING_SET_AGE_INTERVAL memory attribute elapses and after the shorter interval set by the PAGE_AGING_INTERVAL attribute elapses, the working set is aged.

The following example assumes that page faults occur before the interval set by JOB_WORKING_SET_AGE_INTERVAL elapses. As a result, the shorter PAGE_AGING_INTERVAL determines when the job's working set is aged. Assume that the PAGE_AGING_INTERVAL is 50 milliseconds (50,000 microseconds). After the job has used 50 milliseconds of CPU time, the next page fault causes the pages in the working set to be aged. Figure 4-1 represents a time line showing when the page faults occurred and the first time the working set was aged (at 59 milliseconds). PAI stands for PAGE_AGING_INTERVAL and pf stands for page fault.

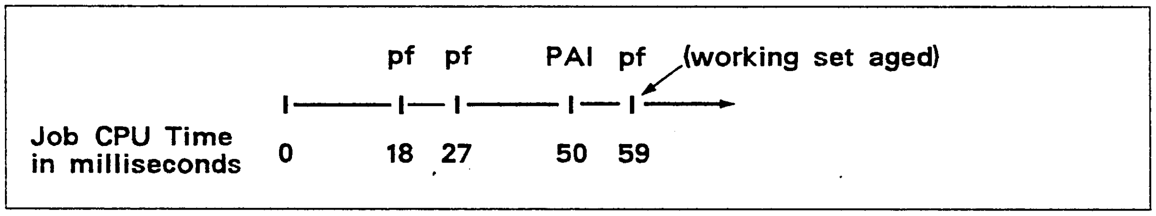


Figure 4-1. CPU Time Line of Page Faults and the PAGE_AGING_INTERVAL

Once the working set is aged, the system calculates the time of the next check. It does this by adding the interval set by PAGE_AGING_INTERVAL to the time of the last page fault. The working set is aged at the first page fault to occur after that time. Thus, as shown in figure 4-1, the working set was aged at 59 milliseconds. The working set pages are next aged at the first page fault that occurs after 109 milliseconds of CPU execution time (59+50=109). In figure 4-2, the page fault occurred at 137 milliseconds.

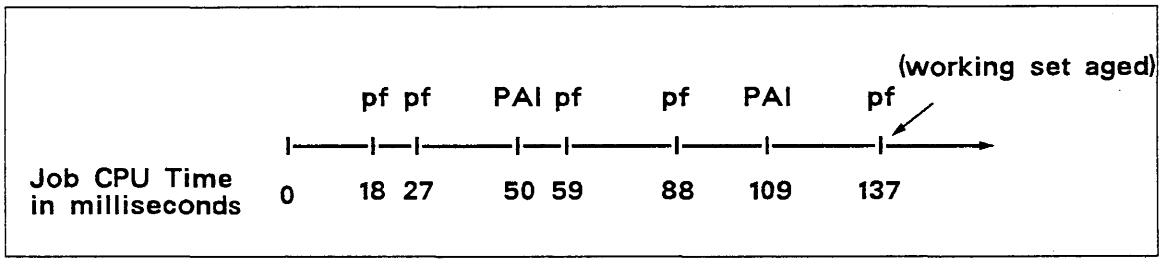


Figure 4-2. CPU Time Line with Working Set Examined Twice

When the working set is aged, each page is examined to see whether it has been used (accessed). The age of a page is then adjusted as follows:

- If a page has been used since the working set was last aged, its age is set to zero.
- If a page has not been used since the working set was last aged, the age of the page is incremented by the value of the CPU time the job has used (since the working set was last aged) divided by the value of the PAGE_AGING_INTERVAL scheduling attribute. In addition, some pages may be removed from the working set during aging. For an explanation of which pages are removed from the working set, refer to the AGE_INTERVAL_CEILING and AGE_INTERVAL_FLOOR memory attributes of the MANAGE_MEMORY utility, described in chapter 3, Managing Memory. For an explanation of how CPU time is computed, see the description of the AGING_ALGORITHM memory attribute in chapter 3.

The PAGE_AGING_INTERVAL attribute can have a significant effect on system performance. If it is set too high, the job working sets can increase to a point at which swapping becomes ineffective.

The default values for the PAGE_AGING_INTERVAL attribute are model-dependent as follows:

Mainframe Model	PAI Default
810, 815	300,000
825, 830, 930-11, 930A, 930C, 932-11	200,000
835, 840, 930-31, 930B, 932-31, 932A, 932B	100,000
840S, 845, 845S, 850, 855, 855S, 860, 865, 960-11, 960-31, 990, 992-31, 992-32, 994-31, 994-32, 995	50,000

The default minimum value for the PAGE_AGING_INTERVAL attribute is 10,000. The default maximum value is 1,000,000,000.

PAGE_AGING_INTERVAL or PAI Application Scheduling Attribute

Specifies a default CPU time interval, in microseconds, used for aging a job's working set. A page fault occurring after this time interval causes the job's working set to be aged. The value specified can be in the range of 1,000 to 3,600,000,000.

If you specify this attribute while defining application scheduling under the ADMINISTER_APPLICATION subutility, the value specified remains in effect as long as the application is executing. When the application completes, the page aging interval reverts to its normal value.

The default value is UNSPECIFIED. This means that the page aging interval for the job does not change while the application is executing.

PROLOG or P Job Class Attribute

Specifies the file path of a file containing SCL commands that are executed directly after the system prolog. When creating a file that is to contain a site prolog, make the file available for public access and specify ACCESS_MODE=(READ,EXECUTE) and SHARE_MODE=(READ,EXECUTE) on the CREATE_FILE_PERMIT command described in the NOS/VE Commands and Functions manual.

The default value for all job classes is NONE (no prolog is used).

QUEUED_JOBS or QJ Service Class Attribute

Shows the number of jobs currently awaiting initiation under the specified job class or service class when the class is displayed under the MANAGE_ACTIVE_SCHEDULING utility.

REQUIRED_CATEGORIES or RC Job Class Attribute

Defines a list of job categories that a job must have before it can be assigned to a specified job class. A job must be assigned all of the listed job categories to belong to the specific job class. Any job category that is in the EXCLUDED_CATEGORIES list cannot be in the REQUIRED_CATEGORIES list. You can also specify the keywords NONE or ALL. The keyword ALL means that all defined categories are required. The default value is NONE.

SCHEDULING_MEMORY_LEVELS or SML Scheduling Control Attribute

Specifies a list of two integer values used for memory allocation. This list appears in the following form:

```
(target memory,thrashing level)
```

For example:

```
SCHEDULING_MEMORY_LEVELS=(100,60)
```

In this example, 100 is the target memory value, and 60 is the thrashing level. The default target memory and thrashing level values are as follows:

(60,20)

The following paragraphs describe these fields in more detail.

Target Memory

The target memory value specifies a level of free and available pages that the job scheduler attempts to keep in real memory. Jobs are not initiated into memory if doing so will cause the number of free and available pages to fall below this level. The value specified can be from 0 to 100,000 pages. The default is 60.

The job scheduler decides whether to swap in or initiate a job based upon this target level. When deciding whether to swap in or initiate a job, the scheduler first looks at the swapin and input queues and selects the job with the highest priority. The scheduler then swaps in or initiates that job, if doing so will not drive the number of free and available pages below the target level. If there are not enough free and available pages to swap in or initiate the job, the scheduler checks the currently active jobs to determine if there is an active job that satisfies one of the following conditions:

- Has exceeded its guaranteed service and has a lower scheduling priority than the job waiting to be swapped in or initiated. (You can set the level of guaranteed service with the GUARANTEED_SERVICE_QUANTUM scheduling attribute.)
- Has a lower dispatching priority than the job waiting to be swapped in or initiated.

If such an active job exists, it is swapped out and the waiting job becomes active.

NOTE

If necessary, the system adjusts the target memory value set by this attribute to a higher value. (The system never sets a lower target memory value.) The system adjusts the target memory value to be large enough to allocate a swap file descriptor for the largest working set currently allowed (a swap file descriptor requires approximately 72 bytes per page). The largest working set currently allowed is the smaller of the following values:

- The largest maximum working set (that is, the upper value of the range set by the MAXIMUM_WORKING_SET scheduling attribute of all classes with initiated jobs.)
- The system-wide maximum working set being enforced by the system (see the MAXIMUM_WORKING_SET scheduling attribute description for more information).

Thrashing Level

The thrashing level value specifies the minimum number of free and available pages to keep in real memory to prevent system thrashing. The value specified can be in the range of 0 to 100,000 pages. The default is 20.

If, during job execution, the number of free and available pages falls below this minimum value, only system jobs are assigned free and available pages. Jobs other than system jobs are swapped out to prevent system thrashing. Thrashing is a condition in which too much paging occurs in a system, and the resulting overhead causes system performance to degrade. One way to prevent thrashing is to make sure that each job's working set fits in the available memory without disturbing the other jobs' working sets. This value, together with the target memory value, gives you some control over thrashing.

The thrashing level and the value specified by the AGGRESSIVE_AGING_LEVEL_2 memory attribute (described in chapter 3, Managing Memory) should be equivalent or within a few pages of each other.

SCHEDULING_PRIORITY or SP Service Class Attribute

Specifies a list of values that indicate the priority range and increments that control a job's eligibility for memory.

This list appears in the following form:

```
   (minimum priority,maximum priority,swap age increment,ready task increment)
```

The minimum priority value sets a floor value for the priority of jobs in the service class. The maximum priority value sets a ceiling value for the priority of jobs in the service class. The swap age increment value defines, for the swap queue, the increment for each interval (specified by the SWAP_AGE_INTERVAL scheduling attribute) that a job is in the swapin queue. The ready task increment value defines the swapin queue priority boost given to a job that is swapped out for long wait and that has a ready task.

For example:

```
   SCHEDULING_PRIORITY=(200,2000,1,100)
```

In this example, 200 is the minimum priority set for the job, 2,000 is the maximum priority, 1 is the job's swap age increment, and 100 is the job's ready task increment. Values you can specify for this attribute range from 0 to 16,000,000.

All jobs are initiated into memory at the maximum priority and are reduced, at intervals determined by the SERVICE_CALCULATION_INTERVAL scheduling attribute, by the amount of service already received. If more than one job reaches the minimum priority level and a swapout is required, the oldest of these jobs becomes the most likely candidate for swapout.

Swapped jobs are placed in the swapin candidate queue with the minimum priority or minimum priority plus ready task increment if the job is swapped out for long wait. The priority is incremented by the swap age increment for each interval (specified by the SWAP_AGE_INTERVAL scheduling attribute) that the job remains in the swapin candidate queue.

The job's scheduling priority in the swapin queue is computed as follows:

```
scheduling priority=(minimum priority + ready task increment)
+ (swap age * swap age increment / swap age interval)
```

If the job was placed in the swapin queue because it had an idle task go ready, the ready task increment value is also added in when the scheduling priority is computed. If the computed scheduling priority exceeds the service class's maximum priority, the scheduling priority is set to the maximum priority. If more than one job reaches its maximum priority, the jobs that have aged the longest have priority over younger jobs.

Job Preemption Based on Relative Priorities

The term *preemption* refers to the case in which a job is swapped out of memory to make way for another job of higher priority. The job scheduler makes job preemption decisions based on the following rules:

- Jobs with a higher dispatching priority are always able to preempt jobs with a lower dispatching priority. This applies even if the job with a lower dispatching priority has not reached guaranteed service. Service is guaranteed only when jobs with equal dispatching priorities are considered.
- Jobs with a lower dispatching priority can never preempt jobs with a higher dispatching priority. This is true even if the job with the higher dispatching priority has reached guaranteed service.
- For jobs with equal dispatching priorities, the job scheduler uses their scheduling priorities to determine whether preemption can occur. A job in the input queue or swapin queue can preempt an active job with an equal dispatching priority and an equal or lower scheduling priority only if the active job has exceeded the value specified by the GUARANTEED_SERVICE_QUANTUM scheduling attribute.

The dispatcher keeps the job scheduler informed of the dispatching priority queues from which it can dispatch tasks to the CPU. If the dispatcher has not sent any tasks with a lower priority to the CPU for a specified amount of time, it swaps out all jobs with the lower dispatching priority. The jobs with a lower dispatching priority are not considered for activation again until the dispatcher indicates that it is able to dispatch the lower priority tasks.

The amount of time the scheduler must wait before classifying a dispatching control queue as idle (nondispatchable) can be varied by the IDLE_DISPATCHING_QUEUE_TIME scheduling attribute.

Jobs that are preempted before reaching guaranteed service are inserted into the swapin candidate queue in front of jobs of the same service class and dispatching priority that had reached guaranteed service when they were swapped. When such a job is swapped back into memory, the amount of service guaranteed for it is the amount it had remaining at the time it was preempted.

Once a job is swapped into memory, its scheduling priority is assigned the maximum priority. The job's scheduling priority is decremented by the amount of service the job used each interval specified by the SERVICE_CALCULATION_INTERVAL attribute. Refer to the description of the SERVICE_FACTORS scheduling attribute to see how service is calculated.

The following are the values for the minimum priority, maximum priority, swap age increment, and ready task increment as defined for the five predefined service classes:

	Minimum	Maximum	SAI	RTI
SYSTEM	19,000	22,000	1,000	0
MAINTENANCE	8,000	17,000	1,000	1,000
INTERACTIVE	8,000	17,000	1,000	1,000
BATCH	1,000	10,000	1,000	0
UNASSIGNED	1,000	10,000	1,000	0

The default set of values for the SCHEDULING_PRIORITY attribute is as follows:

```
(1000,10000,1000,0)
```

SELECTION_PRIORITY or SP Job Class Attribute

Specifies a list of values that define the initiation priority of queued input jobs in a job class.

This list appears in the following form:

```
(initial priority,maximum priority,increment)
```

For example:

```
SELECTION_PRIORITY=(400,2000,1)
```

In this example, 400 is the initial priority of a job in the input queue, 2,000 is the maximum priority of a job in the input queue, and 1 is the priority increment for aging a job in the input queue. Values you can specify for this attribute range from 0 to 16,000,000.

A job submitted to the system is assigned the initial priority for its job class. A job's initiation priority is computed as follows:

```
initiation priority=(initial priority) + (queue age * increment
/ initiation age interval)
```

If the computed initiation priority exceeds the job class's maximum priority, the initial priority is set to the maximum priority.

The queue age value is the number of seconds the job has been in the input queue. The initiation age interval value is the aging interval, in seconds, defined by the INITIATION_AGE_INTERVAL attribute for the job class.

The class priority of a job is the smaller of its initiation priority or the maximum priority of its job class. Unless initiation of jobs is restricted for a job class by other controls (such as INITIATION_LEVEL), the next job to be initiated is the one with the highest class priority across all of the job classes. If more than one job is at the maximum priority for its job class, the job with the highest initiation priority is selected. This is the job that has been in the input queue the longest for that job class.

The following are the values for the initial priority, maximum priority, and priority increment values as defined for the five predefined job classes:

	Initial	Maximum	Increment
SYSTEM	19,000	22,000	100
MAINTENANCE	14,000	17,000	1,000
INTERACTIVE	14,000	17,000	1,000
BATCH	5,000	10,000	10
UNASSIGNED	5,000	10,000	10

The default set of values for the SELECTION_PRIORITY attribute is as follows:

(5000,10000,10)

SELECTION_RANK or SR
Job Class Attribute

Changes the order of job classes appearing in the scheduling profile. The job class specified by this attribute indicates the position in the job class list where a higher-ranking job class is to be placed. The job class you designate as higher ranking is specified by the CLASS_NAME parameter of the CHANGE_ATTRIBUTE subcommand of the ADMINISTER_JOB_CLASS subutility. This job class will appear immediately before the job class specified by the SELECTION_RANK attribute.

Changing the relative order of job classes in the profile affects how automatic job class selection is performed. When a job is automatically assigned to a job class, the classes are considered for such assignment in the order in which they appear in the scheduling profile. Only job classes with the AUTOMATIC_CLASS_SELECTION attribute set to TRUE are considered for automatic job class selection.

For example, assume that a site has defined its job classes in the following order:

FAST_INTERACTIVE
EXPRESS_INTERACTIVE
FAST_BATCH
EXPRESS_BATCH

On this basis, a job that may hold membership in either the FAST_INTERACTIVE or EXPRESS_INTERACTIVE job class is assigned to the FAST_INTERACTIVE class if automatic class selection is specified (AUTOMATIC_CLASS_SELECTION=TRUE). If you want to designate the EXPRESS_INTERACTIVE job class as higher-ranking than the FAST_INTERACTIVE job class, you need to issue the following subcommand under the ADMINISTER_JOB_CLASS subutility:

```
change_attribute class_name=express_interactive selection_rank=fast_interactive
```

Since this subcommand places the EXPRESS_INTERACTIVE job class ahead of the FAST_INTERACTIVE job class, jobs that can hold membership in both classes are now assigned to the EXPRESS_INTERACTIVE job class.

To determine a job class's selection rank, use the DISPLAY_PROFILE_SUMMARY subcommand or any of the job class display subcommands with the CLASS_NAME parameter set to ALL. The order in which the job classes are displayed is the class rank order.

SERVICE_CALCULATION_INTERVAL or SCI Scheduling Control Attribute

Specifies the delay, in seconds, between successive calls to calculate service for all active jobs. This job service rate controls an active job's priority adjustments and resource allocation during its execution life-cycle. The value specified can be in the range of 1 to 3,600 seconds. The system default value is 10 seconds.

SERVICE_CLASS or SC Application Scheduling Attribute

Specifies the service class name under which the job processes while a specified application is active. The default service class is UNSPECIFIED (service class does not change).

If you specify this attribute while defining application scheduling under the ADMINISTER_APPLICATION subutility, the value specified remains in effect as long as the application is executing. When the application completes, the service class reverts to its normal value.

SERVICE_FACTORS or SF Service Class Attribute

Specifies the integer multipliers applied when a job's service is computed. The integers appear in the following form:

```
(CPU multiplier,memory multiplier,residence multiplier,I/O multiplier)
```

For example:

```
SERVICE_FACTORS=(1,1,1,1)
```

In this example, 1 is the multiplier applied to all four service factors. Values you can specify for this attribute range from 0 to 100. Setting any of the service factors to zero removes that resource value from the computed service limits.

At intervals determined by the SERVICE_CALCULATION_INTERVAL scheduling attribute, the job scheduler calculates the amount of service each job in memory has used since the last time service was calculated. Service is also calculated each time a job is swapped out for long wait. Service is determined by the following formula:

```
service = (CPU milliseconds used * CPU multiplier)
          + (job working set size * memory multiplier)
          + (seconds swapped in * residence multiplier)
          + (number of page faults * I/O multiplier)
```

Each time the service calculation is performed for a job, the amount of service each job has used is added to its total accumulation of service and to its accumulation of service since being swapped in. Each time a job swap occurs, the count of service accumulated since swapin is reset to zero. The active job's scheduling priority is decremented by the amount of service used.

The service factors affect the actual service given to jobs within a service class. Specifically, in calculating the actual service given to a job, the memory multiplier and CPU multiplier values determine whether the job scheduler gives greater weight to memory space used or to CPU time used. The I/O multiplier and residence multiplier values weight the service cost of page faults generated by a job and the service cost of the amount of real time a job resides in memory.

The default set of SERVICE_FACTORS values for all service classes is as follows:

(1,1,1,1)

SRU_LIMIT or SL
Job Class Attribute

Specifies the default maximum number of system resource units (SRUs) a job can accumulate. If a job is explicitly assigned to a job class and the LOGIN command for that job did not specify an SRU limit, the job is assigned the user's validation limit or the value of the specified job class, whichever is smaller. If the job is automatically assigned to a job class (with a JOB_CLASS=AUTOMATIC parameter on a LOGIN, JOB, or SUBMIT_JOB command), the job is assigned the user's validation limit or the job attribute default, whichever is smaller. In both cases, this limit becomes the job's job abort limit. A job is unconditionally terminated when its job abort limit is reached. If the user-specified SRUs are greater than the current SRU limit, the job is terminated immediately.

Values for this attribute range from 1 to $MAX_INTEGER. You can also specify the keywords UNLIMITED or SYSTEM_DEFAULT. The default value is UNLIMITED. If you specify SYSTEM_DEFAULT, the default SRU limit defined by the CHANGE_JOB_ATTRIBUTE_DEFAULTS command is used.

Basic information about SRUs is found in the NOS/VE Accounting Analysis manual. Information about how a site can define its own SRU formula is found in the Site Tailoring chapter of the NOS/VE System Performance and Maintenance manual, Volume 2.

NOTE

This value is a default only and does not restrict a job with an SRU_LIMIT value that is greater than this default value from being a member of the job class. In order to restrict membership to a job class, use job categories.

SWAP_AGE_INTERVAL or SAI
Service Class Attribute

Specifies the real-time increment, in seconds, by which the computed scheduling priority is to be advanced by the swap age increment value set by the SCHEDULING_PRIORITY attribute. Values for this attribute range from 1 to 36,000. You can also specify the keyword UNLIMITED. The default value for all service classes is 1.

SWAPPED_JOBS or SJ Service Class Attribute

Shows the number of jobs currently swapped out under a specified service class when the class is displayed under the MANAGE_ACTIVE_SCHEDULING utility.

VALIDATION_EXCLUDED_CATEGORIES or VEC Scheduling Control Attribute

Defines a list of job categories that are invalid for the specified mainframe. A login attempt by a job assigned any of the excluded job categories is rejected. A job is queued only if it is valid on at least one mainframe. In a nonleveling environment, the only mainframe checked is the mainframe on which the job is submitted. In a load leveling environment, the mainframes that are checked are those mainframes to which the login family is served for job leveling. Typically, all job categories specified for this control attribute should also be specified for the INITIATION_EXCLUDED_CATEGORIES control attribute to prevent jobs with these job categories from initiating. Likewise, those job categories that are specified for the INITIATION_EXCLUDED_CATEGORIES control attribute should be specified for this control attribute to prevent jobs that never initiate from being submitted.

Any job category that is in the VALIDATION_REQUIRED_CATEGORIES list cannot be in the VALIDATION_EXCLUDED_CATEGORIES list. You can also specify the keywords NONE or ALL. The keyword ALL specifies that all defined job categories are excluded and only jobs that are assigned none of the job categories are valid on the specified mainframe. The default is NONE.

VALIDATION_REQUIRED_CATEGORIES or VRC Scheduling Control Attribute

Defines a list of job categories that a job must be assigned to be valid on the specified mainframe. A job is queued only if it is valid on at least one mainframe. In a nonleveling environment, the only mainframe checked is the mainframe on which the job is submitted. In a load leveling environment, the mainframes that are checked are those mainframes to which the login family is served for job leveling. Typically, all job categories specified for this control attribute should also be specified for the INITIATION_REQUIRED_CATEGORIES control attribute to prevent jobs without these job categories from initiating. Likewise, those job categories that are specified for the INITIATION_REQUIRED_CATEGORIES control attribute should be specified for this control attribute to prevent jobs that never initiate from being submitted.

Any job category that is in the VALIDATION_EXCLUDED_CATEGORIES list cannot be in the VALIDATION_REQUIRED_CATEGORIES list. You can also specify the keywords NONE or ALL. The keyword ALL specifies that all defined categories are required and that only jobs to which all job categories are assigned are valid on the specified mainframe. The default is NONE.

CPU Scheduling 5

CPU Scheduling 5

This chapter describes the CPU scheduling that is performed by the NOS/VE task dispatcher. Specifically, it describes the task dispatching algorithm and the parameters that you can alter to tune the task dispatcher. Note that CPU scheduling is done on a task-by-task basis. In contrast, memory scheduling is handled by the job scheduler.

The following scheduling attributes are referrenced in this chapter and are described in more detail in chapter 4, Job Scheduling:

- CPU_DISPATCHING_ALLOCATION
- CPU_DISPATCHING_INTERVAL
- DISPATCHING_CONTROL

Assigning Task Dispatching Priorities

Every job consists of one or more tasks. The job scheduler swaps a job with a ready task into memory based on the job's scheduling priority. The task dispatcher, on the other hand, schedules tasks for CPU execution based on the task's dispatching priority. To have its tasks dispatched for CPU execution, a job must be swapped into memory.

You can choose one of two methods of assigning task dispatching priorities: nondynamic assignment or dynamic assignment. Under nondynamic assignment of task dispatching priorities, a job's dispatching priority remains at a constant level. Under dynamic assignment of task dispatching priorities, you can establish dispatching controls that enable the system to periodically adjust a job's dispatching priority according to the service the job has already received.

Nondynamic Assignment of Task Dispatching Priorities

NOS/VE has eight user dispatching priorities, numbered P1 to P8. Subsystem and system dispatching priorities are numbered P9 to P14.

Tasks inherit most of the attributes pertaining to task dispatching from the service class of the job to which they belong. Described in the following table, these attributes include the default dispatching priority, the minor time slice, and the major time slice. Tasks maintain the values of these attributes in the execution control block (XCB).

Attribute	Description
Dispatching priority	Designated by the values P1 to P10; P10 is the highest priority. The dispatching priority is defined by the DISPATCHING_CONTROL scheduling attribute.
Minor time slice	Value that specifies the maximum amount of time, in microseconds, that a task can use the CPU before a call is made to the CPU dispatcher to select a different task of equal or higher priority. The value is specified as a multiplier of the CPU_QUANTUM_TIME scheduling attribute. The minor time slice is defined by the DISPATCHING_CONTROL scheduling attribute.
Major time slice	Value that specifies the amount of time, in microseconds, that a task can execute before losing its priority. The value is specified as a multiplier of the CPU_QUANTUM_TIME scheduling attribute. When the task loses its priority, the task is placed at the end of its dispatching priority queue. The major time slice is defined by the DISPATCHING_CONTROL scheduling attribute.

Dynamic Assignment of Task Dispatching Priorities

The DISPATCHING_CONTROL scheduling attribute assigns dynamic dispatching priorities for a service class. This attribute consists of a list of up to five sets. Each set contains the following order-dependent fields:

- Dispatching priority
- Service limit
- Minor time slice (multiplier of CPU_QUANTUM_TIME)
- Major time slice (multiplier of CPU_QUANTUM_TIME)

The following example of the DISPATCHING_CONTROL attribute specifies three sets of dispatching controls:

```
DISPATCHING_CONTROL=((P6,3000,1,10)..
                     (P5,7000,1,10)..
                     (P4,UNLIMITED,1,10))
```

When a job is initiated, all tasks for the job are assigned the dispatching priority and time slice values specified in the first set. Once the job has reached the service limit defined for the set, all tasks for the job are assigned the next set of dispatching values. That set of values remains in effect until the service limit defined by that set is reached. In this manner, the job continues to cycle through each of the defined sets.

You can also specify the keyword UNLIMITED for the service limit. This means that the dispatching controls included in that set are in effect for the life of the job. UNLIMITED can be specified only for the last set.

The list of sets may be circular; when the service limit has expired for the last dispatching control set defined, the values in the first dispatching control set are again in effect.

For interactive mode job classes, the first set of dispatching controls takes effect every time the terminal user enters a carriage return (indicating that data is ready for command processing). Thus, the specified sets of dispatching controls control the job's behavior on a command-by-command basis. For batch mode job classes, the dispatching control values are not reset. The specified sets of dispatching controls control the job's behavior on a job-wide basis.

Using dynamic dispatching priorities for interactive mode jobs allows a site to assign a high dispatching priority to a job as long as the job is processing terminal I/O requests. When an interactive mode job becomes CPU-bound, the lower dispatching controls defined for the class take effect for the job's tasks.

For a more detailed set of examples, see Examples Using Dynamic Dispatching later in this chapter.

Example Using Nondynamic Dispatching

The minor time slice and major time slice values regulate CPU competition between tasks from different job classes. Because a task's dispatching priority has such a dominant effect, the following example reflects only cases where the dispatching priorities for the two classes are the same. In addition, since I/O-bound tasks do not use much CPU time, the example considers two competing CPU-bound tasks from different classes as follows:

- Case 1: Tasks have CPU times that total less than the major time slice. These are called short tasks. In this case, the ratio of the minor time slices of competing job classes governs the relative CPU time that is received by tasks in those classes. If the ratio of the minor time slice in job class 1 to the minor time slice in job class 2 is 3 to 1, CPU-bound tasks from job class 1 receive approximately three times more CPU time than short CPU-bound tasks from job class 2.

- Case 2: Tasks have CPU times that total much more than the major time slice. In this case, the relative CPU time received by competing tasks from different classes is governed by the ratio of major time slices.

The example in this section assumes the following job classes and dispatching time slice values:

Job Class	Minor Time Slice	Major Time Slice
CLASS_1	50,000	200,000
CLASS_2	50,000	100,000

The minimum slice (equal to one-eighth of the minor time slice) for both job classes is 6,250 microseconds (50,000/8). In the short term, CPU-bound tasks should receive about the same service. In the long term, CPU-bound tasks from class 1 should accumulate CPU time twice as fast as tasks from job class 2. Normalizing to the basic time unit of 50,000 microseconds, we have:

Job Class	Minor Time Slice	Major Time Slice
CLASS_1	1	4
CLASS_2	1	2

Assume the following tasks are competing for the CPU:

Task 1:	Belongs to CLASS_1 and comes ready first.
Task 2:	Belongs to CLASS_2.

Table 5-1 tracks the dispatching of these two tasks for several time units.

Table 5-1. Task Dispatching

	Task 1				Task 2				
Time Int	Rem Minor Slice	Rem Major Slice	Bias	Acc CPU	Rem Minor Slice	Rem Major Slice	Bias	Acc CPU	Task to Execute
1 B	1	4	H	0	1	2	H	0	1
1 E	0	3	H	1	1	2	H	0	
2 B	1	3	M	1	1	2	H	0	2
2 E	1	3	M	1	0	1	H	1	
3 B	1	3	M	1	1	1	M	1	1
3 E	0	2	M	2	1	1	M	1	
4 B	1	2	M	2	1	1	M	1	2
4 E	1	2	M	2	0	0	M	2	
5 B	1	2	M	2	1	2	L	2	1
5 E	0	1	M	3	1	2	L	2	
6 B	1	1	M	3	1	2	L	2	1
6 E	0	0	M	4	1	2	L	2	
7 B	1	4	L	4	1	2	L	2	2
7 E	1	4	L	4	0	1	L	3	
8 B	1	4	L	4	1	1	M	3	2
8 E	1	4	L	4	0	0	M	4	
9 B	1	4	L	4	1	2	L	4	1
9 E	0	3	L	5	1	2	L	4	
10 B	1	3	M	5	1	2	L	4	1
10 E	0	2	M	6	1	2	L	4	
11 B	1	2	M	6	1	2	L	4	1
11 E	0	1	M	7	1	2	L	4	
12 B	1	1	M	7	1	2	L	4	1
12 E	0	0	M	8	1	2	L	4	
13 B	1	4	L	8	1	2	L	4	2
13 E	1	4	L	8	0	1	L	5	
14 B	1	4	L	8	1	1	M	5	2
14 E	1	4	L	8	0	0	M	6	
15 B	1	4	L	8	1	2	L	6	1
15 E	0	3	L	9	1	2	L	6	
16 B	1	3	M	9	1	2	L	6	1
16 E	0	2	M	10	1	2	L	6	
17 B	1	2	M	10	1	2	L	6	1
17 E	0	1	M	11	1	2	L	6	
18 B	1	1	M	11	1	2	L	6	1
18 E	0	0	M	12	1	2	L	6	
19 B	1	4	L	12	1	2	L	6	2

Table 5-1 reflects the following activity:

1. Initially, both tasks have high priority. This is because the remaining minor time slice and remaining major time slice for both tasks are greater than their minimum slices.

2. Task 1 is dispatched, since it appeared first in the list.

3. At the end of interval 1:

   a. Task 1 has used up its minor time slice. Accordingly, an interrupt is generated and it loses the CPU.

   b. Task 1 is still ready.

   c. Its remaining minor time slice is less than its minimum slice.

   d. Its remaining major time slice is greater than its minimum slice.

   e. Accordingly, task 1 is placed in the dispatching list with medium priority bias and is given a new minor time slice.

   f. The value of its major time slice is decremented by the amount of CPU time it used. (This is shown at the beginning of time interval 2.)

   g. Task 1 is now placed in the list after task 2 and has medium priority bias.

4. At the end of interval 2:

   a. Task 2 has used up its minor time slice.

   b. Accordingly, it is placed in the dispatching list with medium priority bias and is given a new minor time slice.

5. At the beginning of interval 3, task 1 is selected for execution (since task 2 was put in the list with medium priority bias after task 1).

6. In all subsequent intervals, task 1 and task 2 alternate between low and medium priority biases. The pattern for these intervals is as follows:

   a. Task 1 remains at medium priority bias for three intervals, while task 2 is at low priority bias.

   b. Subsequently, task 1 is given low priority bias, while task 2 is dispatched for two minor time slice intervals.

   c. When task 2's major time slice expires, it again receives low priority bias.

   d. The ratio of 4 to 2 continues until one of the tasks terminates. Thus, the ratio of CPU time for the tasks follows the ratio of their major time slices.

NOTE

Although this example demonstrates how two particular tasks compete for the CPU, this type of tuning does not guarantee that two job classes will necessarily accumulate CPU time along these ratios.

Examples Using Dynamic Dispatching

These examples use the highest-priority-first dispatching algorithm. Consider the following DISPATCHING_CONTROL entries:

For job class INTERACTIVE:

```
DISPATCHING_CONTROL=((P6,3000,1,1)..
                     (P5, 7000,1,1)..
                     (P4, UNLIMITED,1,1))
```

For job class QUICK_BATCH:

```
DISPATCHING_CONTROL=((P6,10000,1,1)..
                    (P3,UNLIMITED,1,1))
```

For job class HIGH_BATCH:

```
DISPATCHING_CONTROL=((P5,15000,1,1)..
                     (P4,15000,1,1))
```

For job class LOW_BATCH:

```
DISPATCHING_CONTROL=((P4,UNLIMITED,1,1))
```

Under the dispatching controls specified above and using a highest-priority-first task selection algorithm, jobs experience the following CPU dispatching activity:

INTERACTIVE jobs:

- INTERACTIVE jobs start with a dispatching priority of P6.
- Dispatching priority remains at P6 as long as each interaction requires no more than 3,000 service units to complete.
- Dispatching priority drops to P5 if an interaction requires more than 3,000 service units to complete.
- Dispatching priority drops to P4 if an interaction requires more than 7,000 additional service units. This priority remains in effect until the command completes. The task is assigned to the CPU only when no more P5 or P6 tasks are ready to execute.
- Each time a command is read from the terminal, the service accumulator is reset to zero and the dispatching priority is reset to P6.

QUICK_BATCH jobs:

- QUICK_BATCH jobs start with a dispatching priority of P6 and thus are able to compete for the CPU with INTERACTIVE mode jobs.
- If the job is still running after accumulating 10,000 service units, the dispatching priority drops to P3 and remains there until the job terminates.

HIGH_BATCH and LOW_BATCH jobs:

- Dispatching priority alternates between P5 and P4 for HIGH_BATCH jobs, thus allowing a site to run both HIGH_BATCH and LOW_BATCH jobs.
- If HIGH_BATCH jobs run only at P5, all LOW_BATCH jobs are excluded.

The number and types of jobs in the system at any given time affect the selection of tasks for dispatching. As long as the interactive load of jobs does not totally claim the CPU, and as long as QUICK_BATCH jobs are submitted only occasionally, HIGH_BATCH and LOW_BATCH jobs can be dispatched to the CPU with little difficulty. However, if new QUICK_BATCH jobs are continually submitted, tasks of HIGH_BATCH and LOW_BATCH jobs are excluded from the CPU.

Task Selection

Only ready tasks are considered for task dispatching. Idle, waiting, or blocked tasks are not considered. Within each dispatching priority, P1 to P14, ready tasks are kept in a linked, ordered list. A ready task is linked into one of the dispatching control table (DCT) queues based on the task's dispatching priority. Pointers are kept to allow tasks to be placed in the list at three different positions: high-priority bias, medium-priority bias, and low-priority bias.

The system uses a default task selection algorithm of highest-priority-first. A task is selected for CPU execution from the head of the highest priority DCT queue. A task in a lower priority DCT queue is selected to execute only when all higher priority DCT queues are empty. When a task becomes CPU-bound, all lower tasks with a lower priority are excluded from the CPU.

Allocating the CPU to User Dispatching Priorities

Rather than using a highest-priority-first task selection algorithm, you can use the CPU_DISPATCHING_ALLOCATION scheduling attribute to allocate the CPU among the user dispatching priorities (P1 to P8). You can define the minimum and maximum percent of an interval of CPU execution time to devote to executing tasks of a given dispatching priority. This ensures that low-priority tasks get some execution time even if higher-priority tasks are CPU-bound.

The CPU_DISPATCHING_INTERVAL scheduling attribute defines the number of seconds of CPU execution time to which the minimum and maximum percentages apply. The CPU_DISPATCHING_ALLOCATION scheduling attribute consists of a list of one to eight sets. Each set consists of the following order-dependent fields: dispatching priority or range of priorities, minimum percent, maximum percent, and enforce maximum percent. See the description of the CPU_DISPATCHING_ALLOCATION control attribute in chapter 4, Job Scheduling, for a description of these fields.

Rules That Apply to Dispatching Allocation

The following rules apply to dispatching allocation:

- Time spent executing system or subsystem priority tasks or NOS tasks is not counted as part of the CPU execution interval for user dispatching priorities.
- Tasks with system or subsystem priorities are always selected to execute first, using a highest-priority-first algorithm.

- The task dispatcher starts at the highest priority DCT queue and selects tasks from that DCT queue until the minimum percentage for that priority is satisfied. When the minimum percentage is satisfied or no ready tasks are left in that DCT queue, tasks from the next lower DCT queue are considered for selection.

- Higher priority tasks are selected over lower priority tasks as long as the higher priority DCT queue has not reached its minimum percentage.

- When the minimum percentages for all DCT queues have been satisfied, higher priority tasks are again selected over lower priority tasks until the higher priority DCT queue reaches its maximum percentage.

- Tasks are selected from a DCT queue that has reached its maximum percentage only if all DCT queues that have not reached their maximum percentage are empty and if the enforce maximum percent field of the CPU_DISPATCHING_ALLOCATION control attribute is FALSE.

- When a task is selected from a DCT queue, it executes for its entire minor time slice, even if the minimum or maximum percentage of CPU execution time for the DCT queue expires first. For example, if a task with a dispatching priority of P4 has 20,000 microseconds of execution time left until its minimum percentage is satisfied and P4 tasks have a minor time slice of 30,000 microseconds, the P4 task is selected and executes for its entire 30,000 microsecond time slice.

Example of Task Selection

The following table lists the default values for the CPU_DISPATCHING_ALLOCATION scheduling attribute. Because the minimum percent is automatically satisfied and the maximum percent cannot be exceeded, the task from the head of the highest-priority DCT queue is always selected to execute.

Dispatching Priority	Minimum Percent	Maximum Percent	Enforce Maximum Percent
P8	0	100	FALSE
P7	0	100	FALSE
P6	0	100	FALSE
P5	0	100	FALSE
P4	0	100	FALSE
P3	0	100	FALSE
P2	0	100	FALSE
P1	0	100	FALSE

For this example, assume that you issue the following CHANGE_CONTROLS subcommand of the MANAGE_ACTIVE_SCHEDULING utility to change some of the values of the CPU_DISPATCHING_ALLOCATION scheduling attribute.

```
/manage_active_scheduling
MAS/change_controls cpu_dispatching_allocation=((P7,10,10,true),..
MAS../(P6,25,100,false),(P5,15,100,false),(P4,10,25,false))
MAS/quit
```

The following table lists the new values for the dispatching controls.

Dispatching Priority	Minimum Percent	Maximum Percent	Enforce Maximum Percent
P8	0	100	FALSE
P7	10	10	TRUE
P6	25	100	FALSE
P5	15	100	FALSE
P4	10	25	FALSE
P3	0	100	FALSE
P2	0	100	FALSE
P1	0	100	FALSE

Dispatching priority P8 does not have a minimum percent of the CPU execution time defined. In this case, P8 tasks are not selected to execute until dipatching priorities P7 to P4 have achieved their minimum percentages or until no ready tasks remain in any of the DCT queues for dispatching priorities P7 to P4.

Dispatching priority P7 tasks are selected first. When they have executed for 10 percent of the CPU time interval, or until no more ready tasks are in the P7 DCT queue, dispatching priority P6 tasks are selected. Because a maximum of 10 percent is defined and enforce maximum percent is TRUE for dispatching priority P7, P7 tasks execute for 10 percent of the CPU time interval and are no longer selection candidates for the rest of the interval.

One reason to define a relatively high dispatching priority with a low maximum percent of the CPU time interval is to control I/O-bound tasks, yet respond to them quickly. Job and service classes can be set up so that jobs with tasks that are known to be I/O-bound have a dispatching priority of P7. I/O-bound tasks are typically in page wait most of the time. They go ready, execute for a short time, issue an I/O request, and wait for the I/O to complete. A high dispatching priority allows an I/O-bound task to execute as soon as the I/O request completes. Setting a maximum percent of the CPU time interval that P7 tasks can use prevents them from monopolizing the CPU if there are many P7 tasks or if one of the P7 tasks becomes CPU-bound.

Dispatching priority P6 could be used for short interactive commands. P6 tasks execute for 25 percent of the CPU time interval or until no more P6 tasks are ready. Defining lower dispatching priorities with a percent of the CPU time interval devoted to them ensures that lower priority tasks get some execution time, even if many P6 tasks are ready.

Dispatching priorities P5 and P4 could be used for long interactive commands or batch jobs. Tasks are selected from the P5 and then P4 DCT queues until each DCT queue achieves the minimum percentage defined. When P4 tasks have executed for 25 percent of the CPU time interval, they are no longer selected unless no ready tasks exist in any other DCT queue except P7.

Any time a task goes ready in a higher priority DCT queue that has not reached the minimum percentage defined, the higher priority task is selected.

When all minimum percentages have been satisfied, tasks are selected using a highest-dispatching-priority-first algorithm. When the maximum percentage has been reached, tasks in that DCT queue are not selected unless no ready tasks exist in any other DCT queue.

Dispatching priorities P3, P2, and P1 could be used for background batch jobs. No minimum percentage is defined for these priorities. Tasks with these priorities are selected to execute only if no other tasks are ready, or if only P7 tasks or P4 tasks are ready and they have already used 10 percent and 25 percent of the CPU time interval, respectively.

Tuning the Task Dispatcher

The purpose of tuning the task dispatcher is to control competition for the CPU between tasks of the same service class or of different service classes.

You can use the DISPATCHING_CONTROL and CPU_DISPATCHING_ALLOCATION scheduling attributes to tune the task selection algorithm. Remember that the task dispatcher starts with the highest priority DCT queue and satisfies all minimum percentages allocated. After all minimum percentages have been satisfied, the highest-priority-first algorithm again takes effect.

Keep in mind that when a minor time slice or major time slice expires, a temporary penalty is imposed on a task: the task is placed at the end of the appropriate priority bias list (DCT queue). As tasks ahead of it in the list use up their slices, its relative position in the list improves. When the task is allowed to execute (and subsequently repeat the cycle of losing the CPU, coming ready, and needing placement in the dispatching list), its placement is based on the current values of its slices, rather than on its previous position in the dispatching list.

It is important to know how the minor time slice and major time slice attributes affect competition for the CPU among jobs of the same class. The following paragraphs pertain to I/O-bound jobs and CPU-bound jobs and how they compete for the CPU.

- The significant dispatching characteristic of an I/O-bound job is that it uses only a small amount of CPU time before it becomes blocked for a page fault from disk. As a result, it may take many page faults before an I/O-bound task uses up one minor time slice.

- In general, an I/O-bound task does not interfere with CPU time requests by other tasks. This is because the task is not eligible to run during the period of time it is waiting for the page fault to be satisfied. Frequently, I/O-bound tasks have high-priority bias in the dispatching list.

- One type of I/O-bound job is handled differently by the system. This is an I/O-bound job that is running at its maximum working set. This class of job can easily cause severe I/O bottlenecks and degrade system performance for all users. Typically, this job needs to run at a larger maximum working set.

  To prevent an I/O-bound job that is running at its maximum working set from adversely affecting other users, NOS/VE slows that job down. For example, assume a job has 10 outstanding I/O requests and the system must remove a page from the job's working set because the job is at its maximum working set limit. In this case, the system forces that job to issue a 20-millisecond WAIT request. Because the execution of the job is slowed down, it does not retard system performance by flooding the system with too many page faults.

- Due to its dispatching characteristics, a CPU-bound task holds the CPU for its entire minor time slice before giving it up. A CPU-bound job does not receive a high-priority bias in the dispatching list. Instead, it frequently has a medium-priority bias. Occasionally, it has a low-priority bias.

- Because of their relative priority biases, an I/O-bound job will most likely have access to the CPU as soon as the currently executing task loses it. I/O-bound tasks do not accumulate behind CPU-bound tasks. CPU-bound tasks are not at a disadvantage due to this scheme, since they usually do not have to wait for even a full minor time slice before an I/O-bound task loses the CPU.

- The purpose of the minimum slice value (equal to one-eighth of the minor time slice) is to prevent the task dispatcher from knowingly putting a task in execution that will be forced to lose the CPU in a very short time. Thus, the overhead involved in switching such tasks is avoided.

Task Execution

Using a highest-priority-first algorithm, the task dispatcher always selects for execution the first executable task with the highest priority. Thus, if you set the interactive class dispatching priority to P6 and the batch class dispatching priority to P5, a heavy interactive workload may completely exclude batch tasks from executing.

In a dual-CPU system, a job is limited, by default, to having only one of its tasks execute at a time. Thus, if a ready task is ready to be selected for execution but another task from the same job is already executing in the other CPU, that task is not selected. Instead, the task dispatcher selects the next executable task. The task that was skipped remains at the head of the dispatching queue. However, you can choose to have tasks from the same job executing in both CPUs simultaneously by entering the SET_MULTIPROCESSING_OPTIONS command with the MULTIPROCESS parameter set to ON. In this case, a second ready task can be selected for execution. For more information about this commnad, see the NOS/VE Commands and Functions manual.

When a task is dispatched, the current value of its minor time slice is written to the sytem interval timer (SIT). The SIT is a clock that decrements at a microsecond rate. The SIT decrements all the time regardless of whether the CPU is executing in job mode or monitor mode. For performance reasons, the SIT continues to decrement even when the CPU is switched to execute NOS or NOS/BE.

Once a task is selected for execution, it executes until an interrupting event occurs. When an executing task loses the CPU, the amount of CPU time used by the task is subtracted from both the task's remaining minor time slice and major time slice (described under the DISPATCHING_CONTROL scheduling attribute in chapter 3, Job Scheduling). If the resulting value for either time slice is negative, its value is set to zero.

The following events can interrupt a task's execution:

- When SIT decrements to zero. This event generates an interrupt that causes the task dispatcher to be called to select the highest priority task for execution. When the SIT decrements to zero, the task's minor time slice has expired.
- A task with a higher priority goes ready. This must be a task with a higher dispatching priority rather than merely a task that, with the next pass of the task dispatcher, precedes the current executing task in the same priority dispatching list.
- The task becomes blocked on some event. Typically, this event constitutes waiting for satisfaction of a page fault from disk. Creating new pages or satisfaction of page faults from the available queues does not cause a task to lose the CPU.
- The task executes a WAIT, DELAY, or CYCLE request.

The algorithm is different if the executing task has system or subsystem tables interlocked. In this case, the task is allowed to continue execution until one of the following conditions appears:

- The task becomes blocked.
- The task executes a WAIT, DELAY, or CYCLE request.

If the task gives up the CPU for either of these reasons, the task's dispatching priority is raised to P10 until it gives up the system table interlocks. A flag is set so that the task releases the CPU when system tables are no longer interlocked.

Page Aging 6

Page Aging 6

Page aging is a method of controlling the size of working sets. When a working set is aged, unused pages may be removed. Removal is performed through writing to disk, or through deleting unmodified pages. Pages that are accessed frequently remain in memory, while those pages that are accessed less often are removed.

Only pages in the shared working set page queues and the job working set page queue are aged. Each shared working set page queue is aged according to its own set of attributes. Job working set aging is based in part on attributes that are defined by the job's class.

Four different page aging algorithms are described in this chapter:

Page Aging Algorithm	Description
Shared working set cyclic aging	Aging based on use and wall clock time; pertains to working sets that are shared among jobs. The count of the number of times a page in a shared working set is aged in this way is called the shared working set cyclic age.
Job working set cyclic aging	Aging based on use and wall clock time; pertains to working sets unique to a job. The count of the number of times a page in a job working set is aged in this way is called the job working set cyclic age.
Job working set swapout count aging	Aging based on the number of times the job's pages have been swapped out of memory; pertains to working sets unique to a job. The count of the number of times a page in a job working set is aged in this way is called the job working set swapout count age.
Job working set CPU aging	Aging based on a job's accumulated CPU time; pertains to working sets unique to a job. The count of the number of times a page in a job working set is aged in this way is called the job working set CPU age.

The memory and shared queue attributes listed below apply to page aging and are described in more detail in chapter 3, Managing Memory. Pages of a shared file are placed in one of several shared working set queues based on the file type. Each shared queue has its own queue parameters that specify periodic aging, minimum queue size, and maximum queue size.

PERIODIC_CALL_INTERVAL
SHARED_WORKING_SET_AGE_INTERVAL
AGE_INTERVAL_CEILING (shared queues)
MINIMUM_SIZE (shared queues)

The following memory attributes affect page aging of the job working set. They are described in more detail under the description of the MANAGE_MEMORY utility in chapter 3, Managing Memory.

AGE_INTERVAL_CEILING
AGE_INTERVAL_FLOOR
AGING_ALGORITHM
JOB_WORKING_SET_AGE_INTERVAL
PERIODIC_CALL_INTERVAL
SWAPPING_AIC

The following scheduling attributes apply to page aging and are described in more detail in chapter 4, Job Scheduling:

CYCLIC_AGING_INTERVAL
MINIMUM_WORKING_SET
INITIAL_WORKING_SET
MAXIMUM_WORKING_SET
PAGE_AGING_INTERVAL

Cyclic Aging of Shared Working Sets

Cyclic aging of shared working set page queues (also called shared queues) makes use of the following attributes under the control of the MANAGE_MEMORY utility. The MANAGE_MEMORY utility is decribed in chapter 3, Managing Memory.

PERIODIC_CALL_INTERVAL

Memory attribute that specifies the amount of time between calls to the NOS/VE memory manager to perform memory cleanup and aging functions. These functions are determined by the SHARED_WORKING_SET_AGE_INTERVAL memory attribute and the JOB_WORKING_SET_AGE_INTERVAL memory attribute. The default value for this memory attribute is 1,000,000 microseconds. Allowable values range from 500,000 to 10,000,000.

SHARED_WORKING_SET_AGE_INTERVAL

Memory attribute that specifies the rate, in microseconds, at which the system ages shared queues. The default value for this memory attribute is 8,000,000 microseconds. Allowable values range from 1,000,000 to 999,999,999.

AGGRESSIVE_AGING_LEVEL

Memory attribute that specifies a value at which the system forces the aging of the shared queues and the job working sets.

AGE_INTERVAL_CEILING

Shared queue attribute that defines the time interval, in microseconds, during which a page must be used before the page is aged out of the shared queue.

MINIMUM_SIZE

Shared queue attribute that defines the minimum number of pages that a shared queue must contain.

Cyclic aging of shared queues is based on their frequency of use and the expiration of wall clock time intervals. Before this type of aging takes place, the memory management periodic function manager is called at intervals established by the PERIODIC_CALL_INTERVAL memory attribute. Within the periodic function, shared queue aging proceeds when one of the following conditions occurs:

- An interval greater than that established by the SHARED_WORKING_SET_AGE_INTERVAL memory attribute has elapsed since the last time the shared queue was aged. Because the expiration of the shared working set age interval is tested within a procedure that is only called when the interval specified by the PERIODIC_CALL_INTERVAL memory attribute expires, the value of the PERIODIC_CALL_INTERVAL memory attribute should be less than the value of the SHARED_WORKING_SET_AGING_INTERVAL memory attribute.

- The number of reassignable pages in the system is less than the value specified by the AGGRESSIVE_AGING_LEVEL memory attribute.

Shared queue aging attempts to examine every page in a shared queue's working set. However, if at any time the shared queue's size is less than or equal to the value specified by the MINIMUM_SIZE shared queue attribute, the current pass of shared queue aging is stopped. The one exception to this process is as follows: if the aging of a shared queue is called because of aggressive aging, the system does not check the limit set by the MINIMUM_SIZE shared queue attribute.

A controlling factor in the aging of shared queues is the used bit. A used bit is associated with each page in memory. This bit is set to indicate that the page has been used since the last time the shared queue was aged. When the system encounters a set used bit during a page aging pass, the following activity takes place:

1. The used bit is cleared.
2. The cyclic age of the page is reset to 0.
3. The CPU age of the page is reset to 0.

If a page's used bit is clear, the page has not been used since the last time the shared working set was aged. In this case, the following possibilities exist:

- The cyclic age could be less than the value specified by the shared queue's AGE_INTERVAL_CEILING attribute. If so, the cyclic age of the page is incremented by 1 (set equal to 1).
- The cyclic age could be greater than or equal to the value specified by the shared queue's AGE_INTERVAL_CEILING attribute. If so, the page is removed from the shared queue. Pages removed that have been modified are placed in the available modified queue. Pages removed that have not been modified are placed in the available queue.

In summary, the process defined by the shared queue aging algorithm attempts to remove any page that has not been used for consecutive aging intervals that equal the value of the AGE_INTERVAL_CEILING shared queue attribute plus one. However, the shared queue aging algorithm does not attempt to reduce the shared queue to a size smaller than the value specified by the shared queue's MINIMUM_SIZE attribute.

Cyclic Aging of Job Working Sets

Job working set aging is similar to shared working set aging except that it pertains to pages that are used exclusively by a particular job. Job working set aging takes place when one of the following conditions occurs:

- An interval greater than that established by the JOB_WORKING_SET_AGE_INTERVAL memory attribute has elapsed since the last time a job's working set was cyclically aged.
- The number of reassignable pages in the system is less than or equal to the value set by the AGGRESSIVE_AGING_LEVEL memory attribute. (Reassignable pages refers to the sum of the pages in the free queue and the available queue.)

The aging algorithm considers every job in the active job list (AJL) in the following ways:

- If the current job's cyclic aging interval has expired, the job's working set is cyclically aged.
- If the job's cyclic aging interval has not expired, the job's working set becomes a candidate for CPU aging (described later in this chapter). CPU aging is initiated at this time only when one of the following conditions exists:
  - The number of reassignable pages in the system is less than the value set by the AGGRESSIVE_AGING_LEVEL memory attribute.
  - The job has accumulated CPU time greater than three page aging intervals without being CPU-aged. The definition of CPU time depends on which aging algorithm is in effect (as specified by the AGING_ALGORITHM memory attribute).

Job working set cyclic aging attempts to examine every page in the job's working set. However, if at any time the job's working set size is less than or equal to the job's minimum working set, the current pass of this job's working set cyclic aging is stopped.

Cyclic aging causes the system to set the job's next CPU aging time by adding the job's page aging interval to its current accumulated CPU time.

A page's used bit also plays a part in the aging of job working sets. If a page's used bit is set, the page has been used since the last time the job's working set was aged. When a set used bit is encountered during a subsequent page aging pass, the following activity takes place:

1. The used bit is cleared.
2. The cyclic age of the page is reset to 0.
3. The CPU age of the page is reset to 0.

If a page's used bit is clear, the page has not been used since the last time the job's working set was aged. In this case, the following possibilities exist:

- The cyclic age could be less than 1 (equal to 0). If so, the cyclic age of the page is incremented by 1 (set equal to 1).
- The cyclic age could be greater than or equal to 1. If so, the page is removed from the job's working set. Pages that are aged out of the job's working set that have been modified are put into the available modified queue. Pages that are aged out of the job's working set that have not been modified are put into the available queue.

NOTE

The cyclic age used in job working set cyclic aging is the same variable as the cyclic age used in job working set swapout count aging. Thus, it is possible for the two types of aging to affect each other. However, if you use swapout count aging and job working set cyclic aging as they are intended, the interference should be minimal. The released job class default cyclic aging interval is 1,000 seconds. This setting effectively disables job working set cyclic aging. The intent is to use swapout count aging and CPU aging to age job working sets. Job working set cyclic aging is left as an alternative if the other two types of aging do not meet the site's needs. In this case, we recommend you consider setting the PAGE_AGING_INTERVAL scheduling attribute high so that CPU aging is effectively disabled. In addition, we recommend that you set the SWAPPING_AIC memory attribute high to effectively disable swapout count aging. As a result, the interference between the two algorithms is minimized.

Because the expiration of the job working set age interval is tested within a procedure that is only called when the interval established by the PERIODIC_CALL_INTERVAL memory attribute expires, the value of the PERIODIC_CALL_INTERVAL memory attribute should be less than the job working set age interval. Similarly, since the expiration of the cyclic aging interval is tested within a loop that is only entered when the job working set age interval expires, the value of the job working set age interval should be less than the cyclic aging interval.

In summary, the job working set aging algorithm attempts to remove any page that has not been used for two consecutive cyclic aging intervals. However, the process defined by the job working set aging algorithm does not attempt to reduce the job working set to a size smaller than the value of the minimum job working set.

Swapout Count Aging of Job Working Sets

Swapout count aging removes pages according to how often the job's pages have been swapped out of memory. Specifically, the page's age is a count of the number of times a page has been swapped out without being used for the duration of time the job was swapped in. This type of aging is initiated by the job swapper, which must first be called by the job scheduler.

The job swapper initiates swapout count page aging when one of the following conditions occurs:

- The job is in long wait. It makes no difference whether the job is swapped to disk or stays memory-resident while swapped out. The most common example of a job in long wait is an interactive job in which the user has received the prompt at the terminal and has not yet responded.

- The cyclic aging interval of the job being swapped has expired. Checking the cyclic aging interval period at swapout ensures that jobs do not escape aging just because their swapout causes them to miss the next job working set age interval. However, the aging performed at swapout uses the value in the SWAPPING_AIC memory attribute as the maximum cyclic age of an unused page rather than an age of two cyclic aging intervals.

Swapout count aging attempts to examine every page in the job's working set. However, if at any time the job's working set size is less than or equal to the job's minimum working set, the current pass of the job's swapout count aging is stopped.

Swapout count aging causes the system to set the job's next CPU aging time by adding the job's page aging interval to its current accumulated CPU time. The definition of CPU time depends on the aging algorithm being used.

The used bit plays a role in swapout count aging. If a page's used bit is set, the page has been used since the last time the job's working set was aged. In this case, the following activity takes place:

1. The used bit is cleared.

2. The cyclic age of the page is reset to 0.

3. The CPU age of the page is reset to 0.

If a page's used bit is clear, the page has not been used since the last time the job's working set was aged. In this case, one of the following actions takes place:

- If the cyclic age is less than the value in the SWAPPING_AIC memory attribute, the cyclic age of the page is incremented by 1.

- If the cyclic age is greater than or equal to the value in the SWAPPING_AIC memory attribute, the page is removed from the job's working set. Pages aged out of the job's working set that have been modified are put into the available modified queue. Pages aged out of the job's working set that have not been modified are put into the available queue.

NOTE

The cyclic age used in job working set swapout count aging is the same variable as the cyclic age used in job working set cyclic aging. Thus, it is possible for the two types of aging to affect each other. However, if you use swapout count aging and job working set cyclic aging as they are intended, the interference should be minimal. The released job class default cyclic aging interval is 1,000 seconds. This setting effectively disables job working set cyclic aging. The intent is to use swapout count aging and CPU aging to age job working sets. Job working set cyclic aging is left as an alternative if the other two types of aging do not meet the site's needs. In this case, we recommend you consider setting the PAGE_AGING_INTERVAL scheduling attribute high so that CPU aging is effectively disabled. In addition, we recommend that you set the SWAPPING_AIC memory attribute high to effectively disable swapout count aging. As a result, the interference between the two algorithms is minimized.

In summary, the process defined by the swapout count aging algorithm attempts to remove any page that has not been used across the number of swapouts set by the SWAPPING_AIC memory attribute. However, the job working set aging algorithm does not attempt to reduce the job working set to a size smaller than the value of the minimum job working set. Using this algorithm before a job is swapped out minimizes the size of the job being swapped.

CPU Aging of Job Working Sets

CPU aging is based on page faulting and accumulated CPU time. Accordingly, this type of aging is most commonly initiated by the page fault processor. Every time a page fault occurs, the following activity occurs:

1. The page fault processor checks to see whether it is time to perform CPU aging on the job's working set.

2. One of the following actions is performed:

   - If the CPU time accumulated by the job since the last time the job's working set was aged is less than the value in the PAGE_AGING_INTERVAL scheduling attribute, the job's working set is not aged.

   - If the CPU time accumulated by the job since the last time the job's working set was aged is greater than the value of the PAGE_AGING_INTERVAL scheduling attribute, the job's working set is aged and the CPU age of each page is incremented.

CPU aging can also be initiated by the memory management periodic function manager under the following circumstances:

- The system is low on reassignable pages.

- The job in question is a CPU-bound job. A CPU-bound job is one that has accumulated three page aging intervals of CPU time without activating CPU aging; that is, it has accumulated CPU time without page faulting.

When a job is swapped out, it is also checked for this CPU-bound condition. If it has not been aged for three page aging intervals, the job is CPU-aged before it is swapped out. If a job is cyclically aged or swapout count aged, the process defined by either algorithm will reset the time when the job is to be CPU-aged next. It does this by adding the value in the PAGE_AGING_INTERVAL scheduling attribute to the job's current accumulated CPU time. CPU aging does not reset the time when the job is to be cyclically aged next.

Job working sets are kept in an ordered list. The order is based on the page's CPU age rather than its cyclic age. The most recently used pages are kept at the end of the list, and the least recently used pages are kept at the beginning of the list.

CPU aging attempts to examine every page in the job's working set. The working set is scanned from the least recently used page to the most recently used page. However, if at any time the job's working set size is less than or equal to the job's minimum working set, the current pass of this job's CPU aging is stopped.

A page's used bit also comes into play in CPU aging of job working sets. If a page's used bit is set, the page has been used since the last time the job's working set was aged. In this case, the following activity takes place:

1. The used bit is cleared.

2. The cyclic age of the page is reset to 0.

3. The CPU age of the page is reset to 0.

4. The page is relinked into the job's working set at the place in the list (beginning or end) that was most recently used.

If a page's used bit is clear, the page has not been used since the last time the job's working set was aged. The following actions are then performed:

1. The page's new CPU age is computed by adding the number of page aging intervals of CPU time that have been used by the job since it was last aged to its existing CPU age.

2. If the CPU age is greater than or equal to the age interval ceiling, the page is removed from the job's working set. Pages aged out of the job's working set that have been modified are put into the available modified queue. Pages aged out of the job's working set that have not been modified are put into the available queue.

When the system has examined the entire working set once, the least recently used page's CPU age is compared to the value of the AGE_INTERVAL_FLOOR memory attribute. If its CPU age is greater than or equal to that value, the page is removed from the job's working set. This is done for only one page per CPU aging pass.

Managing Page Aging Periodic Functions

The management of page aging periodic functions is controlled by a routine that itself is called periodically. The frequency with which this algorithm is called depends on the value of the PERIODIC_CALL_INTERVAL memory attribute.

These periodic functions include the following:

1. The system determines whether the number of reassignable pages in memory is less than the value of the AGGRESSIVE_AGING_LEVEL memory attribute.

   Reassignable pages are pages in the free and available page queues. If the number of reassignable pages is less than the value of the AGGRESSIVE_AGING_LEVEL memory attribute, the system considers itself low on memory and performs what is called aggressive aging. Aggressive aging means that the system ages queues even if their normal aging interval has not expired.

2. The system determines whether it is time to age the shared working set. If the age interval for the shared working set has expired, or if aggressive aging is in effect, the shared working set is aged.

3. The system again checks whether it is low on memory. If the system is low on memory, or if the age interval of the job working set has expired, it performs the following:

   - It tests each active job to see whether its working set should be aged. If the job's cyclic aging interval has expired, its working set is cyclically aged.

   - It tests whether the job fits its definition of a CPU-bound job (a job that has accumulated three page aging intervals of CPU time without page faulting). If the job is CPU-bound, its working set is CPU-aged.

Algorithm Descriptions in Pseudocode

The following figures contain pseudocode that logically describes the page aging algorithms. However, the actual code may be different. Tuning parameters that are site-definable as system attributes or class attributes are capitalized.

```
PROCEDURE manage page aging periodic functions

  {  Purpose:  Calls periodic page aging functions at correct    }
  {              times.                                          }

  { If system is low on memory, initiate "aggressive aging"   }
  {   to get more reassignable pages                           }
  IF reassignable pages < AGGRESSIVE_AGING_LEVEL THEN
      aggressive aging := true
    ELSEIF reassignable pages ≥ AGGRESSIVE_AGING_LEVEL THEN
      aggressive aging := false
  IFEND

  { Age shared queue if system is low on memory or if  }
  {   it is the normal time to age the shared queue    }
  IF (aggressive aging = true) OR
     (current time > next time to age shared queue) THEN
      age shared queue

  { Base the next time to age the shared queues on wall }
  {   clock time                                         }
      next time to age shared queue :=
        current time + SHARED_WORKING_SET_AGE_INTERVAL
  IFEND

  { Recompute whether the system is low on memory after   }
  {   aging the shared queue                               }
  IF reassignable pages < AGGRESSIVE_AGING_LEVEL THEN
      aggressive aging := true
    ELSEIF reassignable pages ≥ AGGRESSIVE_AGING_LEVEL THEN
      aggressive aging := false
  IFEND

  { Age job working sets if system is low on memory or if  }
  {   it is the normal time to age job working sets        }
  IF (aggressive aging = true) OR
     (current time > next time to age job working sets) THEN

    { Loop through all active jobs  }
    FOR job := first active job TO last active job DO

      { Compute job's accumulated CPU time based on which system  }
      {   aging algorithm is in force                              }
```

Figure 6-1. Manage Page Aging Periodic Functions

(Continued)

(Continued)

```
      IF AGING_ALGORITHM ≥ 4 THEN
          job CPU time := total job mode CPU time (job)
        ELSEIF AGING_ALGORITHM < 4 THEN
          job CPU time := total job mode CPU time (job) +
                         total monitor mode CPU time (job)
      IFEND

      { Age job cyclically if it is the normal time to do so  }
      IF current time > next cyclic aging time (job) THEN

        cyclic age job working set

      IFEND

      { Calculate if job is "CPU-bound"                      }
      IF job CPU time > next CPU time to age job working set (job) +
                       2 * PAGE_AGING_INTERVAL (job) THEN
          CPU-bound job := true
        ELSE
          CPU-bound job := false
      IFEND

      { Use job working set CPU aging algorithm if the system is  }
      {   low on memory or if the job is CPU-bound                }
      IF (aggressive aging = true) OR (CPU-bound job = true) THEN

        CPU age job working set

      IFEND

    FOREND

    { Compute the next time to consider aging job working sets  }
    {   based on wall clock time                                }
    next time to age job working sets :=
      current time + JOB_WORKING_SET_AGE_INTERVAL

  IFEND

PROCEND manage page aging periodic functions
```

Figure 6-1. Manage Page Aging Periodic Functions

```
PROCEDURE age shared queue

  { Purpose:  Cyclically age the shared working set page queues }
  {                                                             }
  { Called by:  manage page aging periodic functions            }

  { Age the shared queues until all pages have been aged. }
  {   Do not age below MINIMUM_SIZE                       }
  FOR each shared queue DO
    WHILE (there are still unexamined pages in shared queue) AND
          (pages remaining in shared queue > MINIMUM_SIZE) DO

      { If the page has been used since the last time the shared  }
      {   queue was aged, reset page aging data                   }
      IF current page's used bit is set THEN

        reset current page's used bit
        cyclic age (current page) := 0
        CPU age (current page) := 0

        { If the page was not used since the last time the shared  }
        {   queue was aged, increases the page's age by 1          }
        ELSEIF (current page's used bit is not set)                 AND
               (cyclic age (current page) < AGE_INTERVAL_CEILING)   THEN

          cyclic age(current page) := cyclic age(current page) + 1

        { If the page was not used since the last time the shared         }
        {   queue was aged and if its current age ≥ AGE_INTERVAL_CEILING, }
        {   remove the page from the shared queue                         }
        ELSEIF (current page's used bit is not set)                 AND
               (cyclic age (current page) ≥ AGE_INTERVAL_CEILING)   THEN

          remove page from shared queue

      IFEND

    WHILEND

  FOREND

PROCEND age shared queue
```

Figure 6-2. Cyclic Aging of Shared Queue

```
PROCEDURE cyclic age job working set

  { Purpose:  Cyclically age a job's working set            }
  {                                                         }
  { Called by:  manage page aging periodic functions        }

  { Compute the next time the job's working set should be   }
  {   cyclically aged based on wall clock time              }
  next cyclic aging time (job) := current time +
                                  CYCLIC_AGING_INTERVAL (job)

  { Compute job's accumulated CPU time based on which system  }
  {   aging algorithm is in force                             }
  IF AGING_ALGORITHM ≥ 4 THEN
      job CPU time := total job mode CPU time (job)
    ELSEIF AGING_ALGORITHM < 4 THEN
      job CPU time := total job mode CPU time (job) +
                      total monitor mode CPU time (job)
  IFEND

  { Compute the next time the job's working set should be   }
  {   CPU-aged based on CPU time                            }
  next time to CPU-age job working set :=
      job CPU time + PAGE_AGING_INTERVAL (job)

  { Age pages in the job's working set until all pages have  }
  {   been examined and as long as the job's working set is  }
  {   greater than MINIMUM_WORKING_SET                       }
  WHILE (there are still unexamined pages in job working set) AND
        (pages in job's working set > MINIMUM_WORKING_SET (job)) DO

    { If the current page has been used since the last time the  }
    {   job's working set was aged, reset the page's ages        }
    IF  current page's used bit is set THEN

        reset current page's used bit
        cyclic age (current page) := 0
        CPU age (current page) := 0
```

Figure 6-3. Cyclic Aging of Job Working Set

(Continued)

(Continued)

```
        { If the current page has not been used since the last  }
        {   time the job's working set was aged, and if its     }
        {   current age < 1, add 1 to the age of the page       }
        ELSEIF (current page's used bit is not set) AND
               (cyclic age (current page) < 1)      THEN

          cyclic age(current page) := cyclic age(current page) + 1

        { If the current page has not been used since the last  }
        {   time the job's working set was aged, and if its     }
        {   current age ≥ 1, remove the page from the job's     }
        {   working set                                         }
        ELSEIF (current page's used bit is not set) AND
               (cyclic age (current page) ≥ 1)     THEN

          remove page from job's working set

      IFEND

    WHILEND

  PROCEND cyclic age job working set
```

Figure 6-3. Cyclic Aging of Job Working Set

```
PROCEDURE swapout count age job working set

  { Purpose:  Use cyclic aging to age job's working set    }
  {             when it is swapped out for long wait       }
  {                                                        }
  { Called by:  Job swapper                                }

  { Compute the next time the job's working set should be  }
  {   cyclically aged based on wall clock time             }
  next cyclic aging time (job) := current time +
                                  CYCLIC_AGING_INTERVAL (job)

  { Compute job's accumulated CPU time based on which system  }
  {   aging algorithm is in force                             }
  IF AGING_ALGORITHM ≥ 4 THEN
      job CPU time := total job mode CPU time (job)
    ELSEIF AGING_ALGORITHM < 4 THEN
      job CPU time := total job mode CPU time (job) +
                   total monitor mode CPU time (job)
  IFEND

  { Compute the next time the job's working set should be  }
  {   CPU-aged based on CPU time                           }
  next time to CPU-age job working set :=
      job CPU time + PAGE_AGING_INTERVAL (job)

  { Age pages in the job's working set until all pages have  }
  {   been examined and as long as the job's working set is  }
  {   greater than MINIMUM_WORKING_SET                        }
  WHILE (there are still unexamined pages in job working set) AND
        (pages in job's working set > MINIMUM_WORKING_SET (job)) DO

    { If the current page has been used since the last time the  }
    {   job's working set was aged, then reset the page's age    }
    IF  current page's used bit is set THEN

        reset current page's used bit
        cyclic age (current page) := 0
        CPU age (current page) := 0
```

Figure 6-4. Swapout Count Aging of Job Working Set

(Continued)

(Continued)

```
        { If the current page has not been used since the last  }
        {   time the job's working set was aged, and if its     }
        {   current age < SWAPPING_AIC, add 1 to the page's     }
        {   cyclic age                                          }
        ELSEIF (current page's used bit is not set) AND
               (cyclic age (current page) < SWAPPING_AIC)    THEN

          cyclic age(current page) := cyclic age(current page) + 1

        { If the current page has not been used since the last  }
        {   time the job's working set was aged, and if its     }
        {   current age >= SWAPPING_AIC, remove the page from   }
        {   the job's working set                               }
        ELSEIF (current page's used bit is not set) AND
               (cyclic age (current page) ≥ SWAPPING_AIC)    THEN

          remove page from job's working set

      IFEND

    WHILEND

  PROCEND swapout count age job working set
```

Figure 6-4. Swapout Count Aging of Job Working Set

```
PROCEDURE CPU-age job working set

  { Purpose:  Age job's working set based on its accumulated  }
  {             CPU time                                      }
  {                                                           }
  { Called by:  manage page aging periodic functions          }
  {             page fault processor                          }
  {             job swapper                                   }

  { Compute job's accumulated CPU time based on which system  }
  {   aging algorithm is in force                             }
  IF AGING_ALGORITHM ≥ 4 THEN
      job CPU time := total job mode CPU time (job)
    ELSEIF AGING_ALGORITHM < 4 THEN
      job CPU time := total job mode CPU time (job) +
                     total monitor mode CPU time (job)
  IFEND

  { Compute the number of PAGE_AGING_INTERVALS that have      }
  {   elapsed since the last time the job's working set       }
  {   was CPU-aged.  This increment is computed, since this   }
  {   type of aging is not necessarily done every             }
  {   PAGE_AGING_INTERVAL.  Based on CPU time                 }
  elapsed aging intervals := (CPU time - CPU time last time working
              set was CPU-aged) / PAGE_AGING_INTERVAL

  { Compute the next time to CPU-age the job's working set    }
  {   based on the job's accumulated CPU time                 }
  next time to CPU-age job working set :=
      job CPU time + PAGE_AGING_INTERVAL (job)

  { The working set is examined from most recently used to    }
  {   least recently used pages.  It is aged until every      }
  {   page in the working set has been examined or until      }
  {   its working set size is ≤ the job's MINIMUM_WORKING_SET }
```

Figure 6-5. CPU Aging of Job Working Set

(Continued)

(Continued)

```
  WHILE (there are still unexamined pages in job working set) AND
        (pages in job's working set > MINIMUM_WORKING_SET (job)) DO

    { If the current page has been used since the last time the  }
    {   job's working set was aged, then reset the page's age    }
    {   and used bit, and relink the page into the job's         }
    {   working set as the most recently used page               }
    IF  current page's used bit is set THEN

        reset current page's used bit
        cyclic age (current page) := 0
        CPU age (current page) := 0
        put current page at beginning of most recently used pages

      { If the current page has not been used since the last   }
      {   time the job's working set was aged, and if the      }
      {   page's CPU age is greater than the AGE_INTERVAL_     }
      {   CEILING, remove the page from the job's working set }
      ELSEIF (current page's used bit is not set) AND
             (CPU age (current page) + elapsed aging intervals >
                   AGE_INTERVAL_CEILING)   THEN

        remove page from job working set

      { If the current page has not been used since the last   }
      {   time the job's working set was aged, and if the      }
      {   page's CPU age is  ≤ AGE_INTERVAL_CEILING,           }
      {   increment the page's CPU age by the                  }
      {   number of elapsed PAGE_AGING_INTERVAL increments     }
      ELSEIF (current page's used bit is not set) AND
             (CPU age (current page) + elapsed aging intervals
               ≤ AGE_INTERVAL_CEILING)   THEN

        CPU age (current page) := CPU age (current page) +
          elapsed aging intervals
    IFEND

  WHILEND

  { If the least recently used page in the job's working set  }
  {   has a CPU age > the AGE_INTERVAL_FLOOR, remove that     }
  {   page from the job's working set                         }
  IF (CPU age(least recently used page) > AGE_INTERVAL_FLOOR) THEN

    remove least recently used page from job working set
  IFEND

PROCEND CPU-age job working set
```

Figure 6-5. CPU Aging of Job Working Set

Job Swapping 7

Job Swapping 7

In addition to CPU scheduling and page aging, NOS/VE performs job swapping to ensure more efficient use of system resources. Job swapping refers to moving the job in and out of memory according to the following factors:

- Available resources
- Value specified by the MAX_TIME_SWAPOUT_IO_NOT_INIT system attribute
- Value specified by the MAXIMUM_SWAP_RESIDENT_TIME system attribute
- Value specified by the MINIMUM_AVAILABLE_PAGES memory attribute

The preceding system attributes are described in chapter 2, Adjusting System Attributes; the memory attribute is described in chapter 3, Managing Memory.

Swapping a Job Out

A job that is swapped out for long wait can exist in any of the following swapped-out states:

- *Swapped I/O not initiated.* The job has been aged according to its job working set swapout count and removed from the active job list (see chapter 6, Page Aging). The job has not been written to disk.

  The job remains in this state until one of the following conditions is satisfied:

  - Condition 1: the job goes ready and is swapped in; no I/O needs to be done.
  - Condition 2: the number of reassignable (free and available) pages falls below the level defined by the MINIMUM_AVAILABLE_PAGES memory attribute.
  - Condition 3: the job has been in long wait for the length of time defined by the MAX_TIME_SWAPOUT_IO_NOT_INIT system attribute plus the job's estimated think time. For an explanation of think time, see the description of the MAXIMUM_THINK_TIME system attribute in chapter 2, Adjusting System Attributes.

- *Swapped I/O complete.* The job advances to this state when either condition 2 or condition 3 occurs. The job has been written to disk. However, the job is still swap-resident (that is, the job's memory space has not been relinquished). The memory allocation for a job in this state can be freed immediately if the system needs it.

  A job remains in this state until one of the following conditions is satisfied:

  - Condition A: the job goes ready and is swapped in; no I/O needs to be done.
  - Condition B: a job in the system experiences page faulting while the number of reassignable pages falls below the level defined by the MINIMUM_AVAILABLE_PAGES memory attribute.
  - Condition C: the job has been in long wait for the length of time defined by the MAXIMUM_SWAP_RESIDENT_TIME system attribute plus the job's estimated think time.

- *Swapout complete.* The job advances to this state when either condition B or condition C occurs. All memory allocated for the job has been released. When the job goes ready, swapin I/O must be performed before it can be swapped back in.

Swapping a Job In

Every time the job scheduler executes to activate a job (that is, to initiate it or swap it into memory), the following activity takes place:

1. The job scheduler locates the queued job with the highest priority. This is the job with the highest dispatching priority as determined by its INITIAL_SERVICE_CLASS scheduling attribute and the value computed for the job's initiation priority. For an explanation of how a job's initiation priority is computed, see the description of the SELECTION_PRIORITY scheduling attribute in chapter 4, Job Scheduling.

2. The job scheduler locates the swapped job with the highest priority. This is the job with the highest dispatching priority as determined by its service class and the values specified for the job's SCHEDULING_PRIORITY scheduling attribute. The SCHEDULING_PRIORITY scheduling attribute is described in chapter 4, Job Scheduling.

3. Of the jobs identified in steps 1 and 2 (one queued, one swapped), the job scheduler activates the one with the highest dispatching priority. If both jobs have the same dispatching priority, the job scheduler activates the job with the highest scheduling priority.

Handling Bottlenecks in the System 8

Handling Bottlenecks in the System 8

Part of the performance tuning process consists of finding bottlenecks in the system. This chapter describes how you can identify and remove bottlenecks, as well as how to minimize the effects of bottlenecks that cannot be eliminated altogether.

Within the context of this chapter, a bottleneck is defined as a system resource at which sufficient queueing occurs to cause unacceptable job service.

Locating Bottlenecks

This section describes how to find bottlenecks in three major system resources: the CPU, memory, and the disk I/O subsystem. Disk I/O includes both the channels and the devices themselves.

The best source of data for finding bottlenecks is the system command DISPLAY_SYSTEM_DATA (DISSD). This command is documented in the NOS/VE Software Release Bulletin (SRB). The following section explains how to use various data items returned by the DISPLAY_SYSTEM_DATA command to diagnose system bottleneck problems.

Finding CPU Bottlenecks

You can locate possible bottlenecks in the CPU in two specific areas:

- CPU idle time
- Monitor request time

The CPU idle time data returned by the DISPLAY_SYSTEM_DATA command is the best data item to use for finding a CPU bottleneck in NOS/VE. If the total idle time is close to zero, there is probably a CPU bottleneck. NOS/VE also helps identify why the CPU is idle. CPU idle with I/O active reports the percentage of time the CPU was idle while there was disk I/O occurring in the system. CPU idle without I/O active reports the percentage of time the system was really idle. If neither the CPU nor the I/O subsystem was active, the system must have been idle. An example of this is a system with only an interactive workload and with all its users in think state.

The total CPU time used by monitor requests can help identify the cause of a CPU bottleneck. It can also eliminate monitor mode CPU time from consideration as a main contributor to a CPU bottleneck. Looking at the relative percentage of individual monitor requests can help identify types of system activity (such as page faults and swapping) that are causing CPU bottlenecks.

Finding Memory Bottlenecks

Examine the following areas when searching for memory bottlenecks:

- Disk-to-memory swapping ratio
- Scheduler memory wait
- Page queue sizes

The disk-to-memory swapping ratio constitutes the single most sensitive indicator of a memory bottleneck. This is also expressed as the relative value of the OI-to-OC and (R-to-SO + R-to-FA) swap state transitions. To understand what this ratio means, you need to be aware of the following facts regarding NOS/VE job swapping:

- When an interactive job sends its prompts to the terminal, it goes into an idle state waiting for a response from the user. This is an example of what is called a long wait. Jobs in long wait are swapout candidates.
- Swapout candidates go through the following three phases:
  1. The system prepares the job to be swapped out. The job is removed from the active job list, and its pages are threaded into the long wait page queue. No disk I/O or memory-to-memory transfer is performed at this point.
  2. The system selects a job from the long wait queue and actually writes that job's memory image to disk. The job's image now resides in both memory and on disk. Its pages now reside in the swap resident queue. If the job comes ready while in this state, it can still be swapped in from memory.

     This phase occurs under either of the following conditions:

     - The number of reassignable pages in the system falls below the site-definable threshold, MINIMUM_AVAILABLE_PAGES. (The number of reassignable pages in the system is the sum of the pages in the free queue plus the available queue.)
     - The time a job spends in long wait exceeds the value specified by the MAX_TIME_SWAP_IO_NOT_INIT system attribute.
  3. The system frees the memory image of a job in the swap resident queue. At this point, the job's image exists only on disk; it is completely swapped out. This phase takes place under one of the following conditions:
     - A page fault occurs for any job in the system while the number of reassignable pages is below the value specified by the MINIMUM_AVAILABLE_PAGES memory attribute.
     - If the time a job spends in swap resident exceeds the time specified by the MAXIMUM_SWAP_RESIDENT_TIME system attribute.

In this context, the OI-to-OC and (R-to-SO + R-to-FA) swapout state transitions have the following meanings:

Transition	Description
OI-to-OC	The number of swapouts that actually caused disk I/O to take place. Typically, they occur when memory availability is below a specified threshold.
R-to-SO + R-to-FA	The number of all long waits that occurred.

Any comparison of the above counts has to take into account the number of swaps that could have gone to disk (R-to-SO + R-to-FA) against the number of swaps that actually went to disk (OI-to-OC), presumably because there were too few reassignable pages in the system.

Scheduler memory wait is the second set of data you can examine when looking for memory bottlenecks. The job scheduler keeps track of the number of times it attempted to schedule a job into memory but could not. There are two main reasons this could happen:

- Not enough memory remained in the system to swap the job in. This is the *memory wait - no preempt* data item.
- The job could not get an AJL (active job list) ordinal. This could be either because the class the job belongs to was at the limit specified by the MAXIMUM_ ACTIVE_JOBS system attribute or because there were no more AJL ordinals in the system. This is tracked by the *wait for ajlo - no preempt* data item.

The third place to look for a memory bottleneck is in the number of pages in each of the page queues. This is not the best indicator of a memory bottleneck, but it can be valuable information when you are trying to tune the system. For example, the size of the free queue and the available queue shows how much memory is not being exclusively used by anyone. The important thing to remember about this data is that it is a single instantaneous sample taken at the time the DISPLAY_SYSTEM_DATA command is executed.

Finding Disk I/O Bottlenecks

When trying to diagnose a disk bottleneck, examine the following data items:

- CPU idle with I/O active
- Channel utilization
- Channel average queue size
- Average wait in queue before processing

The amount of time the CPU is idle when the I/O devices are active gives an estimate of the I/O time that would ideally have overlapped with CPU usage, but could not. If the CPU is idle only when it is waiting for I/O, any reduction in this idle time should translate directly into improved throughput for the system.

By itself, channel utilization is generally not a good indicator of an I/O bottleneck. However, when used in conjunction with the other data items in this list, it can reveal much about the nature of the I/O workload in the system.

The channel average queue size indicates the average number of disk requests queued at the channel. This count includes the request currently being processed by the channel. Data for this item is collected only when there is at least one request at the channel. If there are no requests at the channel, no data is collected for this idle time. The net value of these two points is a theoretical minimum value of 1.

If the channel average queue size is large and the channel utilization is low, this is an indication that the I/O workload sees spikes rather than constant use. This condition can be caused by a single job that generates I/O requests in clusters and then waits for them all to be satisfied.

If the channel average queue size is medium to large and the channel utilization is relatively high, this is indicative of a more chronic channel bottleneck.

The average wait in queue before processing indicates the number of milliseconds an average disk request must wait in the I/O queue before it is processed by the channel. This number is a function of both the average queue size and the average time it takes to process an I/O request.

Example

The following is an example of bottleneck identification using the data items described previously. This example is taken from an IM/DM benchmark with both updates and queries being performed. In addition, the data reflects performance with journaling turned on (that is, with the database manager tracking all updates to the database).

The example reflects the following hardware configuration:

- CYBER 930 with 32 megabytes memory
- Two disk channels
- Six 9836 disk spindles
- Twenty-five interactive terminals

The example assumes that all interactive users are logged in and signed on to the database. In addition, it is assumed that all terminals are running for approximately the same time interval.

CPU Data

The following table shows the actual data obtained from an untuned version of the system. In addition to the data items described earlier, the table indicates the time interval over which the data is collected, as well as the average response time.

Item	Value
Elapsed time (sec)	1,634
Average response time (sec)	3.9
CPU idle time (%)	
With I/O active	7
Without I/O active	5
Total	12
Monitor request CPU time (sec)	
Page fault	98
Process IO	68
Write modified pages	12
Monitor swap requests	44
Total monitor	366

This data reveals a CPU bottleneck, since there is little CPU idle time. The majority of the idle time is with I/O active; therefore, if we can reduce this small amount of idle time, we can have even more use of the CPU.

The monitor request data shows that a significant portion of monitor mode CPU time is used for page fault processing and for processing I/O.

Memory Data

The next table lists the memory data. This data shows that there is no memory bottleneck on this system. Almost all of the long wait swapouts (R-to-SO + R-to-FA) remain cached in memory. There is very little swapping to disk (13 out of 4,125).

Item	Value
Swap state transitions	
OI-to-OC count	13
(R-to-SO + R-to-FA) count	4,125
Scheduler statistics	
Memory wait–no preempt	0
Wait for ajlo–no preempt	0
Page queue sizes	
Available	4,031
Shared queues	639
Mainframe wired	585
Job fixed	367
Job working sets	2,384
Number of active jobs	25

The other item of interest is the size of the available queue. It is very large relative to the other page queues. This indicates that we have quite a bit of memory to work with before we need to worry about encountering a memory bottleneck.

Disk I/O Data

The I/O data shown in the next table indicates that there is not much queueing for I/O. While the average channel queue size is larger than its theoretical minimum value of 1, it is not much larger. The average wait in queue before processing (expressed in msec units) reinforces the diagnosis of little queueing.

Item	Value
Channel utilization (%)	
Channel 1	11
Channel 2	19
Channel average queue size	
Channel 1	2
Average wait in queue before processing	46

We have made the diagnosis of a CPU bottleneck with some small disk I/O queueing that prevents us from taking full advantage of the CPU resources available to us. Now we need to formulate a tuning response to this situation. The next table includes the data to help us do that.

Item	Value
Disk request totals	
Reads	4,824
Writes	24,026
Page faults per second	
Available	17
Available modified	2
Disk	2
New	3
Total	24
Average swap file size	76

In the preceding table, note that the number of writes to the disks is much larger than the number of reads from the disk. One way this can occur is for a job to repeat the cycle of modifying a page, letting it age out of the job's working set (which would cause the data to be written to disk), reclaiming the page from the available queue, and then modifying the page again.

This diagnosis is strengthened if we look at the system page fault data; most of the page faults are being satisfied from the available queue.

After supplying some general information about how to alleviate bottlenecks, we will analyze this example further.

Alleviating Bottlenecks

The following subsections describe tuning steps that can be taken to fix bottlenecks. Again, we consider three major system resources: CPU, memory, and disk I/O. Hardware solutions to fixing bottlenecks are not considered here. In addition, released default values are the assumed starting points.

Fixing CPU Bottlenecks

CPU bottlenecks are the most difficult to fix through tuning. In most cases, all you can do is reduce the demand on the CPU.

The following measures help reduce CPU demand:

- Minimize dual-state partner overhead. See chapter 4, Job Scheduling, for more information.
- Dedicate one CPU to NOS in a dual-CPU system.
- Optimize applications on NOS/VE. See the information on application efficiency in the NOS/VE Object Code Management manual.
- Reduce the number of jobs allowed to execute in the system.
- Reduce NOS/VE memory management.

To dedicate one CPU to NOS, set the DEDICATE_A_CPU_TO_NOS system attribute to one (see chapter 2, Adjusting System Attributes).

Reducing the amount of CPU time used by NOS/VE memory management is perhaps the most important step to follow in fixing CPU bottlenecks. This measure should take precedence over differentiating service using job scheduling.

Fixing Memory Bottlenecks

Use one or both of the following techniques to fix memory bottlenecks:

- Reduce working set sizes; this step increases the rate at which pages are aged out.
- Increase long wait swapping to disk.

The most direct way of fixing a memory bottleneck is to reduce the memory demand of the system workload. NOS/VE memory management gives the site the capability to perform tuning that can affect the working set job sizes. The recommended way to do this is to age pages out of a job's working set more quickly. This is accomplished by reducing the PAGE_AGING_INTERVAL scheduling attribute. The same idea applies to the shared queues by adjusting the SHARED_WORKING_SET_AGE_INTERVAL memory attribute.

The other technique that can be used to ease a memory bottleneck is to let fewer swapped jobs stay cached in memory in the long wait queue or the swap-resident page queue. The way to do this is to increase the value of the MINIMUM_AVAILABLE_PAGES memory attribute.

NOTE

Be aware that anything you do to reduce the memory demand on the system increases the I/O workload of the system. In NOS/VE, memory and disk I/O are, in a sense, just different types of the same concept: storage. If the total amount of storage in the system stays the same and the actual storage demand of the system workload stays the same, reducing the amount of this storage demand on one storage resource must increase the storage demand on another system resource. Thus, a *push-pop* effect is at work; pushing down on the memory demand causes the disk I/O demand to pop up. The tuning challenge is to find the optimal balance between disk and memory demand. As a result, a memory-bound, I/O-bound system is more difficult to tune satisfactorily.

Fixing Disk I/O Bottlenecks

Because of the nature of disk I/O, the following tuning steps will *not* cause an increase in memory demand:

- Make sure that the disk I/O workload is evenly distributed across all channels and disks. This is the most important of all the disk I/O tuning steps. If a single disk is experiencing a bottleneck while the others are idle, then simply off-loading some of the busier files to other disks can alleviate the disk bottleneck without increasing memory demand. The REQUEST_MASS_STORAGE command (described in the NOS/VE System Performance and Maintenance manual, Volume 2) can be used for this purpose.

- Transfer more data with each seek. This can be done either by increasing page size or by reducing the value of the PAGE_STREAMING_THRESHOLD and/or the PAGE_STREAMING_PRESTREAM memory attributes. Any of these options run the risk of increasing memory demand. The risk is that, with larger data transfers, the system is bringing in more unused data than it would with smaller data transfers. Thus, more memory is wasted for the same amount of useful memory demand.

The following measures also reduce disk I/O demand but generally result in an increase in memory demand:

- Reduce disk paging I/O. This means reducing the number of page faults satisfied from disk, either by keeping pages in job working sets longer or by keeping them in the available queue. You can keep pages in working sets longer by increasing the values of the PAGE_AGING_INTERVAL or MINIMUM_WORKING_SET scheduling attributes. We recommend using PAGE_AGING_INTERVAL, since MINIMUM_WORKING_SET fixes a limit for all jobs in a class independent of memory size. However, careful use of MINIMUM_WORKING_SET can be very helpful under some conditions. The memory attribute SHARED_WORKING_SET_AGE_INTERVAL and the individual shared queue attributes of AGE_INTERVAL_CEILING and MINIMUM_SIZE can also be used to hold pages longer in working sets.

- Control the page streaming mode of page fault processing. Page streaming provides an improvement in overall system performance by reading multiple pages in response to a page fault when the page fault occurs within a file that is being processed sequentially. The page fault processor first detects that page faults are occurring sequentially within a segment. It then initiates prestream mode, in which page faults are satisfied by reading in several pages at a time. If page faults continue sequentially, the page fault processor initiates page streaming mode in which a sufficient number of pages are read ahead to stream pages from the disk.

  Since the detection of sequential file processing and the initiation of page streaming is automatic, most sites will not need to make any changes to this area of page fault processing. However, the following memory attributes are provided to tune page streaming. These memory attributes are described in chapter 3, Managing Memory.

  - Use the PAGE_STREAMING_PRESTREAM memory attribute to adjust the number of page faults that will cause the page fault processor to initiate prestream mode.

  - Use the PAGE_STREAMING_THRESHOLD memory attribute to specify the amount of data processed sequentially before the page fault processor initiates page streaming mode.

  - Use the PAGE_STREAMING_READS memory attribute to specify the number of page streaming transfer units that are read at one time when operating in page streaming mode.

  - Use the PAGE_STREAMING_RANDOM_LIMIT memory attribute to specify the number of random page faults that will terminate page streaming mode.

- Reduce the number of active jobs. This reduces the total working set demand which allows more memory to be used for page caching. Page caching involves removing pages from memory into a storage area that allows for more rapid access by the CPU than is possible under secondary storage.

  To reduce the number of active jobs in memory, you can increase the number of reassignable pages the job scheduler attempts to maintain in the system. This is done by increasing the value of the target memory value in the SCHEDULING_MEMORY_LEVELS scheduling attribute.

  You can also change the number of cached pages in the system by changing the MINIMUM_AVAILABLE_PAGES memory attribute. Increasing this value causes the swapping cache to be smaller, and thus allows for a larger paging cache. (Unlike a paging cache, a swapping cache contains entire jobs rather than selected pages.)

- Reduce the amount of swapping to disk. The technique for doing this is to tune the system so the effective size of the memory swapping cache is increased. This is done by decreasing the value of the MINIMUM_AVAILABLE_PAGES memory attribute.

  In addition to a memory availability threshhold, NOS/VE sets a time limit for jobs to be in the swapping cache. This is because the system can generally make better use of memory for active users than caching the job of a logged-in user who is not currently using the system. Increasing the values of the MAXIMUM_SWAP_RESIDENT_TIME and MAX_TIME_SWAP_IO_NOT_INIT system attributes can increase the amount of time a job stays in the swapping cache.

  Changing the length of a WAIT that defines long wait swapping can affect whether or not a job is a swap candidate. Increasing the values of the LONG_WAIT_SWAP_TIME and LONG_WAIT_FORCE_SWAP_TIME system attributes can reduce the number of swaps to disk simply by reducing the number of times jobs become swapout candidates by going into long wait. However, these measures are not as generally useful for reducing swapping to disk as those described earlier.

Certain tradeoffs occur between swap caching and page caching. At any instant, the system is using a certain number of pages for running active jobs. These pages are distributed among the shared queues, job working set queue, job fixed queue, and mainframe-wired queue. The other pages in the system are available for caching. The tuning decisions the site makes can affect how these pages are distributed between the paging and the swapping cache. Tuning to increase the size of the swapping cache necessarily decreases the number of pages in the paging cache. Tuning to decrease the size of the swapping cache will increase the number of pages that can be used for the paging cache. You must determine which type of disk I/O you want to reduce: swapping or caching. The optimal solution is not to eliminate one type of disk I/O or the other, but to find the proper balance between the two types of I/O. Swapping requests occur less often than paging requests; however, because each swapping request causes more data to be transferred, it takes fewer of them to dominate a disk I/O workload.

Example Revisited

From the example in this chapter, we conclude the following:

- A CPU bottleneck exists.
- The disk I/O is not a bottleneck but is causing idle CPU time.
- CPU time is being used to process page faults from the available queue.
- Most of the disk I/O is caused by modified pages being aged out of working sets.

The tuning goals are aimed at reducing the system's CPU overhead as much as possible and at improving the efficiency of the system by reducing the CPU idle time with I/O active. The following strategy is designed to accomplish these goals:

- Because a significant amount of monitor mode CPU time is spent processing page faults from the available queue, reduce the number of page faults from the available queue. To do this, keep pages in working sets longer.
- To reduce the write requests to disk, and thus reduce the CPU idle time with I/O active, keep modified pages in job working sets longer.

NOTE

We do not recommend that you apply the tuning changes reflected in this example to all cases. Tuning requirements change according to mainframe model and configuration.

Tuning Changes

The following table shows the tuning changes used to implement this strategy:

Attribute	Base	Tuned
System attribute		
Shared working set age interval	8,000,000	20,000,000
Long wait swap time	6,000,000	600,000,000
Long wait force swap time	6,000,000	600,000,000
Interactive class attribute		
Minimum working set	20	120

In the example, the most important tuning change is increasing the minimum working set for the interactive class from 20 pages to 120 pages. This has the effect of holding modified and unmodified pages in job working sets for a longer time. Another way of accomplishing the same thing is to increase the page aging interval. Had this alternative been chosen, the value of this attribute would have at least doubled. The MINIMUM_WORKING_SET scheduling attribute was used for this system because there was so much spare memory.

In addition, the value of the SHARED_WORKING_SET_AGE_INTERVAL memory attribute is increased from 8 to 20 seconds. This causes shared pages to stay in the shared working set much longer. Although this was not a major problem, sufficient memory facilitated this minor tuning change.

Finally, because the example consists of an IM/DM workload with more long wait swaps than anticipated, the values of the LONG_WAIT_SWAP_TIME and LONG_WAIT_FORCE_SWAP_TIME system attributes are incremented to 600 seconds from their default values of 6 seconds.

The tuning methodology reflected in this example is strongly recommended:

1. Make a list of problem areas.
2. Create a logical description of what you want to accomplish.
3. Formulate these goals in terms of specific tuning parameter changes.

You will find this approach more effective than making random changes to parameters that merely seem relevant.

Effects on CPU Performance

The following table shows the effects of the tuning changes introduced earlier in this section on CPU performance:

Item	Base	Tuned
Elapsed time	1,634	1,714
Average response time (sec)	3.9	1.4
CPU idle time (%)		
With I/O active	7	3
Without I/O active	5	8
Total	12	12
Monitor request CPU time		
Page fault	98	27
Process I/O	68	57
Write modified pages	12	9
Monitor swap requests	44	24
Total monitor	366	251

These performance results indicate the following:

- The data collection intervals for the base and tuned runs are approximately the same. This means that comparison of data items that are time-dependent, such as counts or CPU seconds, has some meaning.
- Response time is improved.
- The amount of time the CPU is idle with I/O active is significantly reduced.
- The total idle time remains the same as before. This is probably because jobs go in and out much quicker than during the base run.
- The total monitor mode CPU time is reduced by over 100 seconds. Most of this improvement results from a reduction in the amount of time spent processing page faults.

Effects on Memory Management

The following table shows the effects of the tuning changes on memory management:

Item	Base	Tuned
Swap state transitions		
OI-to-OC count	13	4
(R-to-SO + R-to-FA) count	4,125	4,974
Scheduler statistics		
Memory wait–no preempt	0	0
Wait for ajlo–no preempt	0	0
Page queue sizes		
Available	4,031	2,438
Shared queues	639	477
Mainframe wired	585	599
Job fixed	367	168
Job working sets	2,384	1,611
Number of active jobs	25	6

The memory data shows that none of the tuning changes actually increase memory demand to the point of creating a memory bottleneck. While this data shows the base run having more pages in job working sets, computing the average number of pages in a working set per active job shows the average working set size to be higher in the tuned run.

Effects on Disk I/O

The I/O data shown in the next table indicates a reduction in the I/O demand and in I/O queueing:

Item	Base	Tuned
Channel utilization (%)		
Channel 1	11	8
Channel 2	19	8
Channel avg. queue size		
Channel 1	2	1
Channel 2	3	1
Average wait in queue before processing (msec)	46	17

Additional Effects

The following table shows additional data affected by the tuning changes:

Item	Base	Tuned
Disk request totals		
Reads	4,824	3,291
Writes	24,026	13,408
Page faults per second		
Available	17	3
Available modified	2	0
Disk	2	2
New	3	0
Total	24	5
Average swap file size	76	131

Here we also see desired changes. The write requests to disk are significantly reduced, as are the number of page faults from the available queue. The average swap file size shows that the tuning changes performed do in fact cause more pages to stay in job working sets.

We can summarize the effects of the tuning changes as follows:

- By reducing the number of page faults, you reduce the amount of CPU overhead required by the memory manager.
- Causing modified pages to stay in working sets longer reduces disk queueing to a minimum, thus helping the system make more efficient use of the CPU.
- Tuning the system facilitates better service to jobs.

Minimizing Bottlenecks

Situations arise where it is simply not possible to provide good service to all jobs. Historically, the tuning response has been to select a set of jobs (usually interactive jobs) to receive good service and a set of jobs (usually batch) to receive a lower level of service. It is possible, however, to tune the system so that all jobs receive an adequate level of service. The method used to accomplish this goal is known as job class service differentiation. This approach applies only to service for both the CPU and memory; it does not apply to disk I/O.

CPU Bottleneck Service Differentiation

Service differentiation with respect to CPU bottlenecks must take into account the following two situations:

- A system that has both a CPU bottleneck and a memory bottleneck.
- A system that has only a CPU bottleneck.

If a system has both a CPU bottleneck and a memory bottleneck, CPU service differentiation can often be performed indirectly through memory service differentiation (described in the next section). This is because a job must first have access to memory to get access to the CPU. Before the advent of large memories, this was the most common type of service differentiation.

If a system has a CPU bottleneck but does not have a memory bottleneck, CPU service differentiation must be performed directly. The following types of CPU service differentiation are available:

- Dispatching priority
- Dispatching time slice
- Subpriority (for dual-state systems)
- Class maximum active jobs
- Class switch mechanism
- Memory bottleneck service differentiation (only on memory-bound systems)

The most direct way of differentiating CPU service between jobs is to give jobs of one class a higher CPU priority than jobs of another class. The difficulty with this approach is that, if there are enough jobs of the higher priority class, jobs of the lower priority class will never get access to the CPU.

Dispatching time slices (minor time slice and major time slice) are controlled by the DISPATCHING_CONTROL scheduling attribute (described in chapter 4, Job Scheduling, and chapter 5, CPU Scheduling). CPU-bound jobs from different classes tend to accumulate CPU time in the same proportion as the ratio of their major time slices.

The service class switch mechanism for differentiating CPU service involves dynamically adjusting CPU priorities based on switching to different service classes. By defining a circular set of service classes, you can favor jobs from one set of classes without completely locking out jobs from another set of classes. This alternative is explained under the DISPATCHING_CONTROL scheduling attribute description in chapter 4, Job Scheduling, and chapter 5, CPU Scheduling.

Memory Bottleneck Service Differentiation

Memory bottleneck service differentiation is characterized by the following:

- Jobs that are not gaining access to memory are placed in memory queues, and their memory priority is gradually increased.
- Jobs currently in memory have their memory priority adjusted according to their use of system resources.

Jobs exist in one of four states:

State	Description
Jobs in the input queue	Not yet allowed to execute.
Jobs in the swapin candidate queue	Initiated and want access to the CPU, but are currently not allowed access to memory.
Active jobs	Currently in memory.
Jobs in wait state	Initiated but not active; are not swapin candidates. An example is an interactive job in think state.

The following is a list of the scheduling attributes that can be used to tune memory priority. These attributes are described in chapter 4, Job Scheduling.

GUARANTEED_SERVICE_QUANTUM
SCHEDULING_PRIORITY
SELECTION_PRIORITY
SERVICE_FACTORS
CLASS_SERVICE_THRESHOLD

The following scheduling attributes can be used to affect the calculation of memory priority for jobs in the swapin candidate queue. These attributes are described in chapter 4, Job Scheduling.

SCHEDULING_PRIORITY
SELECTION_PRIORITY

The following scheduling attributes affect the memory priority of jobs in the input queue. These attributes are described in chapter 4, Job Scheduling.

SCHEDULING_PRIORITY
SELECTION_PRIORITY

Other Recommended Performance Tuning Activities

The following subsections describe other steps you can take to reduce system bottlenecks, thereby increasing system performance. These activities include the following:

- Adjusting system attributes with the SET_SYSTEM_ATTRIBUTE system core command.
- Dedicating system resources to NOS/VE on dual-state systems.
- Deactivating unnecessary statistics.
- Setting a limit on the number of initiated jobs.
- Defining disk residence for catalogs and files.
- Improving login/logout performance.

Adjusting System Attributes

Control Data sets the default values of the parameters of the SET_SYSTEM_ATTRIBUTE command to provide the best performance over a wide range of mainframes and workloads. (The parameters of the SET_SYSTEM_ATTRIBUTE command are referred to as system attributes.) The best setting for some of the system attributes, however, varies from one processor or workload to another. Therefore, you can modify the values of these attributes using the SET_SYSTEM_ATTRIBUTE command to improve NOS/VE performance as required. See chapter 2, Adjusting System Attributes, for a description of the system attributes.

NOTE

Change system attributes only with great care, because they are parameters that the system uses to govern itself. Changes to system attributes can dramatically affect how NOS/VE behaves, positively or negatively.

Dedicating Resources to NOS/VE on Dual-State Systems

In dual-state operations, the CPU is shared between NOS (or NOS/BE) and NOS/VE. For dual-state sites with little or no NOS (or NOS/BE) workload, you can improve NOS/VE performance not only by making certain adjustments to NOS/VE, but by adjusting NOS (or NOS/BE) to use minimal system resources. You can do one or more of the following:

On a NOS dual-state system:

- Using the subpriority field of the DUAL_STATE_PRIORITY_CONTROL scheduling attribute of the ADMINISTER_CONTROLS subutility, adjust the CPU priority in favor of NOS/VE. In a dual-CPU system, you can also set the DEDICATE_A_CPU_TO_NOS system attribute to 1 to dedicate CPU 0 to NOS.

- Idle MAGNET, RBF, RHF, and IAF. This allows interactive and batch access to NOS/VE only.

- If you want to run NOS/VE strictly in batch mode, idle NAM and PASSON. Deactivate the NOS/VE interactive system task IFEXEC using the following operator command:

  ```
  DEACTIVATE_SYSTEM_TASK IFEXEC
  ```

- If no input or output is going through the NOS system, idle BATCH I/O (BIO) and the IRHF job. In addition, deactivate the NOS/VE tasks RHINPUT and RHOUTPUT with the DEACTIVATE_SYSTEM_TASK command. Jobs run faster with BIO idled and RHINPUT and RHOUTPUT deactivated. You can reactivate all three after the run has been completed.

- To reduce NOS memory and NOS CPU requirements, idle all unnecessary NOS subsystems.

  To determine whether NOS has more memory than it needs, use the ICPD and ACPD commands of the NOS utility TRACER. Among other things, TRACER reports the amount of memory used by NOS and the amount available to NOS. The TRACER utility is described in the NOS 2 Analysis Handbook.

- To further reduce NOS memory requirements, do the following:

  - Reduce the size of the NOS tables by altering the values of the FNT, FOT, and EJT entries in the CMRDECK.

  - Reduce the number of NOS control points.

  - Modify the LIBDECK so that the only programs specified as central memory resident are those in frequent use. To determine which NOS programs should be specified as central memory resident, use the NOS PROBE utility, described in the NOS 2 Analysis Handbook.

- Once memory requirements have been reduced, you can reduce the memory allotted to NOS by changing the MINCM entry in the NOS CMRDECK. For information about the NOS CMRDECK, see the NOS 2 Analysis Handbook.

On a NOS/BE dual-state system:

- Using the subpriority field of the DUAL_STATE_PRIORITY_CONTROL attribute of the ADMINISTER_CONTROLS subutility, adjust the CPU priority in favor of NOS/VE.
- Idle all unnecessary NOS/BE subsystems.
- Once NOS/VE is running, idle the IRHF job to allow more memory for NOS/VE.
- If no NOS/BE input or output is occurring, idle JANUS. The NOS/VE tasks RHINPUT and RHOUTPUT can also be deactivated with the DEACTIVATE_SYSTEM_TASK command. Jobs run faster with JANUS idled and RHINPUT and RHOUTPUT deactivated. You can reactivate all three after the run has been completed.

Deactivating Unnecessary Statistics

To save system overhead in resources and processing time, deactivate the recording of those statistics that you no longer want to collect. For more information about deactivating statistics, see chapter 9, Statistics Facility.

Setting a Limit on Initiated Jobs

The number of jobs allowed to be initiated in the system is governed using the INITIATION_LEVEL attribute within the MANAGE_ACTIVE_SCHEDULING utility. Limiting the number of initiated jobs in the system can improve the performance of jobs already in the system.

If a service class is mapped to only one job class, the job mode CPU time for the job scheduler may be reduced slightly. This happens only if the value of the INITIATION_LEVEL attribute for that job class is less than or equal to the value of the MAXIMUM_ACTIVE_JOBS scheduling attribute for that service class.

The MAXIMUM_ACTIVE_JOBS system attribute defines the current limit to the number of jobs that can be active in memory. The MAXIMUM_ACTIVE_JOBS scheduling attribute defines the current limit to the number of jobs of a specified service class that can be active in memory. The MAXIMUM_ACTIVE_JOBS system attribute is described in chapter 2, Adjusting System Attributes; the MAXIMUM_ACTIVE_JOBS scheduling attribute is described in chapter 4, Job Scheduling.

Making Catalog and File Device Assignments

The way in which you define disk residence for files and catalogs is of critical importance, both to improve system performance and to enhance file recoverability if a device fails. Appropriate segregation of file types can improve I/O data transfers and avoid bottlenecks that degrade performance. Concentrating permanent file catalogs on a minimal number of devices simplifies the task of recovering lost files.

Use the CHANGE_MS_CLASS subcommand of the Logical Configuration Utility (LCU) to change file and catalog assignments. The CHANGE_MS_CLASS description in the NOS/VE System Performance and Maintenance manual, Volume 2, gives a number of suggestions to help you decide how to allocate the disk devices in your system.

Improving Login/Logout Performance

For NOS/VE Version 1.4.2 or later, the SYSTEM_PROLOG command is no longer required in the system prolog file ($SYSTEM.PROLOGS_AND_EPILOGS.SYSTEM_PROLOG). If your system prolog file contains this command, it can be deleted to improve login performance. If deleting this command makes your system prolog file empty, you can also delete the system prolog file to further improve login performance.

System epilog processing is also enhanced. NOS/VE performs the following actions before executing your job class and system epilog:

- Closes all open files.
- Closes any file that has a user file access procedure.
- Deletes all System Command Language (SCL) variables.
- Detaches all tape files.
- Removes all command libraries from the command list.
- Deletes all file connections except connections to $RESPONSE.
- Resets program attributes to their default login values.
- Deletes all files from the program library list.

If your system epilog file ($SYSTEM.PROLOGS_AND_EPILOGS.SYSTEM_EPILOG) contains a DETACH_ALL_FILES command, it can be deleted. If deleting this command makes your system epilog file empty, you can also delete the system epilog file to improve logout performance.

For Further Information

Refer to the following materials for further information about performance tuning:

- NOS/VE Object Code Management manual
- FORTRAN for NOS/VE: Topics for FORTRAN Programmers

Part II: Statistics Facility

Statistics Facility 9

Statistics Facility 9

This chapter discusses how statistics (units of collected data) are gathered and analyzed. The utilities and commands encompassing the handling of statistics in NOS/VE are known as the Statistics Facility. These commands and utilities include:

- Commands for activating and deactivating statistics
- MANAGE_PERIODIC_STATISTICS utility
- PROCESS_STORAGE command
- ANALYZE_BINARY_LOG utility

After providing an overview of NOS/VE statistics and the commands and utilities listed above, this chapter describes the groups of statistics that are available for analysis and inclusion in reports.

NOS/VE Statistics: An Overview

During system execution, NOS/VE produces statistical information to report on many aspects of system activity. Generally, this information falls into the following categories:

- Information about the kind and quantity of resources used by a job. This information can be used for accounting and performance evaluation.
- Information describing abnormal conditions that may occur in the system's hardware and software. This information can aid in the analysis of hardware and software failures.
- Information describing an audit trail for security-related operations. The audit trail includes a date/time stamp, the name of the job performing the operation, the identity of the user, the terminal name (if interactive), and the success or failure of the operation. This information can be used by authorized users to audit security-related operations. Refer to the NOS/VE Security Administration manual for details concerning security-related statistics.

Although NOS/VE operating system code constantly emits statistical information about system activity, most NOS/VE statistics must be explicitly activated before the information they produce is collected and written to a binary log file. The section called Activating and Deactivating Statistics explains how to activate the various types of system statistics to binary logs.

The Statistics Facility can also be used to report statistics information collected by application programs running on NOS/VE. In this case, the application is responsible for collecting the relevant data, placing it into a specific statistic, and directing NOS/VE to write the statistic information to a binary log.

The following sections explain the format of NOS/VE statistics and how they are collected and analyzed. Major topics include:

Statistic format	The type and structure of the information collected, and the names of the statistics.
Statistic frequency	Some statistics are emitted at predefined intervals; others are emitted as the result of an event.
Binary logs	The binary logs are the disk files to which statistics are written.
Statistic activation	For statistics to be written to a log, they must be activated. The Activating and Deactivating Statistics section later in this chapter describes how to do this.
Setting intervals	The MANAGE_PERIODIC_STATISTICS utility is used to define the time interval at which certain statistics (known as periodic statistics) are emitted. The MANAGE_ PERIODIC_STATISTICS Utility section later in this chapter explains these statistics and how to set their intervals.
Log analysis	The ANALYZE_BINARY_LOG utility is used to analyze and report on the statistical contents of a binary log. This is discussed in greater detail in the ANALYZE_BINARY_LOG Utility section later in this chapter.

Database Management System Parallels

The process of analyzing statistics is similar to manipulating a database management system. The following parallels can be drawn:

- The databases you work with are binary logs.
- The record types are the statistics themselves.
- Each statistic consists of various fields (for example, the statistic code, counters, and the time the statistic was emitted). All fields except the counter field can serve as key fields. The most important key fields are the statistic codes, the time, and the date.
- Data entry takes place in real time batch mode. The system automatically generates data and immediately writes it to a binary log.
- A report is generated (using the ANALYZE_BINARY_LOG utility). For this purpose, you name fields, supply report specifications, and specify the criteria by which data is selected from a binary log.

Format of a Statistic

Each statistic consists of a name (statistic code) and data. The statistic name is a two-letter identifier followed by a numeric code. The alphabetic identifiers classify the statistics into groups. The following groups are defined:

Statistic Identifier	Statistic Group
AV	Accounting
CB	COBOL compilation
CL	Command language
CM	Configuration management
ES	NOS/VE editor
FC	FORTRAN Version 1 compilation
FV	FORTRAN Version 2 compilation
JM	Job management
LG	Logging
MV	Mail/VE Version 1 and Version 2
NA	NAM/VE
OD	Optical disk
OS	Operating system
PM	Program management
SF	NOS/VE security audit

The code number within the statistic code is a unique integer identifying the statistic within its group. For example, the OS group includes statistics OS0 and OS4, among others. OS0 reports statistical information on page fault activity, and OS4 reports on disk I/O activity. Statistic code numbers from 0 to 9,999 are reserved for Control Data. Statistic code numbers from 10,000 to 19,999 are reserved for site codes and user-defined codes.

The data contained in a statistic cannot exceed 4,095 bytes. The data can be a string of characters, a list of integers, or a combination of both. The character string is called the descriptive data field. The individual integer values are called counters. The limit of 4,095 bytes includes the statistic header, the descriptive data field, and the counters. Generally, this means a statistic can have about 4,000 characters of descriptive data and no counters, about 500 counters and no descriptive data, or some combination of descriptive data and counters that total about 4,000 bytes.

Event vs Periodic Statistics

The system records a statistic either because some event has occurred, such as the start of a new job, or because you requested the information to be recorded at certain time intervals (for example, to generate a report showing the accumulated number of page faults from the available queue).

These two methods of emission reflect two kinds of statistics: event and periodic.

Event statistics	Are automatically emitted by the system when something happens, such as when a task or job terminates.
Periodic statistics	Are emitted at certain intervals of time. This time interval is defined using the MANAGE_PERIODIC_STATISTICS utility, described in this chapter. Only operating system (OS) statistics and most NAM/VE (NA) statistics are periodic.

Periodic Statistic Counters

Periodic statistics have two basic types of counters: incremental and nonincremental. The only type of nonincremental counter currently in use is a snapshot counter.

- Incremental counters contain values that are the result of adding a new value to the previous value of the counter. The value of this counter reports activity for the entire time the counter was used. An example of an incremental counter is counter 1 for the OS6 statistic (CPU idle time).

- Snapshot counters contain values that represent only the moment at which the value is obtained; no change in value over a time interval is being measured. An example of a snapshot counter is counter 1 for the OS0 statistic (number of free pages).

For information about reporting on periodic statistic counters, see the Remarks section of the ADD_FIELD subcommand description for the ANALYZE_BINARY_LOG utility later in this chapter.

Binary Logs

When the system or an application program emits a statistic and the statistic has been activated, the statistic is written to a binary disk file called a log. Statistics recorded in a log are time-stamped and entered sequentially.

For each statistic, the following information is written in the log:

- Date (month, day, year)
- Time (hour:minute:second.millisecond)
- System-supplied job name
- Global task identification
- Statistic name (identifier and code)
- Statistic data

Two types of binary logs exist: global and local.

Global Logs

Global logs are system-wide logs that record statistics for all jobs on the system. Global logs are permanent files and are protected from unauthorized user access. Both event and periodic statistics can be written to global logs.

The following types of global logs exist:

Global Log Name	Description
Account	Stores accounting statistics on file $SYSTEM.$ACCOUNT_LOG.
Statistics	Stores performance evaluation statistics on file $SYSTEM.$STATISTIC_LOG.
Engineering	Stores configuration management statistics (hardware and software failure reports) on file $SYSTEM.$ENGINEERING_LOG.
Job history	Stores job history statistics on file $SYSTEM.$HISTORY_LOG.
Security	Stores security-related statistics on file $SYSTEM.$SECURITY_LOG.

Global logs are terminated and copied to another file by entering the TERMINATE_LOG command within a System Operator Utility session. This command is documented in the NOS/VE Operations manual. The following commands activate and deactivate the recording of statistics to global logs (except security); these commands are described in Activating and Deactivating Statistics later in this chapter:

```
ACTIVATE_JOB_STATISTICS
ACTIVATE_SYSTEM_STATISTICS
DEACTIVATE_JOB_STATISTICS
DEACTIVATE_SYSTEM_STATISTICS
```

Refer to the ADMINISTER_SECURITY_AUDIT utility described in the NOS/VE Security Administration manual for a description of the commands that activate and deactivate the security global log.

Local Logs

Local logs record statistics emitted by a single job. Local logs are deleted when the job ends. Only event statistics can be emitted to local logs; periodic statistics cannot.

Two kinds of local logs exist:

Local Log Name	Description
Job account	Stores site-defined or system-defined accounting statistics in file $LOCAL.$JOB_ACCOUNT_LOG.
Statistics	Stores user-defined statistics or Control Data-defined performance evaluation statistics on file $LOCAL.$JOB_STATISTIC_LOG.

The following commands activate and deactivate the recording of statistics to local logs:

```
ACTIVATE_JOB_STATISTICS
ACTIVATE_SYSTEM_STATISTICS
DEACTIVATE_JOB_STATISTICS
DEACTIVATE_SYSTEM_STATISTICS
```

Steps in Using Statistics

The following are the basic steps in using statistics on NOS/VE:

1. Determine which statistics you are interested in.

   Decide which of the statistics emitted by NOS/VE contain the information you are interested in, or write your own program to collect information and emit your own statistics. The CYBIL System Interface manual contains a description of the program interface used to manage NOS/VE statistics. The NOS/VE statistics are listed later in this chapter.

2. Activate the statistics to one or more binary logs.

   If you are collecting information for all jobs in the system, you need to activate the statistics to one of the global logs by using the ACTIVATE_SYSTEM_STATISTICS command.

   If a statistic is periodic, you need to assign it to an emission set, enable the emission set, and assign an emission period using the MANAGE_PERIODIC_STATISTICS utility.

3. Perform the activities to be measured.

   This could be anything from executing a performance benchmark on a dedicated machine to running the system in a production environment.

4. Analyze and report on the statistics that have been recorded.

   The ANALYZE_BINARY_LOG utility analyzes and generates reports on statistics that have been recorded in a binary log. It lets you analyze an active binary log or a copy of a binary log (produced by the TERMINATE_LOG command, described in the NOS/VE Operations manual).

The NOS/VE Accounting Analysis System provides the capability to charge for resource consumption, recorded as statistics in either the job account log or the global account log. Additional information on the Accounting Analysis System can be found in the NOS/VE Accounting Analysis System manual.

Activating and Deactivating Statistics

Although NOS/VE continually emits statistics, the information generated for a particular statistic is not saved or reported unless the statistic has been activated. Activating a statistic causes the statistic to be written to one or more logs (binary files) from which the information can be extracted and analyzed. Once activated, the statistic continues to be logged until it is deactivated.

The following commands are available for activating and deactivating statistics:

```
ACTIVATE_JOB_STATISTICS
DEACTIVATE_JOB_STATISTICS
ACTIVATE_SYSTEM_STATISTICS
DEACTIVATE_SYSTEM_STATISTICS
```

Use the ACTIVATE_JOB_STATISTICS and DEACTIVATE_JOB_STATISTICS commands to control statistics logging for individual jobs in the system. ACTIVATE_SYSTEM_STATISTICS and DEACTIVATE_SYSTEM_STATISTICS control statistics logging on a system-wide basis.

Some statistics are automatically activated by the system and do not have to be explicitly activated and deactivated. These statistics are listed in the ACTIVATE_SYSTEM_STATISTICS command description later in this chapter.

Use the following commands to display job and system statistics:

```
DISPLAY_ACTIVE_JOB_STATISTICS
DISPLAY_ACTIVE_SYSTEM_STATS
```

ACTIVATE_JOB_STATISTICS Command

Purpose Instructs the Statistics Facility to begin recording statistics information for the job that issued the command. The information is written to selected binary logs.

Format **ACTIVATE_JOB_STATISTICS** or
ACTIVATE_JOB_STATISTIC or
ACTJS
  STATISTICS=list of statistic code
  LOGS=list of keyword
  LOCK=boolean
  STATUS=status variable

Parameters **STATISTICS** or **STATISTIC** or **S**

Specifies the names of one or more statistics to be activated for the job. This parameter is required.

LOGS or *LOG* or *L*

Specifies one or more logs to which statistics are to be written. You can specify the following keywords; the default is JOB_STATISTIC_LOG:

JOB_ACCOUNT_LOG or JAL

Local account log.

JOB_STATISTIC_LOG or JSL

Local statistics log.

ACCOUNT_LOG or AL

Global account log.

ENGINEERING_LOG or EL

Global engineering log.

HISTORY_LOG or HL

Global job history log.

STATISTIC_LOG or SL

Global statistic log.

LOCK

Specifies a boolean value indicating whether the specified statistics can be deactivated by a subsequent DEACTIVATE_JOB_STATISTICS command. The default is FALSE.

TRUE

Attempts to deactivate statistics from the specified logs are rejected.

FALSE

The specified statistics can be deactivated.

STATUS

Returns the completion status for this command.

Remarks

- The use of this command requires that the appropriate validation capability be active within a System Operator Utility session. The System Operator Utility is described in the NOS/VE Operations manual. The required validation capability is a function of the logs specified in the command. The logs and their validation capabilities are as follows:

Log	**Validation Capability**
JOB_ACCOUNT_LOG	ACCOUNTING_ADMINISTRATION
JOB_STATISTIC_LOG	None required
ACCOUNT_LOG	ACCOUNTING_ADMINISTRATION
ENGINEERING_LOG	CONFIGURATION_ADMINISTRATION
HISTORY_LOG	SYSTEM_DISPLAYS
STATISTIC_LOG	CONFIGURATION_ADMINISTRATION

- The ACTIVATE_JOB_STATISTICS command cannot be used to activate the AV6, JM0, JM1, or JM2 statistics, since these statistics are emitted prior to execution of the system prolog.

- Statistics activated by the ACTIVATE_JOB_STATISTICS command remain active until deactivated by a DEACTIVATE_JOB_STATISTICS command. Specifically, this means that:

  - Statistics activated by the ACTIVATE_JOB_STATISTICS command cannot be deactivated by the DEACTIVATE_SYSTEM_STATISTICS command.

  - A statistic activated by the ACTIVATE_JOB_STATISTICS command remains active regardless of any subsequent ACTIVATE_JOB_STATISTICS or ACTIVATE_SYSTEM_STATISTICS commands that may be entered to activate additional statistics.

ACTIVATE_SYSTEM_STATISTICS Command

Purpose

Instructs the Statistics Facility to begin recording statistics information for all jobs in the system. The information generated is written to selected binary logs.

Format

ACTIVATE_SYSTEM_STATISTICS or
ACTIVATE_SYSTEM_STATISTIC or
ACTSS
  STATISTICS=list of statistic code
  LOGS=list of keyword
  LOCK=boolean
  STATUS=status variable

Parameters

STATISTICS or **STATISTIC** or **S**

Specifies the names of one or more statistics to be activated. This parameter is required.

LOGS or *LOG* or *L*

Specifies one or more logs to which statistics are to be written. You can specify the following keywords; the default is STATISTIC_LOG:

JOB_ACCOUNT_LOG or JAL

Local account log.

JOB_STATISTIC_LOG or JSL

Local statistics log.

ACCOUNT_LOG or AL

Global account log.

ENGINEERING_LOG or EL

Global engineering log.

HISTORY_LOG or HL

Global job history log.

STATISTIC_LOG or SL

Global statistic log.

LOCK

Specifies a boolean value indicating whether the specified statistics can be deactivated by a subsequent DEACTIVATE_SYSTEM_STATISTICS command. The default is FALSE.

TRUE

Attempts to deactivate statistics from the specified logs are rejected.

FALSE

The specified statistics can be deactivated.

STATUS

Returns the completion status for this command.

Remarks

- The use of this command requires that the appropriate validation capability be active within a System Operator Utility session. The System Operator Utility is described in the NOS/VE Operations manual. The required validation capability is a function of the logs specified in the command. The logs and their validation capabilities are as follows:

Log	Validation Capability
JOB_ACCOUNT_LOG	ACCOUNTING_ADMINISTRATION
JOB_STATISTIC_LOG	CONFIGURATION_ADMINISTRATION
ACCOUNT_LOG	ACCOUNTING_ADMINISTRATION
ENGINEERING_LOG	CONFIGURATION_ADMINISTRATION
HISTORY_LOG	SYSTEM_DISPLAYS
STATISTIC_LOG	CONFIGURATION_ADMINISTRATION

- A number of statistics are automatically activated during deadstart; you do not have to activate them. These statistics and the logs in which they are recorded are as follows:

Statistic Name	Log
AV6 (begin accounting)	ACCOUNT_LOG and JOB_ACCOUNT_LOG
AV7 (end accounting)	ACCOUNT_LOG and JOB_ACCOUNT_LOG
AV9 (begin application)	ACCOUNT_LOG and JOB_ACCOUNT_LOG
AV10 (end application)	ACCOUNT_LOG and JOB_ACCOUNT_LOG
AV11 (record unit measured activity)	ACCOUNT_LOG and JOB_ACCOUNT_LOG
AV37 (record connect time and number of bytes transferred)	ACCOUNT_LOG and JOB_ACCOUNT_LOG
CM0 to CM7999 (all configuration management statistics)	ENGINEERING_LOG

- Statistic CL2 generates an exceptionally large amount of data. As a precaution against the unintentional activation of this statistic, you must enter the following command at the system console (in addition to the ACTIVATE_SYSTEM_STATISTICS command) before CL2 will be activated:

```
set_system_attribute command_statistics_enabled 1
```

- Statistics activated by the ACTIVATE_SYSTEM_STATISTICS command remain active until deactivated by a DEACTIVATE_SYSTEM_STATISTICS command. Specifically, this means that:
  - Statistics activated by the ACTIVATE_SYSTEM_STATISTICS command cannot be deactivated by a DEACTIVATE_JOB_STATISTICS command.
  - A statistic activated by the ACTIVATE_SYSTEM_STATISTICS remains active regardless of any subsequent ACTIVATE_JOB_STATISTICS or ACTIVATE_SYSTEM_STATISTICS commands that may be entered to activate additional statistics.
- Statistics JM4 to JM18 collect the statistical information output from the DISPLAY_JOB_HISTORY and DISPLAY_OUTPUT_HISTORY commands described in the NOS/VE Commands and Functions manual. All of the statistics JM4 to JM18 must be activated to ensure that these commands will work properly.

  The operator commands ACTIVATE_HISTORY_LOG and DEACTIVATE_HISTORY_LOG are provided to activate and deactivate the entire block of statistics JM4 to JM18. This is the recommended way of activating and deactivating these statistics. The ACTIVATE_HISTORY_LOG and DEACTIVATE_HISTORY_LOG commands are described in the NOS/VE Operations manual.

DEACTIVATE_JOB_STATISTICS Command

Purpose

Instructs the Statistics Facility to stop recording specified statistics information to selected binary logs. This command affects only statistics being recorded for the executing job.

Format

DEACTIVATE_JOB_STATISTICS or
DEACTIVATE_JOB_STATISTIC or
DEAJS
  STATISTICS=list of statistic code
  LOGS=list of keyword
  STATUS=status variable

Parameters

STATISTICS or **STATISTIC** or **S**

Specifies the names of one or more statistics to be deactivated for the job. This parameter is required.

LOGS or *LOG* or *L*

Specifies one or more logs for which the Statistics Facility is to be deactivated. You can specify the following keywords; the default is JOB_STATISTIC_LOG:

JOB_ACCOUNT_LOG or JAL

Local account log.

JOB_STATISTIC_LOG or JSL

Local statistics log.

ACCOUNT_LOG or AL

Global account log.

ENGINEERING_LOG or EL

Global engineering log.

HISTORY_LOG or HL

Global job history log.

STATISTIC_LOG or SL

Global statistic log.

STATUS

Returns the completion status for this command.

Remarks

- The use of this command requires that the appropriate validation capability be active within a System Operator Utility session. The System Operator Utility is described in the NOS/VE Operations manual. The required validation capability is a function of the logs specified in the command. The logs and their validation capabilities are as follows:

Log	Validation Capability
JOB_ACCOUNT_LOG	ACCOUNTING_ADMINISTRATION
JOB_STATISTIC_LOG	None required
ACCOUNT_LOG	ACCOUNTING_ADMINISTRATION
ENGINEERING_LOG	CONFIGURATION_ADMINISTRATION
HISTORY_LOG	SYSTEM_DISPLAYS
STATISTIC_LOG	CONFIGURATION_ADMINISTRATION

- This command is used to deactivate statistics activated by an ACTIVATE_JOB_STATISTICS command. It cannot be used to deactivate statistics activated by an ACTIVATE_SYSTEM_STATISTICS command.

DEACTIVATE_SYSTEM_STATISTICS Command

Purpose

Instructs the Statistics Facility to stop recording specified statistics information to selected binary logs. This command affects statistics being recorded for every job in the system.

Format

DEACTIVATE_SYSTEM_STATISTICS or
DEACTIVATE_SYSTEM_STATISTIC or
DEASS
  STATISTICS=list of statistic code
  LOGS=list of keyword
  STATUS=status variable

Parameters

STATISTICS or **STATISTIC** or **S**

Specifies the names of one or more statistics to be deactivated. This parameter is required.

LOGS or *LOG* or *L*

Specifies one or more logs for which the Statistics Facility is to be deactivated. You can specify the following keywords; the default is STATISTIC_LOG:

JOB_ACCOUNT_LOG or JAL

Local account log.

JOB_STATISTIC_LOG or JSL

Local statistics log.

ACCOUNT_LOG or AL

Global account log.

ENGINEERING_LOG or EL

Global engineering log.

HISTORY_LOG or HL

Global job history log.

STATISTIC_LOG or SL

Global statistic log.

STATUS

Returns the completion status for this command.

Remarks

- Use this command to deactivate statistics activated by the ACTIVATE_SYSTEM_STATISTICS command. You cannot use it to deactivate statistics activated by the ACTIVATE_JOB_STATISTICS command.
- The use of this command requires that the appropriate validation capability be active within a System Operator Utility session. The System Operator Utility is described in the NOS/VE Operations manual. The required validation capability is a function of the logs specified in the command. The logs and their validation capabilities are as follows:

Log	Validation Capability
JOB_ACCOUNT_LOG	ACCOUNTING_ADMINISTRATION
JOB_STATISTIC_LOG	CONFIGURATION_ADMINISTRATION
ACCOUNT_LOG	ACCOUNTING_ADMINISTRATION
ENGINEERING_LOG	CONFIGURATION_ADMINISTRATION
HISTORY_LOG	SYSTEM_DISPLAYS
STATISTIC_LOG	CONFIGURATION_ADMINISTRATION

- Do not deactivate configuration management statistics; they are necessary for equipment maintenance.
- For information about deactivating statistics JM4 to JM18, see the Remarks section of the ACTIVATE_SYSTEM_STATISTICS command description.

DISPLAY_ACTIVE_JOB_STATISTICS Command

Purpose Displays the names of statistics that are activated for the currently executing job (that is, statistics that were activated by an ACTIVATE_JOB_STATISTICS command).

Format **DISPLAY_ACTIVE_JOB_STATISTICS** or
DISPLAY_ACTIVE_JOB_STATISTIC or
DISAJS
OUTPUT=file
DISPLAY_OPTIONS=list of keyword
STATUS=status variable

Parameters *OUTPUT* or *O*

Specifies the name of the file to receive the output of this command. The default is $OUTPUT.

DISPLAY_OPTIONS or *DO*

Specifies the logs for which active statistics are to be displayed. You can specify one or more of the following keywords; the default is JOB_STATISTIC_LOG:

JOB_ACCOUNT_LOG or JAL

Local account log.

JOB_STATISTIC_LOG or JSL

Local statistics log.

ACCOUNT_LOG or AL

Global account log.

ENGINEERING_LOG or EL

Global engineering log.

HISTORY_LOG or HL

Global job history log.

SECURITY_LOG

Global security log.

STATISTIC_LOG or SL

Global statistic log.

STATUS

Returns the completion status for this command.

Remarks

The use of this command requires that the appropriate validation capability be active within a System Operator Utility session. The System Operator Utility is described in the NOS/VE Operations manual. The required validation capability is a function of the logs specified in the command. The logs and their validation capabilities are as follows:

Log	Validation Capability
JOB_ACCOUNT_LOG	ACCOUNTING_ADMINISTRATION
JOB_STATISTIC_LOG	None required
ACCOUNT_LOG	ACCOUNTING_ADMINISTRATION
ENGINEERING_LOG	CONFIGURATION_ADMINISTRATION
HISTORY_LOG	SYSTEM DISPLAYS
SECURITY_LOG	SYSTEM_ADMINISTRATION
STATISTIC_LOG	CONFIGURATION_ADMINISTRATION

DISPLAY_ACTIVE_SYSTEM_STATS Command

Purpose Displays the names of globally activated statistics (that is, statistics activated by means of the ACTIVATE_SYSTEM_STATISTICS command).

Format **DISPLAY_ACTIVE_SYSTEM_STATISTIC** or
DISPLAY_ACTIVE_SYSTEM_STATS or
DISASS
OUTPUT=file
DISPLAY_OPTIONS=list of keyword
STATUS=status variable

Parameters *OUTPUT* or *O*

Specifies the name of the file to receive the output of this command. The default is $OUTPUT.

DISPLAY_OPTIONS or *DO*

Specifies the logs for which active statistics are to be displayed. You can specify one or more of the following keywords; the default is JOB_STATISTIC_LOG:

JOB_ACCOUNT_LOG or JAL

Local account log.

JOB_STATISTIC_LOG or JSL

Local statistics log.

ACCOUNT_LOG or AL

Global account log.

ENGINEERING_LOG or EL

Global engineering log.

HISTORY_LOG or HL

Global job history log.

SECURITY_LOG

Global security log.

STATISTIC_LOG or SL

Global statistic log.

STATUS

Returns the completion status for this command.

Remarks The use of this command requires that the appropriate validation capability be active within a System Operator Utility session. The System Operator Utility is described in the NOS/VE Operations manual. The required validation capability is a function of the logs specified in the command. The logs and their validation capabilities are as follows:

Log	Validation Capability
JOB_ACCOUNT_LOG	ACCOUNTING_ADMINISTRATION
JOB_STATISTIC_LOG	None required
ACCOUNT_LOG	ACCOUNTING_ADMINISTRATION
ENGINEERING_LOG	CONFIGURATION_ADMINISTRATION
HISTORY_LOG	SYSTEM_DISPLAYS
SECURITY_LOG	SYSTEM_ADMINISTRATION
STATISTIC_LOG	CONFIGURATION_ADMINISTRATION

MANAGE_PERIODIC_STATISTICS Utility

Before they can be logged, periodic statistics must be activated with the ACTIVATE_SYSTEM_STATISTICS command. Periodic statistics must also be added to an emission set that is enabled and has a time interval defined. This is done by using the MANAGE_PERIODIC_STATISTICS utility. To use this utility, you must be within a System Operator Utility session and must be validated for either the CONFIGURATION_ADMINISTRATION capability or the SYSTEM_DISPLAYS capability. The System Operator Utility is described in the NOS/VE Operations manual.

There are two groups of periodic statistics: the OS statistics and all of the NA statistics except NA10. For more information about these statistics, refer to the NAM/VE Statistics and Operating System Statistics sections later in this chapter. One or more of these statistics may be added to an emission set. The time interval defined in the emission set determines how often the statistic is issued by NOS/VE. If an emission set's time interval (referred to as period) is set to 5, each statistic in the emission set is issued every 5 minutes. To be entered in a log, the statistic must also be activated. If the statistic is not activated, it is issued every 5 minutes and then lost, since the system would not write it to a log.

The MANAGE_PERIODIC_STATISTICS utility works with groups of one or more statistics called emission sets. An emission set consists of one or more periodic statistics that have been assigned a time interval. NOS/VE lets you define up to four periodic emission sets; each periodic emission set can have its own time interval and statistics. These periodic emission sets are known to the MANAGE_PERIODIC_STATISTICS utility only by the following keywords and their abbreviations:

SET_1 or S1
SET_2 or S2
SET_3 or S3
SET_4 or S4

You can also define one immediate emission set. The immediate emission set cannot have a time interval assigned. The statistics contained in the immediate emission set are issued once upon leaving the MANAGE_PERIODIC_STATISTICS utility. Use the immediate emission set to establish baseline values for statistics that also belong to periodic emission sets. The immediate emission set is known to the MANAGE_PERIODIC_STATISTICS utility only by the following keyword and its abbreviation:

IMMEDIATE_EMISSION or IE

Once you add one or more statistics to an emission set and define a time interval (period) with the ADD_STATISTIC and CHANGE_PERIOD subcommands of the MANAGE_PERIODIC_STATISTICS utility, you must enable the emission set with the CHANGE_STATE subcommand. Only then can the system begin recording statistics.

You can access the MANAGE_PERIODIC_STATISTICS utility with write privilege assigned or with read-only privilege assigned. The RESERVE_WRITE_PRIVILEGE parameter of the MANAGE_PERIODIC_STATISTICS command determines which privilege is chosen. With write privilege assigned, you can change emission sets by adding or deleting statistics, by changing the time interval, or by changing the state. With read-only privilege, you can only display the emission sets. Multiple users can execute the MANAGE_PERIODIC_STATISTICS utility in read mode. But, only one user can execute the MANAGE_PERIODIC_STATISTICS utility in write mode. This ensures that only one user is changing the emission sets at any given time.

Changes to emission sets are made to a working copy rather than to the emission sets being used by the system at that moment. The working copy is created when the MANAGE_PERIODIC_STATISTICS utility is started. Once changes are made, issue the QUIT subcommand to replace the emission sets being used by the system with the working copy.

The following command initiates the MANAGE_PERIODIC_STATISTICS utility:

```
MANAGE_PERIODIC_STATISTICS
```

The MANAGE_PERIODIC_STATISTICS utility includes the following subcommands:

Subcommand	**Description**
ADD_STATISTIC	Adds a statistic to an emission set.
CHANGE_PERIOD	Changes the time interval (period) of an emission set.
CHANGE_STATE	Enables or disables an emission set.
CLEAR_EMISSION_SETS	Copies an empty emission set into the working copy of emission sets.
DELETE_STATISTIC	Deletes a statistic from an emission set.
DISPLAY_EMISSION_SET	Displays an emission set.
QUIT	Terminates the MANAGE_PERIODIC_STATISTICS utility and either replaces the current system emission sets with the working copy of emission sets or discards the working copy.
READ_EMISSION_SETS	Copies the system's current emission sets into the working copy of emission sets.

These commands are described on the following pages. For a sample utility session, refer to Examples of Using Statistics later in this chapter.

MANAGE_PERIODIC_STATISTICS Command

Purpose Initiates execution of the MANAGE_PERIODIC_STATISTICS utility.

Format **MANAGE_PERIODIC_STATISTIC** or
MANAGE_PERIODIC_STATISTICS or
MANPS
READ_EMISSION_SETS=boolean
RESERVE_WRITE_PRIVILEGE=boolean
STATUS=status variable

Parameters *READ_EMISSION_SETS* or *RES*

Specifies a boolean value indicating whether emission sets are copied into the working copy when the MANAGE_PERIODIC_STATISTICS utility begins execution. The default is TRUE.

TRUE

A copy is made of the emission sets that the system is currently using. This is the working copy, which may be changed by using the utility's subcommands.

FALSE

An initialized copy of emission sets is made. In this copy, no statistics appear in the sets, all sets are disabled, and all time periods are set to 0. This blank working copy may be changed by using the utility's subcommands.

RESERVE_WRITE_PRIVILEGE or *RWP*

Specifies a boolean value indicating whether emission sets can be written during this MANAGE_PERIODIC_STATISTICS utility session. NOS/VE allows only one MANAGE_PERIODIC_STATISTICS utility session to be active at any given time with the RESERVE_WRITE_PRIVILEGE parameter set to TRUE. The default is TRUE.

TRUE

Emission sets can be written to a working copy during this MANAGE_PERIODIC_STATISTICS utility session.

FALSE

Emission sets can only be displayed; they cannot be written during this MANAGE_PERIODIC_STATISTICS utility session.

STATUS

Returns the completion status for this command.

Remarks

- The MANAGE_PERIODIC_STATISTICS utility defines, displays, and changes emission set definitions. To use this utility, you must be within a System Operator Utility session and must be validated for either the CONFIGURATION_ADMINISTRATION capability or the SYSTEM_DISPLAYS capability. The System Operator Utility is described in the NOS/VE Operations manual.

- The working copy is created when the MANAGE_PERIODIC_STATISTICS utility first begins execution with the MANAGE_PERIODIC_STATISTICS command. This working copy replaces the currently active system emission sets when the MANAGE_PERIODIC_STATISTICS utility is terminated with the QUIT subcommand as follows:

```
quit write_emission_sets=true
```

Examples Refer to Examples of Using Statistics later in this chapter.

ADD_STATISTIC Subcommand

Purpose Adds a statistic to a specified emission set.

Format

ADD_STATISTIC or
ADD_STATISTICS or
ADDS
  EMISSION_SET=keyword
  STATISTICS=list of name
  STATUS=status variable

Parameters

EMISSION_SET or **ES**

Specifies the emission set to which to add statistics. This parameter is required. You must specify one of the following keywords:

SET_1 or S1
SET_2 or S2
SET_3 or S3
SET_4 or S4
IMMEDIATE_EMISSION or IE

STATISTICS or **S**

Specifies the name of a statistic or list of statistics to be added to the emission set. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks

- Each emission set can have a maximum of 20 statistics.
- The emission set must be in the working copy of emission sets. For more information about the working copy, refer to the READ_EMISSION_SETS parameter and the RESERVE_WRITE_PRIVILEGE parameter of the MANAGE_PERIODIC_STATISTICS command.

Examples Refer to Examples of Using Statistics later in this chapter.

CHANGE_PERIOD Subcommand

Purpose Changes the time interval (period) at which the statistics in an emission set are periodically issued.

Format CHANGE_PERIOD or
CHAP
    EMISSION_SET=keyword
    PERIOD=integer or **time_increment**
    STATUS=status variable

Parameters **EMISSION_SET or ES**

Specifies the emission set whose period is to be changed. This parameter is required. You must specify one of the following keywords:

SET_1 or S1
SET_2 or S2
SET_3 or S3
SET_4 or S4

PERIOD or P

Specifies the time between each emission of the statistics in the emission set. You can specify an integer representing the time period in minutes, or you can specify a time_increment value. This parameter is required.

By specifying a time_increment value, you can change the unit of time measurement. The time_increment value is specified as a record consisting of one or more of the following fields:

(hours,minutes,seconds)

See the NOS/VE System Usage manual for time_increment values.

STATUS

Returns the completion status for this subcommand.

Remarks The emission set specified on the EMISSION_SET parameter must reside in the working copy of emission sets. For more information about the working copy, refer to the READ_EMISSION_SETS parameter and the RESERVE_WRITE_PRIVILEGE parameter of the MANAGE_PERIODIC_STATISTICS command.

Examples In the following example, the time interval for emission set SET_1 is changed to 30 seconds:

```
change_period emission_set=set_1 period=0:0:30
```

Refer also to Examples of Using Statistics later in this chapter.

CHANGE_STATE Subcommand

Purpose Enables or disables the specified emission set in the working copy.

Format

CHANGE_STATE or
CHAS
  EMISSION_SET=keyword
  STATE=keyword
  STATUS=status variable

Parameters

EMISSION_SET or **ES**

Specifies the emission set whose state is to be changed. This parameter is required. You must specify one of the following keywords:

SET_1 or S1
SET_2 or S2
SET_3 or S3
SET_4 or S4
IMMEDIATE_EMISSION or IE

STATE or **S**

Specifies the new state of the selected emission set. This parameter is required. You must specify one of the following keywords:

ENABLED

All statistics in the emission set are issued periodically by NOS/VE at the frequency specified by the emission set's period.

DISABLED

The statistics in the emission set are not issued at all.

STATUS

Returns the completion status for this subcommand.

Remarks

- If the state of the emission set is changed to ENABLED, any statistics in the emission set are emitted with the frequency specified by the emission set period. If the state is set to DISABLED, emission of the statistics in the set stops.
- An emission set must be enabled before its statistics can be emitted.

Examples Refer to Examples of Using Statistics later in this chapter.

CLEAR_EMISSION_SETS Subcommand

Purpose Copies an empty emission set into the working copy of emission sets.

Format

CLEAR_EMISSION_SET or
CLEAR_EMISSION_SETS or
CLEES
  EMISSION_SET=list of keyword
  STATUS=status variable

Parameters

EMISSION_SET or **ES**

Specifies the emission set(s) in the working copy to be overwritten with an initialized copy. This parameter is required. You must specify one of the following keywords:

SET_1 or S1
SET_2 or S2
SET_3 or S3
SET_4 or S4
IMMEDIATE_EMISSION or IE
ALL

If you specify the keyword ALL, all emission sets in the working copy are overwritten.

STATUS

Returns the completion status for this subcommand.

Remarks This command allows you to create a new emission set by adding to an empty, disabled emission set. Each emission set in the initialized copy is disabled, has no statistics, and has an infinite period.

DELETE_STATISTIC Subcommand

Purpose Deletes a statistic from an emission set.

Format **DELETE_STATISTIC** or
DELETE_STATISTICS or
DELS
  EMISSION_SET=keyword or **list of name**
  STATISTICS=keyword or **list of name**
  STATUS=status variable

Parameters **EMISSION_SET** or **ES**

Specifies the emission set from which statistics are to be deleted. This parameter is required. You must specify one of the following keywords:

SET_1 or S1
SET_2 or S2
SET_3 or S3
SET_4 or S4
IMMEDIATE_EMISSION or IE

STATISTICS or **S**

Specifies the name of one or more statistics to be deleted from the emission set. You can also specify the keyword ALL. If you specify ALL, all statistics in the specified emission set are deleted. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks The emission set specified on the EMISSION_SET parameter must be in the working copy of emission sets. For more information about the working copy, refer to the READ_EMISSION_SETS parameter and the RESERVE_WRITE_PRIVILEGE parameter of the MANAGE_PERIODIC_STATISTICS command.

DISPLAY_EMISSION_SETS Subcommand

Purpose Displays an emission set.

Format **DISPLAY_EMISSION_SET** or
DISPLAY_EMISSION_SETS or
DISES
EMISSION_SET=list of keyword
OUTPUT=file
STATUS=status variable

Parameters *EMISSION_SET* or *ES*

Specifies the emission set or sets to be displayed from the working copy. You can specify one of the following keywords; the default is ALL (all emission sets are displayed):

SET_1 or S1
SET_2 or S2
SET_3 or S3
SET_4 or S4
IMMEDIATE_EMISSION or IE
ALL

OUTPUT or *O*

Specifies the name of the file that will receive the emission set display. The default is $OUTPUT.

STATUS

Returns the completion status for this subcommand.

QUIT Subcommand

Purpose Terminates the MANAGE_PERIODIC_STATISTICS utility. In addition, it either replaces the current system emission sets with the working copy of emission sets or discards the working copy.

Format

QUIT or
QUI
    WRITE_EMISSION_SETS=boolean
    STATUS=status variable

Parameters *WRITE_EMISSION_SETS* or *WES*

Specifies a boolean value indicating whether the system's current emission sets are to be replaced by the working copy of emission sets. By default, the working copy replaces the system's current emission sets if any emission sets were changed during this MANAGE_PERIODIC_STATISTICS utility session. If no emission sets were changed, the working copy is discarded and the system continues to use its current emission sets.

TRUE

The working copy of emission sets replaces the current system emission sets if the RESERVE_WRITE_PRIVILEGE parameter of the MANAGE_PERIODIC_STATISTICS command is set to TRUE.

FALSE

The working copy of the emission sets is discarded; the system continues to use its current emission sets.

STATUS

Returns the completion status for this subcommand.

Examples Refer to Examples of Using Statistics later in this chapter.

READ_EMISSION_SETS Subcommand

Purpose Copies the emission sets being used by the system into the working copy of emission sets.

Format **READ_EMISSION_SET** or
READ_EMISSION_SETS or
REAES
  EMISSION_SET=list of keyword
  STATUS=status variable

Parameters **EMISSION_SET** or **ES**

Specifies the emission set(s) in the working copy to be overwritten with a copy of the version that the system is presently using. This parameter is required. You must specify one or more of the following keywords:

SET_1 or S1
SET_2 or S2
SET_3 or S3
SET_4 or S4
IMMEDIATE_EMISSION or IE
ALL

If you specify the keyword ALL, all emission sets in the working copy are overwritten.

STATUS

Returns the completion status for this subcommand.

Remarks

- This subcommand operates similarly to the MANAGE_PERIODIC_STATISTICS command with the READ_EMISSION_SETS parameter set to TRUE. That is, it overwrites the working copy of emission sets with the system's current emission sets so that you can edit that copy using the other subcommands of the MANAGE_PERIOD_STATISTICS utility. However, unlike the MANAGE_PERIODIC_STATISTICS command, this subcommand lets you specify which emission set or sets are to overwrite the corresponding emission set or sets in the working copy.

- The working copy is created when the MANAGE_PERIODIC_STATISTICS utility begins execution with the MANAGE_PERIODIC_STATISTICS command.

Examples Refer to Examples of Using Statistics later in this chapter.

PROCESS_STORAGE Command

The PROCESS_STORAGE command processes the complete file system catalog structure for specified families. This command also records information about the use of permanent file resources by users, and/or updates the permanent file space limit total accumulator values in the validation file for each family. To use this command, you must have system administrator privileges. The permanent file space information is reported by emitting statistics for:

- Individual catalogs (statistic AV13)
- Individual file cycles (statistic AV14)
- Catalogs, grouped by project member (statistic AV15)
- Files, grouped by project member (statistic AV16)
- Catalogs, grouped by user (statistic AV17)
- Files, grouped by user (statistic AV18)
- Catalogs, grouped by account member (statistic AV24)
- Files, grouped by account member (statistic AV25)

You control the information recorded by activating the appropriate statistics to one or more binary logs. The contents of these statistics are described in Accounting Statistics later in this chapter.

NOTE

Since statistics AV13 and AV14 (record storage space data for individual catalogs and file cycles) record large amounts of information that can fill up the binary logs, we recommend not activating these statistics unless you have a special need to account for catalogs and files on an individual basis.

The PROCESS_STORAGE command has the following format:

PROCESS_STORAGE or
PROS
  FAMILY=keyword or *list of name*
  PROCESSING_OPTION=keyword
  STATUS=status variable

FAMILY or *FAMILIES* or *F*

Specifies the name or names of the NOS/VE families whose permanent file catalog structure you want processed. You can also specify the keyword ALL. By default, information for all families in the system is recorded.

PROCESSING_OPTION or *PROCESSING_OPTIONS* or *PO*

Specifies a keyword indicating the processing options you want to use for this occurrence of the PROCESS_STORAGE command. You can specify the following keywords; the default is ALL:

ALL

Emits the permanent file space statistics and updates the permanent file space limit total accumulator values in the validation file.

EMIT_SPACE_STATISTICS

Emits the permanent file space statistics.

UPDATE_LIMITS

Upates the permanent file space limit total accumulator values in the validation file.

STATUS

Returns the completion status for this command.

ANALYZE_BINARY_LOG Utility

Because NOS/VE statistics are logged in binary format, the ANALYZE_BINARY_LOG utility is provided to read and report on the contents of the logs (called binary logs). This utility can produce either user-readable reports or reports that are used by programs for further analysis of a log's data. Copies of global logs can be produced using the operator command TERMINATE_LOG. The TERMINATE_LOG command is described in the NOS/VE Operations manual. Copies of local logs can be produced using the COPY_FILE command. The ANALYZE_BINARY_LOG utility can also read active binary logs.

Capabilities of the Utility

The ANALYZE_BINARY_LOG utility lets you do the following:

- Define criteria for selecting statistics from a binary log.
- Name fields.
- Generate logs and reports.
- Use multiple input files. This feature is especially helpful to sites that need to analyze logs spanning several days.
- Perform editing functions on collected data. For example, you can change and delete selected statistics, counters, data description fields, and report entries.
- Control the format of the logs and reports.
- Display the names, counts, and time ranges of statistics found in the input log.
- Generate a report without exiting the utility.

These capabilities are described in more detail in the following sections.

Defining the Data to Be Analyzed

The ANALYZE_BINARY_LOG utility has a data definition and naming capability. The basic unit of data in a binary log is the statistic. A statistic consists of a statistic code, statistic attributes, counters, and a descriptive data field. These terms are described as follows:

Term	Description
Statistic code	Name by which a statistic is identified. Examples of statistic codes are JM3, OS2, and CM6.
Statistic attribute	Information contained within a statistic. Statistic attributes include such items as the time the statistic was emitted or the name of the job that emitted the statistic. Statistic attributes appear as parameter specifications on subcommands of the ANALYZE_BINARY_LOG utility.

Term	Description
Counter	Array of integer values in a statistic. Not all statistics have counters. The maximum number of counters allowed is determined by the system constant sfc$max_number_of_counters (4,095).
Descriptive data field	String field within a statistic. This field contains either data or additional statistic attributes. The maximum length of this field is determined by the system constant sfc$max_descriptive_data_size (4,095).

Selections, Fields, and Puts

The following additional terms need to be defined: selection, field, and put. Using these entities, the ANALYZE_BINARY_LOG utility organizes and manipulates binary log information.

Term	Description
Selection	Defines occurrences of a statistic according to characteristics they have in common (that is, according to selection criteria). Examples of selection criteria are statistic code, emission time, and descriptive data field, described in the Selection Criteria subsection. The utility uses selection definitions to extract statistics, or portions thereof, from the input log. Each selection must have a unique name.
Field	Defines a portion of a statistic that will be extracted. Statistic fields are of six types: counter, descriptive data, statistic header, elapsed time, elapsed time calculation, and string. All fields have attributes that can be specified on certain utility subcommands. Some of these attributes include the field name, the selection naming the statistic containing the field, the counter number, the descriptive data field, the elapsed time field, and the elapsed time calculation field.
Put	Defines output specifications for the binary log analysis. Examples of report specifications are the format of a report line, column header information, and the manner in which numeric data should be calculated. PUT is a verb used in several ANALYZE_BINARY_LOG subcommands. Subcommands containing the word PUT are called output definition subcommands.

Selection Criteria

For every selection, characteristics are defined that serve as criteria for evaluating and selecting statistics. These characteristics are called selection criteria. They reside in a list specified when you define or modify a selection. If, when the ANALYZE_BINARY_LOG utility performs the evaluation, all the criteria established for the selection are satisfied, the statistic is selected. Once a selection criterion specified for a selection fails to be satisfied, the utility stops evaluating criteria for that selection and moves on to evaluate the next selection. If the selection has no criteria associated with it (that is, if the selection criteria list is empty), all of the statistics are selected.

The following table describes the selection criteria:

Selection Criteria	Description
Statistic code	Specifies the name of a statistic. This selection criterion specifies that only those statistics having the specified statistic code are to be included in the selection.
Emission time	Specifies a date range, time range, or date/time range. This selection criterion specifies that only those statistics emitted within the indicated range of time are to be included in the selection.
Descriptive data	Specifies a descriptive data field. This selection criterion specifies that only those statistics containing the specified descriptive data field or a substring of the descriptive data field are to be included in the selection. For more information, see Descriptive Data Substring Selection following this table.
Job predecessor	Specifies the name of a selection. This selection criterion determines whether the job emitting the current statistic also emitted another statistic that had previously been selected by a predecessor selection. For more information, see Job Predecessor following this table. Also refer to Joining Predecessors.
Task predecessor	Specifies the name of a selection. This selection criterion determines whether the task emitting the current statistic also emitted another statistic that had previously been selected by a predecessor selection. For more information, see Task Predecessor following this table. Also refer to Joining Predecessors.
System job name	Specifies the system-supplied name of a job. This selection criterion specifies that only statistics emitted by the named job are included in the selection.
Global task ID	Specifies a global task identifier. This selection criterion specifies that only statistics containing the global task identifier specified are included in the selection.

Descriptive Data Substring Selection

The ANALYZE_BINARY_LOG utility can select statistics for analysis by matching a substring or a subfield of the statistic's descriptive data field to a specified string. The descriptive data substring or subfield is specified by a record having the following format:

(*string, position, length, field_number, field_delimiter*)

By using the position, length, field_number, and field_delimiter fields, you can specify a substring or a subfield of the descriptive data field. If a match is found, the descriptive data selection criterion is satisfied.

Job Predecessor

The process by which the job predecessor selection criterion is handled is twofold and takes place whenever the input log is scanned. First, the ANALYZE_BINARY_LOG utility builds and maintains a list of system job names (as well as date/time values) associated with every emitted statistic selected by a predecessor selection. Second, when evaluating the job predecessor selection criterion for the successor selection, the utility compares the system job name of the current statistic for the successor selection with the entries in this list. If a match is found, the job predecessor selection criterion has been satisfied. Once this criterion has been satisfied, the utility begins evaluating the next criterion defined for the selection. Satisfying the job predecessor criterion also means that the report ultimately generated for successor statistics is able to include the elapsed time between the logging of a statistic associated with a predecessor selection and its matching successor statistic.

Task Predecessor

The process by which the task predecessor selection criterion is handled is twofold and takes place whenever the input log is scanned. First, the ANALYZE_BINARY_LOG utility builds and maintains a list of global task IDs (as well as date/time values) associated with every emitted statistic selected by a predecessor selection. Second, when evaluating the task predecessor selection criterion for the successor selection, the utility compares the global task ID of the current statistic for the successor selection with the entries in this list. If a match is found, the task predecessor selection criterion has been satisfied. Once this criterion has been satisfied, the utility begins evaluating the next criterion defined for the selection. Satisfying the task predecessor criterion also means that the report ultimately generated for successor statistics is able to include the elapsed time between the logging of a statistic associated with a predecessor selection and its matching successor statistic.

Joining Predecessors

The ANALYZE_BINARY_LOG utility can join fields from several statistic selections in a chain of job predecessors or task predecessors for a field report. The utility ensures that the field data in each line of the report defined by one or more PUT_FIELD subcommands comes from the same job or task. For an example of how this capability works, see Examples in the PUT_FIELD subcommand description.

Output

The ANALYZE_BINARY_LOG utility generates two types of output. The first type of output is a simple, formatted report intended to be read by users. This report is produced using the GENERATE_REPORT subcommand. The type of output that results from a GENERATE_REPORT subcommand is determined by a list of subcommands beginning with the word PUT (for example, PUT_FIELD_SUMMARY). These are known as output definition subcommands.

The second type of output is a log that can be used by a program for further analysis. This log is generated by the GENERATE_LOG subcommand. The GENERATE_LOG subcommand can produce the following types of output:

- Binary log. This is a compact format.
- Legible file. This format contains statistics on a single line and is particularly useful as input to programs.
- List file. This format has a maximum display width of 255 columns. A list file format is useful not only as input to programs, but it also makes it easier for you to visually inspect statistics.

When using the ANALYZE_BINARY_LOG utility, specify the input log, the information to be extracted from it, the format of the report, and the output.

Table 9-1 summarizes various kinds of information that you can manipulate as you prepare a report or output log. This information includes both the contents of the output and its formatting specifications. The utility subcommands specify these items either as parameters or as fields within parameters.

Table 9-1. Output Specifications

Category	Description
Counter	Number of the counter whose values appear in an output log.
Counter format	Format in which counter values appear in an output log.
Descriptive data	String or record identifying the descriptive data field that appears in the report. You can specify the starting position of the descriptive data field, its length, or a subfield within the descriptive data field. This category applies only when you want to generate output logs.
Descriptive data length	Number of characters from a statistic's descriptive data field printed in an output log.
Display format	Manner in which the output is displayed. An example is program-readable format.
Display headers	Column headers for the report.
Display option	Manner in which numeric data (such as counter values) is displayed in the report. Examples are MEAN and COUNT_PER_SECOND. You can control the placement of display option information in the report by specifying starting columns and column width information.
Field	Report field. Each field has a name and can contain counter values, a descriptive data field, a statistic header, an elapsed time field, an elapsed time calculation field, or a string. You can specify the placement, in the report, of the field name and the contents of the field by specifying starting columns and column widths. If the field contains counter values, display option information accompanies it.
Input file	Name of the file containing the binary log from which the log or report is generated.
Number	Placement of a specified output definition subcommand (a subcommand starting with PUT) in the list of all such subcommands for a report or log.
Number of counters	Number of counters printed in an output log.
Output file	Name of the file to which the log or report is written.
Put	Name of a report entry contained in an output definition subcommand.
Row label format	Format of the labels that identify each row of a report. Row labels can be the same as the field names, or they can be defined separately.
String	One or more strings to be inserted into a report. You can specify where you want the string to appear in the report.

General Steps in Using the ANALYZE_BINARY_LOG Utility

The following is the general procedure for using the ANALYZE_BINARY_LOG utility. All subcommands referred to are described in detail later in this chapter.

1. Perform this step only if you are analyzing periodic statistics: at the start of system operations, call the MANAGE_PERIODIC_STATISTICS utility to define the time interval at which you want the statistics emitted.
2. Activate the statistics to one or more binary logs. Depending on the statistic, issue an ACTIVATE_SYSTEM_STATISTIC or ACTIVATE_JOB_STATISTIC command. After the job or jobs being measured have run, deactivate the statistics. Keep in mind that some statistics are automatically activated.
3. Either read the active log or obtain a copy of the log you want to analyze. Use the COPY_FILE command to copy a local log. For example, use this command to copy the statistic log for an active job ($LOCAL.$JOB_STATISTIC_LOG). Use the TERMINATE_LOG command to copy a global log.
4. Call the ANALYZE_BINARY_LOG utility.
5. If you don't know which statistics are in the log, display them using the DISPLAY_LOGGED_STATISTICS subcommand.
6. Select the statistic or statistics that you want to analyze. Use the ADD_SELECTION subcommand for this purpose.
7. Specify the counter(s) or data description field(s) for which you want data analyzed. Use the ADD_FIELD subcommand for this purpose.
8. Specify the format and specific contents of the report you want to produce. To do this, use one or more of the subcommands starting with a PUT verb.
9. Produce the report or log. The report or log is written to the output file you specify. Use the GENERATE_REPORT or GENERATE_LOG subcommand for this purpose.
10. Edit the definitions of selections, fields, and report entries, as needed. Use subcommands beginning with an ADD, CHANGE, DELETE, or DISPLAY verb.
11. Produce the revised report or log. Return to the preceding step as needed.
12. Exit the utility. Use the QUIT subcommand for this purpose.

You can specify the ANALYZE_BINARY_LOG command and its subcommands interactively, in a procedure, or by issuing an INCLUDE_FILE command. For routine periodic analysis, the procedure is the easiest method. For situational analysis, entering the subcommands interactively provides the most flexibility.

Introductory Example

The following example illustrates how the ANALYZE_BINARY_LOG utility works. Since a periodic statistic is used, the ANALYZE_BINARY_LOG utility session is preceded by a MANAGE_PERIODIC_STATISTICS utility session. Together, these utilities provide the capability to collect and report on the data emitted by occurrences of the statistic.

The example makes use of the OS6 statistic, which reports CPU usage. This statistic is useful for tracking CPU use over a particular time interval. For example, a timesharing workload may make little use of the CPU before 8 a.m. and after 5 p.m.; therefore, CPU idle time may be very high during this time. Peak loads may occur between 10 a.m. and 2 p.m., when relatively little CPU idle time is reported.

The first two counters of the OS6 statistic report the amount of time the CPU was idle. Counter 1 reports the number of microseconds the CPU was idle and no I/O was taking place. Counter 2 reports the number of microseconds the CPU was idle with I/O active.

```
manage_periodic_statistics
  add_statistic emission_set=s1 statistic=os6
  change_period emission_set=s1 period=5
  change_state emission_set=s1 state=enabled
quit write_emission_sets=true

activate_system_statistic statistic=os6
.
.  (System runs)
.
deactivate_system_statistic statistic=os6

terminate_log type=statistic file=$user.system_stat_log

analyze_binary_log
  add_selection selection=cpu_usage statistic_code=os6 time=10:00:00..11:00:00
  add_field field=cpu_idle_without_io_active selection=cpu_usage ..
    counter=(1, 0.000001, true, false)
  add_field field=cpu_idle_with_io_active selection=cpu_usage ..
    counter=(2, 0.000001, true, false)
  put_string string=(('CPU Idle Time Report' 12))
  put_string string=' '
  put_interval_field field=((cpu_idle_without_io_active,,,15, ..
    'CPU Idle ' , 'Without I/O') (cpu_idle_with_io_active,,,15, ..
    'CPU Idle' , 'With I/O')) row_label=(end_time, 1, 12, 'MS')
  put_string string=' '
  add_field field=mean_string selection=cpu_usage string= 'Mean'
  put_field field=((mean_string,,1,12) (cpu_idle_without_io_active mean,,15) ..
    (cpu_idle_with_io_active mean,,15))
  generate_report input=$user.system_stat_log o=$user.my_report
  quit
```

The following report is produced:

```
           CPU Idle Time Report

                  CPU Idle         CPU Idle
                 Without I/O       With I/O
10:07:34.938         273.261         10.978
10:12:35.216         244.654         20.743
10:17:34.912         189.588         35.448
10:22:34.743         144.515         64.521
10:27:35.905          75.866         80.292
10:32:34.859         159.137         55.394
10:37:35.070         176.600         81.263
10:42:35.251         257.172         13.370
10:47:35.467         254.040         23.644
10:52:34.870         230.665         16.715
10:57:35.114         257.053         16.125

Mean                 205.686         38.045
```

The example reflects the following activity:

1. The MANAGE_PERIODIC_STATISTICS utility is called from a System Operator Utility session at the start of system operations. The MANAGE_PERIODIC_STATISTICS utility initiates emission of the OS6 statistic at 5-minute intervals. The OS6 statistic is assigned to emission set S1.

2. The statistic is activated with the ACTIVATE_SYSTEM_STATISTIC command. This step tells the system to begin recording the statistic to a binary log (the global statistic log) each time it is emitted.

3. The job is run.

4. The statistic is deactivated with the DEACTIVATE_SYSTEM_STATISTIC command.

5. At the end of system operations, the global statistic log is written to permanent file $USER.SYSTEM_STAT_LOG. This file is used as input to the ANALYZE_BINARY_LOG utility.

6. An ANALYZE_BINARY_LOG utility session begins. The utility defines the criteria for the statistic data that will be reported, defines the type of data that the report will contain, and, finally, generates the report itself. This activity is described in the remaining steps.

7. The statistical information to be reported is identified in two steps:

   a. An ADD_SELECTION subcommand specifies the name of the statistic to be reported (OS6) and chooses criteria for evaluating which statistics contained in the binary log are selected for reporting (TIME parameter). This subcommand also assigns a name to the selection criteria for the OS6 statistic. The selection name CPU_USAGE refers to all occurrences of the statistic code OS6 that were emitted between 10 a.m. and 11 a.m. Statistics that are emitted outside of this time range are ignored.

   b. An ADD_FIELD subcommand identifies the counter values for the selected statistic. Although OS6 contains six counters, the example only reports the information contained in the first two counters. The field name CPU_IDLE_WITHOUT_IO_ACTIVE refers to counter 1 of the OS6 statistic. The field name CPU_IDLE_WITH_IO_ACTIVE refers to counter 2 of the statistic. Since the data in these counters is expressed in microseconds, a multiplier is supplied to convert the reported data to seconds. Both counter 1 and counter 2 of the OS6 statistic are incremental and negative values are not allowed. The data reported represents interval counter values rather than cumulative values.

8. The format of the report is defined. Subcommands starting with the verb PUT are used for this purpose.

   a. A PUT_STRING subcommand adds a title to the report. The title *CPU Idle Time Report* is placed in column 12 and is followed by a blank line.

   b. A PUT_INTERVAL_FIELD subcommand defines the data that will be collected for each occurrence of the selected statistic. Data from two fields is formatted: CPU_IDLE_WITH_IO_ACTIVE and CPU_IDLE_WITHOUT_IO_ACTIVE. The width of each column is set to 15, and the column headers are identified. The ROW_LABEL parameter specifies the label to be used for each row of data. Since the data for the OS6 statistic reflects the value calculated at the end of a time interval, the label chosen displays the time at the end of the interval being reported. The label is displayed starting at column 1 having a width of 12 columns and is displayed in the format hh:mm:ss.mmm.

   c. An ADD_FIELD subcommand defines the MEAN_STRING field. When the field MEAN_STRING is subsequently referenced, the string 'Mean' is displayed.

   d. A PUT_FIELD subcommand defines a single line of data to be displayed. The line is the last line of the report and contains three values. The first is the value of the MEAN_STRING field and is displayed starting at column 1, which has a width of 12 columns. The second value is the mean value of the CPU_IDLE_WITHOUT_IO_ACTIVE field. The third value is the mean value of the CPU_IDLE_WITH_IO_ACTIVE field. These last two fields are each 15 columns wide.

9. A GENERATE_REPORT subcommand specifies the name of the file containing the binary log and the name of the file to which the report is to be written. This subcommand causes the binary log to be read and the report to be produced. By default, the report is produced in user-readable format (LIST format).

10. A QUIT subcommand is issued to exit the utility.

Other examples are provided in the ANALYZE_BINARY_LOG utility subcommand descriptions and in Examples of Using Statistics later in this chapter.

Summary of Subcommands

The following command initiates the ANALYZE_BINARY_LOG utility:

ANALYZE_BINARY_LOG

Table 9-2 summarizes the ANALYZE_BINARY_LOG utility subcommands:

Table 9-2. ANALYZE_BINARY_LOG Utility Subcommands

Subcommand	Description
ADD_FIELD	Defines a counter, a descriptive data field, an elapsed time field, and elapsed time calculation field, a statistic header, or a string associated with a statistic named by a selection.
ADD_SELECTION	Defines criteria for selecting statistics from the input log.
CHANGE_DEFAULTS	Changes defaults for the ANALYZE_BINARY_LOG utility.
CHANGE_FIELD	Changes attribute values of an existing field.
CHANGE_PUT	Modifies an entry contained in an output definition subcommand.
CHANGE_SELECTION	Changes selection criteria of an existing selection.
DELETE_FIELD	Deletes one or more field definitions established by an ADD_FIELD subcommand.
DELETE_PUT	Deletes one or more report entries specified by an output definition subcommand.
DELETE_SELECTION	Deletes one or more selections defined by an ADD_ SELECTION subcommand.
DISPLAY_DEFAULTS	Displays the current defaults for the ANALYZE_ BINARY_LOG utility.
DISPLAY_FIELD	Displays one or more field definitions.
DISPLAY_LOGGED_ STATISTICS	Displays the names, counts, and time ranges for statistics found in a binary log.
DISPLAY_PUT	Displays the report entries or log entries specified by the output specification subcommands during the current utility session.
DISPLAY_SELECTION	Displays the selection definitions for the current utility session.
GENERATE_LOG	Produces an output log that is based on specifications established by the output definition subcommands.

(Continued)

Table 9-2. ANALYZE_BINARY_LOG Utility Subcommands *(Continued)*

Subcommand	Description
GENERATE_REPORT	Produces a report based on specifications established by the output definition subcommands for the current utility session.
POP_PAGE_HEADER	Deletes a line from the page header.
PUSH_PAGE_HEADER	Adds a line to the page header.
PUT_FIELD	Defines a single report line consisting of one or more fields and their respective display options.
PUT_FIELD_SUMMARY	Defines field summary rows that will appear in a report.
PUT_INTERVAL_FIELD	Defines report entries based on occurrences or groups of occurrences of a statistic.
PUT_NEW_PAGE	Specifies a page eject in the report.
PUT_RECORD	Defines one or more selections whose designated statistics are to be listed in the log produced by a GENERATE_LOG subcommand.
PUT_STRING	Defines a line of text fields to be inserted into a report.
QUIT	Terminates the utility.

Table 9-3 shows the structure of the editing subcommands used within the ANALYZE_BINARY_LOG utility. This structure reflects the interplay of selections, fields, and puts.

Table 9-3. ANALYZE_BINARY_LOG Subcommand Matrix

Operation	Selection	Field	Put
ADD	ADD_SELECTION	ADD_FIELD	PUT_FIELD_SUMMARY PUT_INTERVAL_FIELD PUT_FIELD PUT_RECORD PUT_STRING
CHANGE	CHANGE_SELECTION	CHANGE_FIELD	CHANGE_PUT
DELETE	DELETE_SELECTION	DELETE_FIELD	DELETE_PUT
DISPLAY	DISPLAY_SELECTION	DISPLAY_FIELD	DISPLAY_PUT

NOTE

When defining selections, fields, and puts, do not choose the names ALL or A.

ANALYZE_BINARY_LOG Command

Purpose Calls the ANALYZE_BINARY_LOG utility and prompts you for subcommands.

Format **ANALYZE_BINARY_LOG** or
ANABL

Parameters *STATUS*

Returns the completion status for this subcommand.

Remarks The ANALYZE_BINARY_LOG utility prompts for the entry of subcommands with the following string:

```
ABL/
```

Examples Refer to Examples of Using Statistics later in this chapter.

ADD_FIELD Subcommand

Purpose Defines a counter, a descriptive data field, an elapsed time field, an elapsed time calculation field, a statistic header, or a string associated with a statistic named by a selection.

Format **ADD_FIELD** or
ADDF
FIELD=name
SELECTION=name
COUNTER=record
DESCRIPTIVE_DATA=record
ELAPSED_TIME=keyword
ELAPSED_TIME_CALCULATION=record
HEADER=keyword
STRING=string
STATUS=status variable

Parameters **FIELD** or **F**

Defines the name of the statistic field to be added. The new field must not have been previously defined. The field can be a counter, a descriptive data field, an elapsed time field, an elapsed time calculation field, a statistic header, or a string. This parameter is required.

SELECTION or **S**

Defines the name of the selection naming the statistic from which the field is to be taken. The selection must have been previously defined. This parameter is required.

COUNTER or *C*

Defines which counter of the statistic named by the selection will be used for the field being added. By default, no counter is used for the value of the field. The counter field is defined by a record having the following format:

(*counter_number, multiplier, incremental, allow_negative_increment*)

counter_number

Specifies the counter number. You can specify an integer from 1 to 4095. If you specify a counter number that is greater than the number of counters in the statistic, no data will be collected from that selection.

multiplier

Specifies a real number used as a multiplier for all values of the field specified by the FIELD parameter. To specify a divisor, you can use a decimal fraction (such as 0.001) or you can use an expression of real numbers (such as 1.0/1000.0). The default is 1.0.

incremental

Specifies a boolean value indicating how field values are to be calculated. The default is FALSE.

TRUE

The value of the field is the result of subtracting the values of consecutive occurrences of the counter.

FALSE

The value of the field remains the same as found in the counter.

allow_negative_increment

Specifies a boolean value indicating whether to allow a negative value for a counter whose incremental entry is TRUE. The default is FALSE.

TRUE

Negative values are allowed for an incremental counter.

FALSE

Negative values are not allowed for an incremental counter. The underlying assumption is that the value is negative because a deadstart occurred that reset the value of the counter. In this case, the utility uses the current nonincremental value of the counter.

DESCRIPTIVE_DATA or *DD*

Defines either a descriptive data field or a substring of the descriptive data field to be used for the field being added. The descriptive data field or substring is defined by a record having the following format:

(position, length, field_number, field_delimiter)

position

Specifies the starting position of the descriptive data field or substring. You can specify an integer from 1 to 4,095. The default is 1.

length

Specifies the length of the substring. You can specify an integer from 1 to 4,095. You can also specify the keyword ALL. The default is ALL.

field_number

Specifies the field number for the substring. You can specify an integer from 1 to 4,095. You can also specify the keyword ALL. The default is ALL.

field_delimiter

Specifies a string containing the character that delimits the subfields of a descriptive data field. The default is a comma (,).

ELAPSED_TIME or *ET*

Specifies a keyword indicating how to calculate the elapsed time between statistics belonging to this field's selection. You can specify one of the following keywords:

PREDECESSOR or P

Elapsed time between occurrences of a statistic and its job or task predecessor statistic.

PREDECESSOR_CHAIN_HEAD or PCH

Elapsed time between occurrences of a statistic and its job or task predecessor statistic at the head of the predecessor chain.

PREVIOUS_OCCURRENCE or PO

Elapsed time between successive occurrences of statistics belonging to this field's selection.

ELAPSED_TIME_CALCULATION or *ETC*

Defines a time increment calculation field. This field is defined by a record having the following format:

(calculation, elapsed_time, counter_number, multiplier, incremental, allow_negative_increment)

calculation

Specifies a keyword indicating which calculation method you want to choose. You can specify one of the following keywords:

VALUE_PER_SECOND or VPS

Value calculated by dividing a specified counter by elapsed time.

OCCURRENCE_PER_SECOND or OPS

Value calculated by dividing 1 by elapsed time. If you choose this calculation method, you cannot specify the counter entries.

elapsed_time

The elapsed time value is specified using the following keywords: PREDECESSOR, PREDECESSOR_CHAIN_HEAD, or PREVIOUS_OCCURRENCE as described for the ELAPSED_TIME parameter.

counter_number, multiplier, incremental, allow_negative_increment

These entries specify the counter value. They are described in the COUNTER parameter description. Specify these entries only if you choose the VALUE_PER_SECOND keyword for the calculation method.

HEADER or *H*

Specifies a keyword indicating which one of the statistic headers you want to choose for the field being added. You can specify one of the following keywords:

DATE_TIME or DT
DESCRIPTIVE_DATA_LENGTH or DDL
GLOBAL_TASK_ID or GTI
NUMBER_OF_COUNTERS or NOC
STATISTIC_CODE or SC
SYSTEM_JOB_NAME or SJN

STRING or *STR*

Defines a string of 0 to 4095 characters that you want to appear as output with the named selection.

STATUS

Returns the completion status for this subcommand.

Remarks

- The named field can be referenced in the PUT_FIELD_SUMMARY, PUT_INTERVAL_FIELD, and PUT_FIELD report specification subcommands.
- Only one of the following parameters can be defined for this subcommand: COUNTER, DESCRIPTIVE_DATA, ELAPSED_TIME, ELAPSED_TIME_CALCULATION, HEADER, or STRING.

Examples

- The following example creates the JOB_END and JOB_QUEUING_STARTED selections. Two fields are created for each selection. The CP_TIME_IN_JOB_MODE field is the first counter of the JM3 statistic, and a multiplier converts the value of the field from microseconds to seconds. The PAGE_FAULTS field is the third counter of the JM3 statistic. The SYSTEM_JOB_NAME field is that portion of the descriptive data field of the JM4 statistic starting at position 1 and having a length of 19 characters. The CONTROL_USER field is that portion of the descriptive data field of the JM4 statistic starting at position 144 and having a length of 31 characters.

```
add_selection selection=job_end statistic_code=jm3
add_field field=cp_time_in_job_mode selection=job_end ..
  counter=(1, 0.000001)
add_field field=page_faults selection=job_end counter=3

add_selection selection=job_queuing_started statistic_code=jm4
add_field field=system_job_name selection=job_queuing_started ..
  descriptive_data=(1,19)
add_field field=control_user selection=job_queuing_started ..
  descriptive_data=(144,31)
```

- The following example, creates the DD field. If you specify descriptive_data=() (empty parentheses), the full descriptive data field applies.

```
add_field field=dd selection=task descriptive_data=()
```

- The following example adds an elapsed time calculation field named ET_CALC to selection TASK. The chosen calculation is value_per_second. The elapsed time is computed from the predecessor. The value of the field is taken as the value of the second counter of the selection TASK.

```
add_field field=et_calc selection=task ..
  elapsed_time_calculation=(value_per_second, predecessor, 2)
```

ADD_SELECTION Subcommand

Purpose Defines criteria for selecting statistics from the input log.

Format **ADD_SELECTION** or
ADDS
SELECTION=name
STATISTIC_CODE=statistic_code
TIME=range of date_time
CONTINUOUS_DATE_TIME=boolean
DESCRIPTIVE_DATA=list of record
JOB_PREDECESSOR=name
TASK_PREDECESSOR=name
SYSTEM_JOB_NAME=name
GLOBAL_TASK_ID=string
STATUS=status variable

Parameters **SELECTION or S**

Defines the name of the selection to be created. The new selection must not have been previously defined. This parameter is required.

STATISTIC_CODE or *SC*

Defines a valid statistic code. The statistic code must consist of two characters followed by a 1- to 5-digit number. Statistic code numbers from 0 to 9,999 are reserved for Control Data. Statistic code numbers from 10,000 to 19,000 are reserved for site codes and user-defined codes.

TIME or *T*

Defines a date, time, or date/time range. To be included in a selection, statistics must have been emitted within the range specified. The date_time data type is a record having the following format:

```
(year-month-day.hour:minute:second.millisecond)
```

You can specify the date and time components alone or together. You do not need to specify milliseconds. Examples are as follows:

```
time=1989-11-21.08:00:00 .. 1989-11-21.17:00:00

time=08:00:00 .. 17:00:00
```

If you specify only the date portion of this parameter, the utility completes the time portion for you. Similarly, if you specify only the time portion, the utility completes the date portion for you. The utility completes the date/time value according to its lowest and highest time and date ranges. For example, specifying TIME=1989-10-28 causes the utility to interpret the value of this parameter as follows:

```
1989-10-28.00:00:00.000 .. 1989-10-28.23:59:59.999
```

Specifying TIME=8:00 .. 16.00 and CONTINUOUS_DATE_TIME=FALSE, causes the utility to interpret the value of this parameter as follows:

```
1900-01-01.08:00:00.000 .. 2155-12-31.16:00:00.000
```

The date_time value can also be expressed as a date/time string in a valid date/time format. For a list of date/time formats, see the NOS/VE System Usage manual.

By default, the date and time are not selection criteria. In this case, if all other selection criteria match, a statistic is selected.

CONTINUOUS_DATE_TIME or *CDT*

Specifies a boolean value that modifies the interpretation of the TIME parameter. The CONTINUOUS_DATE_TIME parameter only affects how the values of the TIME parameter are interpreted when both the date and the time are specified as a range. The default is FALSE.

TRUE

Specifies that the date and time ranges will be considered together. This means that the first time in the time range only applies to the first date in the date range, the second time in the time range only applies to the second date in the date range, and so on.

FALSE

Specifies that the time range will be treated independently of the date range. This means that, for every date in the date range, selections are chosen according to the time range for that day.

DESCRIPTIVE_DATA or *DD*

Defines a descriptive data field or a substring or a list of substrings of the descriptive data field that must match the descriptive data field for the statistic you want to include in a selection.

The descriptive data field is defined by a record having the following format:

(string, position, length, field_number, field_delimiter)

string

Defines a descriptive data field expressed as a string of 0 to 4,095 characters. You can also specify a list of strings.

position

Defines the starting position of the descriptive data field or substring. You can specify an integer from 0 to 4,095. The default is 1.

length

Defines the length of the substring. You can specify an integer from 0 to 4,095. You can also specify the keyword ALL. The default is ALL.

field_number

Defines the field number for the substring. You can specify an integer from 1 to 4,095. You can also specify the keyword ALL. The default is ALL.

field_delimiter

Defines a string containing the character that delimits the subfields of a descriptive data field. The default is a comma (,).

JOB_PREDECESSOR or *JP*

Specifies the name of a selection for which a statistic emitted by a particular job was previously logged. This selection must have been previously defined. A selection can have only one job predecessor selection; it may have many successors. If you specify this parameter, you should not specify the TASK_PREDECESSOR parameter. For more information about this selection criterion, see the Selection Criteria subsection.

TASK_PREDECESSOR or *TP*

Specifies the name of a selection for which a statistic emitted by a particular task was previously logged. This selection must have been previously defined. A selection can have only one task predecessor selection; it may have many successors. If you specify this parameter, you should not specify the JOB_PREDECESSOR parameter. For more information about this selection criterion, see the Selection Criteria subsection earlier in this chapter.

SYSTEM_JOB_NAME or *SJN*

Specifies the system-supplied name of the job emitting the statistic you want selected. You must specify the full 19-character name; abbreviations are not valid. The statistic must have been emitted by the named job to be included under the selection currently being defined or changed.

GLOBAL_TASK_ID or *GTI*

Specifies a string indicating the global task identifer of the task identified in the statistic you want selected. The statistic must correspond to the named task to be included under the selection currently being defined or changed.

The global task identifier specified by this parameter has the following format:

```
'nnnnn-nnn'
```

The first five numbers (nnnnn) must be in the range from 0 to 65,535. The last three numbers (nnn) must be in the range from 0 to 255.

STATUS

Returns the completion status for this subcommand.

Remarks

- If the CONTINUOUS_DATE_TIME parameter is set to TRUE, a statistic residing in the input log satisfies the emission time selection criterion only if the statistic occurs after the first occurrence of the time and date combination, but before the second occurrence.

- Avoid using recursive job predecessors and task predecessors. If you do use them, employ one of the following methods to delete a selection from a recursive predecessor chain:

  - Enter the DELETE_SELECTION subcommand with the SELECTION parameter set to ALL.

  - Break the recursive nature of the predecessor by changing the name of the selection in question using the CHANGE_SELECTION subcommand, and then deleting the original selection using the DELETE_SELECTION subcommand.

Examples

- The following example defines a selection called DISK_UNIT_USE. This selection includes all occurrences of the OS4 statistic.

```
add_selection selection=disk_unit_use statistic_code=os4
```

- In the following example, the selection TASK_BEGIN is defined to include only those PM0 statistics that were placed in the input log between 8 a.m. and 4 p.m. every day that the statistic was logged:

```
add_selection selection=task_begin statistic_code=pm0 ..
  time=08:00:00..16:00:00
```

- The following example shows how the TASK_PREDECESSOR parameter is used. The statistics that will belong to the TASK_END selection are only those PM3 statistics that were logged by a task that previously logged the PM0 statistic between 8 a.m. and 4 p.m. (that is, those PM0 statistics that were selected by the TASK_BEGIN selection).

```
add_selection selection=task_begin statistic_code=pm0 ..
  time=08:00:00..16:00:00
add_selection selection=task_end ..
  statistic_code=pm3 task_predecessor=task_begin ..
  descriptive_data='FCP$COMPILE_FORTRAN_SOURCE'
```

- The following example shows how to restrict a string match to a delimited subfield of the descriptive data field rather than to any substring. In this example, any PM3 statistic having subfields delimited by semicolons ';' and having the string 'file_1' in its second subfield belongs to selection example_4.

```
add_selection selection=example_4 statistic_code=pm3 ..
  descriptive_data=(('file_1',,, 2, ';'))
```

- Descriptive data selection can be based on a list of records. In order to belong to such a selection, the specified statistic must match all of the specified selection criteria. That is, the utility performs a logical AND of the different records that are specified.

  In the following example, a PM3 statistic belongs to selection example_5 only if it meets both selection criteria. Its first subfield, which is delimited by a semicolon, must contain the string 'procedure_1' AND its second subfield delimited by a semicolon must contain the string 'file_1'.

  ```
  add_selection selection=example_5 statistic_code=pm3 ..
    descriptive_data=(('procedure_1',,, 1, ';') ..
    ('file_1',,, 2, ';'))
  ```

- The selection criteria for the descriptive data can become very complex when using combinations of the various capabilities. In the following example, a PM3 statistic belongs to selection example_6 if: the first 7 characters of the subfield delimited by a semicolon match the string 'procedu' OR 'program' AND the first 6 characters of the second subfield delimited by a semicolon match the string 'file_1' OR 'file_2'.

  ```
  add_selection selection=example_6 statistic_code=pm3 ..
    descriptive_data=((('procedu' 'program'), 1, 7, 1, ';') ..
    (('file_1' 'file_2'),1, 6, 2, ';'))
  ```

CHANGE_DEFAULT Subcommand

Purpose Changes the current defaults for the ANALYZE_BINARY_LOG utility.

Format **CHANGE_DEFAULT** or
CHANGE_DEFAULTS or
CHAD
COUNTER_FRACTION=keyword or *integer*
DATE_TIME_FORMAT=keyword or *string*
INPUT_LOG=keyword or *list of file*
LEGIBLE_DATA_MAX_PAGE_WIDTH=keyword or *integer*
LIST_MAX_PAGE_WIDTH=keyword or *integer*
TIME_INCREMENT_FORMAT=keyword
STATUS=status variable

Parameters *COUNTER_FRACTION* or *CF*

Defines the default counter fraction size. This parameter sets the number of fractional digits to include in fixed point format. You can specify an integer from 0 to 9, or you can specify the keyword DEFAULT. Specifying DEFAULT causes the ANALYZE_BINARY_LOG utility to use a value of 3. The default is DEFAULT.

DATE_TIME_FORMAT or *DTF*

Defines the default date_time format. You can specify the date_time value expressed as a date/time string or you can specify the keyword DEFAULT. For a list of valid date/time formats, see the NOS/VE System Usage manual. Specifying DEFAULT causes the ANALYZE_BINARY_LOG utility to use the ISOD MILLISECOND format as follows:

```
yyyy-mm-dd.hh:mm:ss.mmmm
```

The default is DEFAULT.

INPUT_LOG or *LOG* or *L*

Defines a list of names of the binary logs to be read. This list of binary logs can be used as the default input for the GENERATE_LOG, GENERATE_REPORT, and DISPLAY_LOGGED_STATISTICS subcommands. You can also specify the keyword UNSPECIFIED, which means that no default list of binary logs exists. The default is UNSPECIFIED.

LEGIBLE_DATA_MAX_PAGE_WIDTH or *LDMPW*

Defines the default maximum page width for the LEGIBLE_DATA format of the DISPLAY_FORMAT parameter of the GENERATE_LOG and GENERATE_REPORT subcommands. You can specify an integer from 1 to 10,000, or you can specify the keyword DEFAULT. Specifying DEFAULT causes the ANALYZE_BINARY_LOG utility to use a value of 10,000 characters. The default is DEFAULT.

LIST_MAX_PAGE_WIDTH or *LMPW*

Defines the default maximum page width for the LIST format of the DISPLAY_FORMAT parameter of the GENERATE_LOG and GENERATE_REPORT subcommands. You can specify an integer from 1 to 10,000, or you can specify the keyword DEFAULT. Specifying DEFAULT causes the ANALYZE_BINARY_LOG utility to use a value of 132 characters. The default is DEFAULT.

TIME_INCREMENT_FORMAT or *TIF*

Defines the defualt time increment format to be used in the ANALYZE_BINARY_LOG utility. You can specify one of the following keywords:

SECONDS or S

Specifies seconds in the following format:

```
nnnn.nnn
```

TIME_INCREMENT or TI

Specifies a time increment in the following format:

```
yy-mm-dd.hh:mm:ss.mmm
```

DEFAULT

Same as specifying the keyword SECONDS.

STATUS

Returns the completion status for this subcommand.

CHANGE_FIELD Subcommand

Purpose Changes attribute values of an existing field.

Format **CHANGE_FIELD** or
CHAF
  FIELD = name
  NEW_FIELD = name
  SELECTION = name
  COUNTER = record
  DESCRIPTIVE_DATA = record
  ELAPSED_TIME = keyword
  ELAPSED_TIME_CALCULATION = record
  HEADER = keyword
  STRING = string
  STATUS = status variable

Parameters **FIELD** or **F**

Specifies the name of the statistic field to be changed. The field must already have been defined. The field can be a counter, a descriptive data field, an elapsed time field, an elapsed time calculation field, a statistic header, or a string. This parameter is required.

NEW_FIELD or *NF*

Defines the new name for the field. The field must not have been previously defined.

SELECTION or *S*

Defines the name of the new selection naming the statistic from which the field is to be taken.

COUNTER or *C*

Defines which counter of the statistic named by the selection will be used for the field being added. By default, no counter is used for the value of the field. The counter field is defined by a record having the following format:

(*counter_number, multiplier, incremental, allow_negative_increment)*

counter_number

Specifies the counter number. You can specify an integer from 1 to 4,095. If you specify a counter number that is greater than the number of counters in the statistic, no data will be collected from that selection.

multiplier

Specifies a real number used as a multiplier for all values of the field specified by the FIELD parameter. To specify a divisor, you can use a decimal fraction (such as 0.001) or you can use an expression of real numbers (such as 1.0/1000.0). The default is 1.0.

incremental

Specifies a boolean value indicating how field values are to be calculated. The default is FALSE.

TRUE

The value of the field is the result of subtracting the values of consecutive occurrences of the counter.

FALSE

The value of the field remains the same as found in the counter.

allow_negative_increment

Specifies a boolean value indicating whether to allow a negative value for a counter whose incremental entry is TRUE. The default is FALSE.

TRUE

Negative values are allowed for an incremental counter.

FALSE

Negative values are not allowed for an incremental counter. The underlying assumption is that the value is negative because a deadstart occurred that reset the value of the counter. In this case, the utility uses the current nonincremental value of the counter.

DESCRIPTIVE_DATA or *DD*

Defines either a descriptive data field or a substring of the descriptive data field to be used for the field being added. The descriptive data field or substring is defined by a record having the following format:,

(*position, length, field_number, field_delimiter*)

position

Defines the starting position of the descriptive data field or substring. You can specify an integer from 1 to 4,095. The default is 1.

length

Defines the length of the substring. You can specify an integer from 1 to 4,095. You can also specify the keyword ALL. The default is ALL.

field_number

Defines the field number for the substring. You can specify an integer from 1 to 4,095. You can also specify the keyword ALL. The default is ALL.

field_delimiter

Defines a string containing the character that delimits the subfields of a descriptive data field. The default is a comma (,).

ELAPSED_TIME or *ET*

Specifies a keyword indicating how to calculate the elapsed time between statistics belonging to this field's selection. You can specify one of the following keywords:

PREDECESSOR or P

Elapsed time between occurrences of a statistic and its job or task predecessor statistic.

PREDECESSOR_CHAIN_HEAD or PCH

Elapsed time between occurrences of a statistic and its job or task predecessor statistic at the head of the predecessor chain.

PREVIOUS_OCCURRENCE or PO

Elapsed time between successive occurrences of statistics belonging to this field's selection.

ELAPSED_TIME_CALCULATION or *ETC*

Defines a time increment calculation field. This field is defined by a record having the following format:

(calculation, elapsed_time, counter_number, multiplier, incremental, allow_negative_increment)

calculation

Specifies a keyword indicating which calculation method you want to choose. You can specify one of the following keywords:

VALUE_PER_SECOND or VPS

Value calculated by dividing a specified counter by elapsed time.

OCCURRENCE_PER_SECOND or OPS

Value calculated by dividing 1 by elapsed time. If you choose this calculation method, you cannot specify the counter entries.

elapsed_time

The elapsed time value is specified using one of the following keywords: PREDECESSOR, PREDECESSOR_CHAIN_HEAD, or PREVIOUS_OCCURRENCE as described for the ELAPSED_TIME parameter.

counter_number, multiplier, incremental, allow_negative_increment

These entries specify the counter value. They are described in the COUNTER parameter description. Specify these entries only if you choose the VALUE_PER_SECOND keyword for the calculation method.

HEADER or *H*

Specifies a keyword indicating which one of the statistic headers to choose for the field being added. You can specify one of the following keywords:

DATE_TIME or DT
DESCRIPTIVE_DATA_LENGTH or DDL
GLOBAL_TASK_ID or GTI
NUMBER_OF_COUNTERS or NOC
STATISTIC_CODE or SC
SYSTEM_JOB_NAME or SJN

STRING or *STR*

Defines a string of 0 to 4,095 characters that are to appear as output with the named selection.

STATUS

Returns the completion status for this subcommand.

Remarks

You can specify only one of the following field types on each CHANGE_FIELD subcommand: COUNTER, DESCRIPTIVE_DATA, ELAPSED_TIME, ELAPSED_TIME_CALCULATION, HEADER, or STRING. If you do not specify any of these field types, the old field type attributes are preserved.

Examples

In the following example, the CONTROL_USER field is created and then changed. The change consists of correcting the descriptive data substring length from 174 to 31; the starting column remains at 144.

```
add_field field=control_user selection=job_queuing_started ..
  descriptive_data=(144,174)
change_field field=control_user descriptive_data=(144,31)
```

CHANGE_PUT Subcommand

Purpose Modifies an entry contained in an output definition subcommand.

Format **CHANGE_PUT** or
CHAP
PUT=name
NUMBER=keyword or *integer*
NEW_PUT=name
NEW_NUMBER=keyword or *integer*
FIELD=list of record
DISPLAY_OPTION=list of record
DISPLAY_HEADERS=boolean
ROW_LABEL=record
ROW_LABEL_FORMAT=record
REPORT_INTERVAL=integer
STRING=list of string
SELECTION=list of name
COUNTER=keyword or *record*
DESCRIPTIVE_DATA=string or *record*
STATUS=status variable

Parameters *PUT* or *P*

Specifies the name of the report entry or log entry. The entry must have been previously defined. You must specify either this parameter or the NUMBER parameter, but you cannot specify both.

NUMBER or *N*

Specifies the current relative position of the entry within the list of output definition subcommands (that is, all subcommands containing the word PUT). You can specify an integer from 1 to $MAX_INTEGER, or you can specify the keyword LAST.

You must specify either this parameter or the PUT parameter, but you cannot specify both.

NEW_PUT or *NP*

Defines the new name of the report entry or log entry. The entry must not have been previously defined.

NEW_NUMBER or *NN*

Defines the new relative position of the entry within the report. You can specify an integer from 1 to $MAX_INTEGER, or you can specify the keyword LAST. This means that the new output definition subcommand is placed after all current output definition subcommands. If you specify an integer, the new subcommand occupies that relative position. If you specify a relative position greater than the largest used ordinal, the subcommand is placed at the end of the list of subcommands.

FIELD or *FIELDS* or *F*

Specifies changed field information that you want reported by a PUT_FIELD, PUT_FIELD_SUMMARY, or PUT_INTERVAL_FIELD subcommand. The field must have been previously defined. The original field entries for the put are replaced by the field entries specified by this parameter. This parameter is represented by one of two record types.

The first record type is an unlimited list and is valid for PUT_FIELD_SUMMARY subcommands. This record has the following format:

(*field_name, row_label*)

field_name

Specifies the name of the field to be changed.

row_label

Specifies the name of a row in the report. This subfield must be expressed as a string.

The second record type is restricted to 25 entries and is valid for PUT_FIELD and PUT_INTERVAL_FIELD subcommands. This record type has the following format:

(*field_name, display_option, start_column, column_width, header_1, header_2*)

field_name

Specifies the name of the field to be changed.

display_optoin

Specifies a keyword indicating how you want numeric data in the specified field summarized. You can specify the following keywords:

ALL_OCCURRENCES or AO or DETAIL or D

Value of the field from each occurrence of the field's selection in the log. This results in one line of output for each occurrence of the field's selection in the log.

FIRST_OCCURRENCE or FO

Value of the field from the first occurrence of the field's selection in the log.

LAST_OCCURRENCE or LO

Value of the field from the last occurrence of the field's selection in the log.

COUNT or C

Number of occurrences of the field in the log.

SUM or S

Sum of the values of all occurrences of the field in the log.

MEAN or M or AVERAGE or AVG

Average as computed by dividing the SUM field by the COUNT field.

STANDARD_DEVIATION or SD

Normal standard deviation with a divisor of (n - 1).

MINIMUM or MIN

Minimum value of all occurrences of the field in the log.

MAXIMUM or MAX

Maximum value of all occurrences of the field in the log.

COUNT_PER_SECOND or CPS

Average as computed by dividing the COUNT field by the average time interval between occurrences of a field.

SUM_PER_SECOND or SPS

Average as computed by dividing the SUM field by the average time interval between occurrences of a field.

start_column

Defines the column in which the display_option value begins. You can specify an integer from 1 to 10,000. You must define starting columns in such a way that columns do not overlap. The default for this field is the last position in the previous field plus 2. If there is no row label, the default start column for the first field is 1. If there is a row label, the default start column for the first field is the last position in the row label plus 1.

column_width

Defines the width of the column in which the display_option value begins. You can specify an integer from 1 to 10,000. You must define column widths in such a way that columns do not overlap. The default for this field depends on what kind of display_option is performed:

- If the display_option entry is COUNT, the default is 10.
- If the display_option entry is SUM, MEAN, STANDARD_DEVIATION, MINIMUM, MAXIMUM, COUNT_PER_SECOND, or SUM_PER_SECOND, the default is 15.

header_1

Defines the name of the first column header in the report. This field must be expressed as a string of 1 to 10,000 characters.

header_2

Defines the name of the second column header in the report. This field must be expressed as a string of 1 to 10,000 characters.

DISPLAY_OPTION or *DO*

Defines one or more records that modify the value of the DISPLAY_OPTION parameter defined earlier by a PUT_FIELD_SUMMARY subcommand. The record has the following format:

(**calculation**, *start_column, column_width*)

calculation

Specifies a keyword indicating how you want numeric data to be represented in the report. This entry is required. For a list of the keywords, see the display_option description under the FIELD parameter description.

start_column

Defines the column in which the calculation value begins. You can specify an integer from 1 to 10,000. You must define starting columns in such a way that columns do not overlap. The default for this field is the last position in the previous field plus 2. If there is no row label, the default start column for the first field is 1. If there is a row label, the default start column for the first field is the last position in the row label plus 1.

column_width

Defines the width of the column in which the calculation value begins. You can specify an integer from 1 to 10,000. You must define column widths in such a way that columns do not overlap. The default for this field depends on what kind of calculation is performed:

- If the calculation entry is COUNT, the default is 10.
- If the calculation entry is SUM, MEAN, STANDARD_DEVIATION, MINIMUM, MAXIMUM, COUNT_PER_SECOND, or SUM_PER_SECOND, the default is 15.

DISPLAY_HEADERS or *DH*

Specifies a boolean value that modifies the value of the DISPLAY_HEADERS parameter specified earlier by a PUT_FIELD_SUMMARY subcommand. The default is FALSE.

TRUE

Column headers are displayed.

FALSE

Column headers are not displayed.

ROW_LABEL or *RL*

Defines a record that modifies the row label information defined earlier by a PUT_INTERVAL_FIELD subcommand. The record has the following format:

(*label, start_column, column_width, date_time_format*)

label

Specifies the row label. You can specify a string of 1 to 4,095 characters, or you can specify one of the following keywords; the default is TIME_RANGE:

START_TIME or ST

Beginning of the interval for the information collected. The beginning of the interval is the date and time of the first statistic in the previous interval. For a nonincremental counter, the full START_TIME value for the first interval is as follows:

```
1900-01-01 00:00:00.000.
```

For an incremental counter, the full START_TIME value for the first interval is the date and time of the base counter.

END_TIME or ET

End of the interval for the information collected. The end of the interval is the date and time of the first statistic in the current interval.

TIME_RANGE or TR

The START_TIME and END_TIME time range.

NONE or N

No row label appears.

start_column

Defines the column in which the row label begins. You can specify an integer from 1 to 10,000. You must define starting columns in such a way that columns do not overlap.

column_width

Defines the width of the column in which the row label begins. You can specify an integer from 1 to 10,000. You must define column widths in such a way that columns do not overlap.

date_time_format

Defines a string describing the format of the date and/or time appearing in the row label. The string must consist of a valid SCL date/time format string as documented in the NOS/VE Commands and Functions manual. The default is 'HMS'.

ROW_LABEL_FORMAT or *RLF*

Defines a record that modifies the value of the ROW_LABEL_FORMAT parameter defined earlier by a PUT_FIELD_SUMMARY subcommand. The record has the following format:

(*start_column, column_width*)

start_column

Defines the column in which the row label begins. You can specify an integer from 1 to 10,000. You must define starting columns in such a way that columns do not overlap.

column_width

Defines the width of the row label column. You can specify an integer from 1 to 10,000. You must define column widths in such a way that columns do not overlap.

REPORT_INTERVAL or *RI*

Specifies an integer that modifies the value of the REPORT_INTERVAL parameter specified earlier by a PUT_INTERVAL_FIELD subcommand. You can specify an integer from 1 to 10,000. Specifying a value of 1 causes all occurrences to be printed. Specifying a value of 10 causes 10 consecutive occurrences to be summarized for one report interval.

STRING or *STRINGS* or *STR*

Defines a list of 1 to 11 strings that replace those strings defined earlier by a PUT_STRING subcommand. Each string consists of 1 to 10,000 characters.

SELECTION or *SELECTIONS* or *S*

Specifies the name of one or more selections that replace those specified earlier by a PUT_RECORD subcommand.

COUNTER or *COUNTERS* or *C*

Specifies a keyword or a list of one or more records indicating which counter or counters of the statistics in the named selection will be displayed in the report. The keyword you can specify is NONE (no counters are printed).

The record has the following format:

(*counter_number, base*)

counter_number

Defines a range of counters whose data is to be printed. You can specify integers from 1 to 4,095. You can also specify the keyword ALL. In this case, all counters for the named selection are printed. The default is ALL.

base

Specifies the numeric base in which the counter or counters are printed. You can specify one of the following keywords; the default is B10:

BASE_2 or B2
BASE_8 or B8
BASE_10 or B10
BASE_16 or B16
BASE_16_GROUP or B16G

DESCRIPTIVE_DATA or *DD*

Defines a different substring of the PUT_RECORD subcommand's DESCRIPTIVE_DATA field that you want to appear as output.

You can define a string of 0 to 4,095 characters, or you can define a record in the following format:

(*position, length, field_number, field_delimiter*)

position

Defines the starting position of the descriptive data field. You can specify an integer from 1 to 4,095. The default is 1.

length

Defines the length of the substring. You can specify an integer from 1 to 4,095, or you can specify the keyword ALL. The default is ALL.

field_number

Defines the field number for the substring. You can specify an integer from 1 to 4,095. You can also specify the keyword ALL. The default is ALL.

field_delimiter

Defines a string containing the character that delimits the subfield. The default is a comma (,).

STATUS

Returns the completion status for this subcommand.

Remarks

- You cannot change the type of put entry. For example, you cannot use the CHANGE_PUT subcommand to change a PUT_FIELD operation to a PUT_FIELD_SUMMARY operation.
- The following table indicates which output definition subcommands are valid for which parameters on the CHANGE_PUT subcommand:

Put Entry	Parameter
PUT_FIELD	PUT NUMBER NEW_PUT NEW_NUMBER FIELD
PUT_FIELD_SUMMARY	PUT NUMBER NEW_PUT NEW_NUMBER FIELD SUMMARY_CALCULATION DISPLAY_HEADERS ROW_LABEL_FORMAT
PUT_INTERVAL_FIELD	PUT NUMBER NEW_PUT NEW_NUMBER FIELD ROW_LABEL REPORT_INTERVAL
PUT_RECORD	PUT NUMBER NEW_PUT NEW_NUMBER FIELD SELECTION COUNTER DESCRIPTIVE_DATA
PUT_STRING	PUT NUMBER NEW_PUT NEW_NUMBER FIELD STRING

Examples

The following example changes the relative position of the TASK_END_INTERVAL report entry, as well as the row label; the date/time format is changed from 'MS' to 'HMS'; the length of the row label is changed from 12 to 8; and the relative position of the report entry is changed to the third position:

```
put_interval_field put=task_end_interval ..
  field=((cp_time_in_job_mode,, 14, 8, 'Job Mode', 'Cpu Time') ..
         (cp_time_in_monitor_mode,,, 12, 'Monitor Mode', '   Cpu Time ') ..
         (page_faults,,, 5, 'Page ', 'Fault') ..
         (page_ins,,, 4, 'Page', 'Ins ') ..
         (page_reclaims,,, 8, ' Page ', 'Reclaims') ..
         (page_assigns,;, 6, ' Page ', 'Assign') ..
         (maximum_job_working_set,,, 11, 'Maximum', 'Working Set') ..
         (number_of_slowed_down,,, 6, 'Slowed', ' Down ') ..
  row_label=(end_time 1  12  'MS')
change_put put=task_end_interval new_number=3 ..
  row_label=(,, 8  'HMS')
```

CHANGE_SELECTION Subcommand

Purpose Changes selection criteria of an existing selection.

Format **CHANGE_SELECTION** or
CHAS
  SELECTION=name
  NEW_SELECTION=name
  STATISTIC_CODE=keyword or *statistic_code*
  TIME=keyword or *range of date_time*
  CONTINUOUS_DATE_TIME=boolean
  DESCRIPTIVE_DATA=keyword or *list of record*
  JOB_PREDECESSOR=keyword or *name*
  TASK_PREDECESSOR=keyword or *name*
  SYSTEM_JOB_NAME=keyword or *name*
  GLOBAL_TASK_ID=keyword or *string*
  STATUS=status variable

Parameters **SELECTION** or **S**

Specifies the name of the selection to be changed. The selection must have been previously defined. This parameter is required.

NEW_SELECTION or *NS*

Defines the new name of the selection. The selection must not have been previously defined.

STATISTIC_CODE or *SC*

Defines a new statistic you want selected. The statistic code must consist of two characters followed by a four- or five-digit number. Statistic code numbers from 0 to 9,999 are reserved for Control Data. Statistic code numbers from 10,000 to 19,000 are reserved for site codes and user-defined codes. You can also specify the keyword NONE. Specifying NONE removes this characteristic from the list of selection criteria for the selection named.

TIME or *T*

Defines a new date, time, or date/time range. To be included in a selection, statistics must have been emitted within the range specified. The date_time data type is a record having the following format:

```
(year-month-day.hour:minute:second.millisecond)
```

You can specify the date and time components alone or together. Examples are as follows:

```
time=1989-11-21.08:00 .. 1989-11-21.17:00

time=08:00 .. 17:00
```

If you specify only the date portion of this parameter, the utility completes the time portion for you. Similarly, if you specify only the time portion, the utility completes the date portion for you. The utility completes the date/time value according to its lowest and highest time and date ranges. For example, specifying TIME=1989-10-28 causes the utility to interpret the value of this parameter as follows:

```
1989-10-28.00:00:00.000 .. 1989-10-28.23:59:59.999
```

Specifying TIME=8:00 .. 16.00 and CONTINUOUS_DATE_TIME=FALSE, causes the utility to interpret the value of this parameter as follows:

```
1900-01-01.08:00:00.000 .. 2155-12-31.16:00:00.000
```

The date_time value can also be expressed as a date/time string in a valid date/time format. For a list of date/time formats, see the NOS/VE System Usage manual.

You can also specify the keyword NONE for this parameter. Specifying NONE removes this characteristic from the list of selection criteria for the selection named.

CONTINUOUS_DATE_TIME or *CDT*

Specifies a boolean value that changes the value of the CONTINUOUS_DATE_TIME parameter specified by an earlier ADD_SELECTION subcommand.

TRUE

The date and time ranges are considered together. This means that the first time in the time range only applies to the first date in the date range, the second time in the time range only applies to the second date in the date range, and so on.

FALSE

The time range is treated independently of the date range. This means that, for every date in the date range, the selection or selections to be chosen are based on the time range for that day.

DESCRIPTIVE_DATA or *DD*

Modifies the value of the DESCRIPTIVE_DATA parameter defined by an earlier ADD_SELECTION subcommand. You can specify a string containing the descriptive data field, a substring or a list of substrings of the descriptive data field, or the keyword NONE. Specifying NONE removes this characteristic from the list of selection criteria for the selection named.

The descriptive data field is defined by a record having the following format:

(string, position, length, field_number, field_delimiter)

string

Defines a descriptive data field expressed as a string of 0 to 4,095 characters. You can also define a list of strings.

position

Defines the starting position of the descriptive data field or substring. You can specify an integer from 0 to 4,095. The default is 1.

length

Defines the length of the substring. You can specify an integer from 0 to 4,095. You can also specify the keyword ALL. The default is ALL.

field_number

Defines the field number for the substring. You can specify an integer from 1 to 4,095. You can also specify the keyword ALL. The default is ALL.

field_delimiter

Defines a string containing the character that delimits the subfields of a descriptive data field. The default is a comma (,).

JOB_PREDECESSOR or *JP*

Modifies the value of the JOB_PREDECESSOR parameter defined by an earlier ADD_SELECTION subcommand. The selection must have been previously defined. You can either specify the name of the selection for which a statistic emitted by a particular job was previously logged, or you can specify the keyword NONE. Specifying NONE removes this characteristic from the list of selection criteria for the selection named. If you specify this parameter, do not specify the TASK_PREDECESSOR parameter.

TASK_PREDECESSOR or *TP*

Modifies the value of the TASK_PREDECESSOR parameter defined by an earlier ADD_SELECTION subcommand. The selection must have been previously defined. You can either specify the name of the selection for which a statistic emitted by a particular task was previously logged, or you can specify the keyword NONE. Specifying NONE removes this characteristic from the list of selection criteria for the selection named. If you specify this parameter, do not specify the JOB_PREDECESSOR parameter.

SYSTEM_JOB_NAME or *SJN*

Modifies the value of the SYSTEM_JOB_NAME parameter defined by an earlier ADD_SELECTION subcommand. You can either specify the system-supplied name of the job by which statistics must have been emitted in order for them to be included in a selection, or you can specify the keyword NONE. Specifying NONE removes this characteristic from the list of selection criteria for the selection named. If you specify the system-supplied job name, it must be the full 19-character name; an abbreviation is not valid.

GLOBAL_TASK_ID or *GTI*

Modifies the value of the GLOBAL_TASK_ID parameter defined by an earlier ADD_SELECTION subcommand. You can either specify the global task identifier of the task emitting the statistic you want selected, or you can specify the keyword NONE. Specifying NONE removes this characteristic from the list of selection criteria for the selection named. For the format of this parameter, see the ADD_SELECTION subcommand description.

STATUS

Returns the completion status for this subcommand.

Remarks

Avoid using recursive job predecessors and task predecessors. If you do use them, employ one of the following methods to delete a selection from a recursive predecessor chain:

- Enter the DELETE_SELECTION subcommand with the SELECTION parameter set to ALL.
- Break the recursive nature of the predecessor by changing the name of the selection in question (CHANGE_SELECTION subcommand) and then deleting the original selection (DELETE_SELECTION subcommand).

Examples

- In the following example, the DISK_UNIT_USE selection is created and then renamed DISK_CHANNEL_USE:

```
add_selection selection=disk_unit_use statistic_code=os3
change_selection selection=disk_unit_use ..
  new_selection=disk_channel_use
```

- In the following example, the TASK_BEGIN selection is created with a restriction on the times when PM0 statistics should be placed in the input log. The CHANGE_SELECTION subcommand removes that time restriction as a selection criterion.

```
add_selection selection=task_begin statistic_code=pm0 ..
  time=08:00:00 .. 16:00:00
change_selection selection=task_begin time=none
```

DELETE_FIELD Subcommand

Purpose Deletes one or more field definitions established by an ADD_FIELD subcommand.

Format
DELETE_FIELD or
DELETE_FIELDS or
DELF
  FIELD=keyword or **list of name**
  STATUS=status variable

Parameters

FIELD or **FIELDS** or **F**

Specifies the name or names of the fields to be deleted. You can also specify the keyword ALL. Specifying ALL causes all field definitions to be deleted. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks Before you can delete a field, you must delete all references to the field definition.

Examples The following example adds the CONTROL_USER field to the JOB_QUEUING_STARTED selection and then deletes that field:

```
add_field field=control_user selection=job_queuing_started ..
  desciptive_data=(144,31)
delete_field field=control_user
```

DELETE_PUT Subcommand

Purpose Deletes one or more report entries specified by an output definition subcommand.

Format **DELETE_PUT** or
DELP
PUT=keyword or *list of name*
NUMBER=keyword or *list of range of integer*
STATUS=status variable

Parameters *PUT* or *P*

Specifies the name or names of the report entries to be deleted. You can also specify the keyword ALL. Specifying ALL causes all report entries to be deleted. You must specify either this parameter or the NUMBER parameter, but you cannot specify both.

NUMBER or *N*

Specifies the relative position, within the list of output definition subcommands, of the report entry or entries to be deleted. You must specify either this parameter or the PUT parameter, but you cannot specify both. You can specify a range of integers or one of the following keywords:

ALL

All report entries are deleted.

LAST or L

The report entry with the highest ordinal is deleted.

STATUS

Returns the completion status for this subcommand.

Examples In the following example, the report entries 1, 2, 3, 7, 15, 16, 17, 18, 19, and 20 are deleted. The remaining report entries are resequenced (entry number 4 now occupies the first position, entry number 5 the second position, and so on).

```
delete_put number=(1..3,7,15..20)
```

DELETE_SELECTION Subcommand

Purpose Deletes one or more selections defined by an ADD_SELECTION subcommand.

Format

DELETE_SELECTION or
DELETE_SELECTIONS or
DELS
  SELECTION=keyword or **list of name**
  STATUS=status variable

Parameters

SELECTION or **S**

Specifies the name or names of the selections to be deleted. You can also specify the keyword ALL. Specifying ALL deletes all selections and their associated fields. This parameter is required.

STATUS

Returns the completion status for this subcommand.

Remarks

- You cannot restore a deleted selection. You will need to recreate it using the ADD_SELECTION subcommand.
- Before you can delete a selection, you must delete all references to the selection definition. These references can be from other selections, field definitions, or put definitions.

Examples The following example creates the DISK_UNIT_USE selection and then deletes it:

```
add_selection selection=disk_unit_use statistic_code=os3
delete_selection selection=disk_unit_use
```

DISPLAY_DEFAULT Subcommand

Purpose Displays the current defaults for the ANALYZE_BINARY_LOG utility.

Format **DISPLAY_DEFAULT** or
DISPLAY_DEFAULTS or
DISD
DISPLAY_OPTION=list of keyword
OUTPUT=file
STATUS=status variable

Parameters *DISPLAY_OPTION* or *DO*

Specifies which of the default values to display. You can specify one or more of the following keywords; the default is ALL:

COUNTER_FRACTION or CF

Displays the default counter fraction size.

DATE_TIME_FORMAT or DTF

Displays the default date_time format.

INPUT_LOG or IL or LOG or L

Displays the list of default binary logs.

LEGIBLE_DATA_MAX_PAGE_WIDTH or LDMPW

Displays the default maximum page width for the LEGIBLE_DATA format.

LIST_MAX_PAGE_WIDTH or LMPW

Displays the default maximum page width for the LIST format.

TIME_INCREMENT_FORMAT or TIF

Displays the default time increment format.

ALL

Displays all information relevent to this parameter.

OUTPUT or *O*

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

STATUS

Returns the completion status for this subcommand.

Remarks

For detailed descriptions of the DISPLAY_OPTION parameter keywords, see the corresponding descriptions for the CHANGE_DEFAULT subcommand earlier in this chapter.

Examples

The following example displays all the default values for the ANALYZE_BINARY_LOG utility.

```
display_default display_option=all
```

The following information is displayed:

```
ANABL Default Values:
        Counter Fraction :  3
        Date Time Format : ISOD MILLISECOND
   Legible data max page width :   10000
   List max page width : 132
   Time Increment Format : SECONDS
   Input Log List        : UNSPECIFIED
```

DISPLAY_FIELD Subcommand

Purpose Displays one or more field definitions.

Format **DISPLAY_FIELD** or
DISPLAY_FIELDS or
DISF
FIELD=keyword or *list of name*
DISPLAY_OPTION=keyword
OUTPUT=file
STATUS=status variable

Parameters *FIELD* or *FIELDS* or *F*

Specifies the name or names of the fields to be displayed. You can also specify the keyword ALL (or A). Specifying ALL causes all defined fields to be displayed. The default is ALL.

DISPLAY_OPTION or *DO*

Specifies the type of field information you want displayed. You can specify one of the following keywords; the default is NAME:

ALL or A

All field attributes, including their names, are displayed.

NAME or NAMES or N

Only the names of the fields are displayed.

OUTPUT or *O*

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

STATUS

Returns the completion status for this subcommand.

Examples

In the following example, these activities are performed:

1. Selection JOB_END is created.
2. The fields JOB_END_HEADER, CP_TIME_IN_JOB_MODE, and PAGE_FAULTS are created for the JOB_END selection.
3. Selection JOB_QUEUING_STARTED is created.
4. The fields SYSTEM_JOB_NAME and CONTROL_USER are added for the JOB_QUEUING_STARTED selection.
5. Information about the fields is displayed.

```
add_selection selection=job_end statistic_code=jm3
add_field field=job_end_header selection=job_end string='JOB END STATISTICS'
add_field field=cp_time_in_job_mode selection=job_end counter=(1, 0.000001)
add_field field=page_faults selection=job_end counter=3
add_selection selection=job_queuing_started statistic_code=jm4
add_field field=system_job_name selection=job_queuing_started descriptive_data=(1,19)
add_field field=control_user selection=job_queuing_started descriptive_data=(144,31)
put_field put=header field=job_end_header
put_field_summary put=job_end_summary field=(cp_time_in_job_mode page_faults)
put_interval_field put=job_control_user ..
  field=((control_user,,,,,'CONTROL USER                   ') ..
  (system_job_name,,,,,'SYSTEM JOB NAME                ')) row_label=(end_time,,15)
display_field display_option=all
```

The display appears as follows:

```
Field: JOB_END_HEADER
  Selection: JOB_END
 Field Type: Text
    Text: JOB END STATISTICS
  Put Report Reference :
    Number :  1   Name : HEADER                   Put Type : Put Field

Field: CP_TIME_IN_JOB_MODE
  Selection: JOB_END
 Field Type: Counter
    Counter Number:     1
    Multiplier:     1.0000000000000E-0006
  Put Report Reference :
    Number :  2   Name : JOB_END_SUMMARY          Put Type : Put Field Summary

Field: PAGE_FAULTS
  Selection: JOB_END
 Field Type: Counter
    Counter Number:     3
    Multiplier:     1.0000000000000E+0000
  Put Report Reference :
    Number :  2   Name : JOB_END_SUMMARY          Put Type : Put Field Summary

Field: SYSTEM_JOB_NAME
  Selection: JOB_QUEUING_STARTED
 Field Type: Descriptive Data
    Position:  1  Length: 19  Field Number: ALL  Field Delimiter: ','
  Put Report Reference :
    Number :  3   Name : JOB_CONTROL_USER         Put Type : Put Interval Field

Field: CONTROL_USER
  Selection: JOB_QUEUING_STARTED
 Field Type: Descriptive Data
    Position:  144  Length: 31  Field Number: ALL  Field Delimiter: ','
  Put Report Reference :
    Number :  3   Name : JOB_CONTROL_USER         Put Type : Put Interval Field
```

DISPLAY_LOGGED_STATISTICS Subcommand

Purpose Displays the names, counts, and time ranges for statistics collected in a binary log.

Format **DISPLAY_LOGGED_STATISTICS** or
DISLS
 INPUT=list of file
 OUTPUT=file
 DISPLAY_OPTION=keyword
 STATUS=status variable

Parameters INPUT or I

Specifies the name or names of the binary logs whose statistics are to be displayed. This parameter is required.

OUTPUT or *O*

Specifies the name of the file to which the logged statistics are to be written. The default is $OUTPUT.

DISPLAY_OPTION or *DO*

Specifies the type of statistic information you wanted displayed. You can specify one of the following keywords; the default is NAMES:

ALL

The following information is displayed:

Statistic code
Number of occurrences in the log
Date and time of first occurrence in the log
Date and time of last occurrence in the log

NAME or NAMES or N

Only the statistic codes of the logged statistics are displayed.

STATUS

Returns the completion status for this subcommand.

Remarks

- The data for the complete report is always retained after this subcommand has been executed once. The data is kept until the input log is changed to a different file. This practice minimizes the number of times the log is scanned.

- Files specified by the INPUT parameter are closed only when either of the following actions takes place:

  - You issue a QUIT subcommand.

  - You issue one of the following subcommands in which the INPUT parameter does not contain the name of the file you want closed:

    DISPLAY_LOGGED_STATISTICS
    GENERATE_LOG
    GENERATE_REPORT

Examples

The following example displays complete information about statistics found in the STATISTIC_LOG input log:

```
display_logged_statistics input=statisic_log display_option=all
```

The following display appears:

```
statistic   number of    first occurrence       last occurrence
  code     occurrences   date       time        date       time
   CB0             288  1989-07-26 00:08:27  1989-07-26 13:47:27
   JM0             920  1989-07-26 00:00:57  1989-07-27 00:00:13
   JM2             920  1989-07-26 00:00:57  1989-07-27 00:00:13
   JM3             919  1989-07-26 00:01:12  1989-07-26 23:57:19
   JM4             927  1989-07-26 00:00:57  1989-07-27 00:00:13
   LG0               2  1989-07-26 00:00:35  1989-07-27 00:00:31
   MV0              74  1989-07-26 05:19:39  1989-07-26 23:14:55
   MV1             566  1989-07-26 07:14:17  1989-07-26 22:50:24
   MV2              42  1989-07-26 07:17:58  1989-07-26 16:10:12
   MV3              49  1989-07-26 07:10:48  1989-07-26 16:11:22
   NA0             288  1989-07-26 00:03:27  1989-07-26 23:58:27
   NA1             288  1989-07-26 00:03:27  1989-07-26 23:58:27
   OS0             288  1989-07-26 00:03:27  1989-07-26 23:58:27
   OS1             288  1989-07-26 00:03:27  1989-07-26 23:58:27
   OS2             288  1989-07-26 00:03:27  1989-07-26 23:58:27
   OS3            3456  1989-07-26 00:03:27  1989-07-26 23:58:27
   OS4            4608  1989-07-26 00:03:27  1989-07-26 23:58:27
   OS5             288  1989-07-26 00:03:27  1989-07-26 23:58:27
```

DISPLAY_PUT Subcommand

Purpose Displays the report entries or log entries specified by the output specification subcommands during the current utility session.

Format **DISPLAY_PUT** or
DISP
PUT=keyword or *list of name*
NUMBER=keyword or *list of range of integer*
DISPLAY_OPTION=list of keyword
OUTPUT=file
STATUS=status variable

Parameters *PUT* or *P*

Specifies the name or names of the report or log entries to be displayed. You can also specify the keyword ALL. Specifying ALL causes all report or log entries to be displayed. You must specify either this parameter or the NUMBER parameter, but you cannot specify both.

NUMBER or *N*

Specifies the relative position, within the list of output definition subcommands, of the report or log entry or entries to be displayed. You must specify either this parameter or the PUT parameter, but you cannot specify both. You can specify a range of integers (1 to $MAX_INTEGER) or one of the following keywords:

ALL

All report or log entries are displayed.

LAST or L

The report or log entry with the highest ordinal is displayed.

DISPLAY_OPTION or *DO*

Specifies the type of information you want displayed. You can specify one or more of the following keywords; the default is NAMES:

NAME or NAMES or N

The names of the report or log entries are displayed.

TYPE or T

The display indicates whether the report or log entry is associated with a PUT_RECORD, PUT_FIELD, PUT_FIELD_SUMMARY, or PUT_INTERVAL_FIELD subcommand.

ENTRIES or E

The display indicates the fields or records that will appear as output from each specified PUT_RECORD, PUT_FIELD, PUT_FIELD_SUMMARY, or PUT_INTERVAL_FIELD subcommand.

ALL

All information relevant to this parameter is displayed. This is equivalent to specifying the NAME, TYPE, and ENTRIES keywords.

OUTPUT or *O*

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

STATUS

Returns the completion status for this subcommand.

DISPLAY_SELECTION Subcommand

Purpose Displays the selection definitions for the current utility session.

Format **DISPLAY_SELECTION** or
DISPLAY_SELECTIONS or
DISS
SELECTION=keyword or *list of name*
DISPLAY_OPTION=keyword
OUTPUT=file
STATUS=status variable

Parameters *SELECTION* or *S*

Specifies the name or names of the selection definitions to be displayed. You can also specify the keyword ALL. This is the default. Specifying ALL (or defaulting to ALL) causes all selection definitions to be displayed.

DISPLAY_OPTION or *DO*

Specifies the type of information you want displayed. You can specify one of the following keywords; the default is NAME:

NAME or NAMES or N

Only the names of the selections are displayed.

ALL

All possible attributes are displayed for each selection specified.

OUTPUT or *O*

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

STATUS

Returns the completion status for this subcommand.

Examples

The following example creates selections TASK_BEGIN and TASK_END. It then creates fields CP_TIME_IN_JOB_MODE and CP_TIME_IN_MONITOR_MODE. Finally, it displays all the information about the selection references:

```
add_selection selection=task_begin statistic_code=pm0 time=08:00:00 .. 16:00:00
add_selection selection=task_end statistic_code=pm3 task_predecessor=task_begin ..
  descriptive_data='FCP$COMPILE_FORTRAN_SOURCE'
add_field field=cp_time_in_job_mode selection=task_end counter=(1, 0.000001)
add_field field=cp_time_in_monitor_mode selection=task_end counter=(2, 0.000001)
display_selection display_option=all
```

The following information is displayed:

```
 Selection : TASK_BEGIN
   Selection Criteria
     Statistic Code   : PM0
     Time             : 1900-01-01 08:00:00.000 .. 2155-12-31 16:00:00.000
     Continous        : FALSE
   Usage
     Number of task successor statistics  :  1

 Selection : TASK_END
   Selection Criteria
     Statistic Code   : PM3
     Descriptive Data :
'FCP$COMPILE_FORTRAN_SOURCE

                  '
     Predecessor Task : TASK_BEGIN
   Usage : No References
   Field References
     Counter Name : CP_TIME_IN_JOB_MODE             Counter Number :    1
     Counter Name : CP_TIME_IN_MONITOR_MODE         Counter Number :    2
```

GENERATE_LOG Subcommand

Purpose Produces an output log that is based on specifications established by the output defintion subcommands.

Format **GENERATE_LOG** or
GENL
  INPUT=list of file
  OUTPUT=file
  DISPLAY_FORMAT=keyword
  COUNTER_FORMAT=keyword
  NUMBER_OF_COUNTERS=keyword or *integer*
  DESCRIPTIVE_DATA_LENGTH=keyword or *integer*
  STATUS=status variable

Parameters INPUT or I

Specifies the name or names of the input binary logs from which the output log is to be generated. This parameter is required.

OUTPUT or O

Defines the name of the file to which the output log is to be written. This parameter is required.

DISPLAY_FORMAT or *DF*

Defines the type of format in which you want the output log information displayed. You can specify one of the following keywords; the default is LEGIBLE_DATA:

BINARY or B

The output format is identical to that of the original binary log. This format is intended for users who want to save a log in binary format.

LEGIBLE_DATA or LD

Program-readable format. All selected statistics appear in a single row.

LIST or L

User-readable format. Data from a single selection is placed in rows having a width not exceeding 255 columns. All selection header information is displayed, as well as all counter and descriptive data fields for every occurrence of the statistic.

COUNTER_FORMAT or *CF*

Defines the format in which the counter information will appear in the output log. This parameter applies to LIST and LEGIBLE_DATA formats. You can specify one of the following keywords; the default is VARIABLE:

FIXED or F

All counter information is printed with the same number of columns throughout.

VARIABLE or V or UNFIXED or UF

Information for each counter is printed with only the number of columns needed.

NUMBER _OF_COUNTERS or *NOC*

Defines the number of counters that will be printed in the output log. You can specify an integer from 0 to 4,095, or you can specify the keyword ALL. The default is ALL. By default, exactly the number of counters defined for the statistic are printed. If an output statistic has fewer counters than are specified for this parameter, the utility pads its output with zero values.

Use this parameter with the LIST and LEGIBLE_DATA formats when you intend to supply the output log as input to a program that wants to read a fixed number of counters.

DESCRIPTIVE _DATA _LENGTH or *DDL*

Defines the number of characters from the descriptive data field that will be printed in the output log. You can specify an integer from 0 to 4,095, or you can specify the keyword ALL. The default is ALL. By default, exactly the number of characters contained in the descriptive data field are printed. If a descriptive data field has fewer characters than are defined for this parameter, the utility pads its output with zero values.

STATUS

Returns the completion status for this subcommand.

Remarks

- The statistics written to the output log are those selected only by PUT_RECORD subcommands. In fact, the only subcommand that works with the GENERATE_LOG subcommand is PUT_RECORD.
- Since all of the counter selection and descriptive data parameters apply (the COUNTER and DESCRIPTIVE_DATA parameters of the PUT_RECORD subcommand, and the NUMBER_OF_COUNTERS and DESCRIPTIVE_DATA_LENGTH parameters of the GENERATE_LOG subcommand), you can use the GENERATE_LOG subcommand to define new statistics that are subsets of old statistics.
- Setting the DISPLAY_FORMAT parameter to BINARY renders the COUNTER_FORMAT parameter meaningless.
- Files specified by the INPUT parameter are closed only when either of the following actions takes place:
  - You issue a QUIT subcommand.
  - You issue one of the following subcommands in which the INPUT parameter does not contain the name of the file you want closed:

    DISPLAY_LOGGED_STATISTICS
    GENERATE_LOG
    GENERATE_REPORT

- The following table indicates the size of fixed counters in a log with LEGIBLE_DATA or LIST format:

Base	**Size**
Base 2	64 digits + 1 sign + 3 radix + 1 space
Base 8	22 digits + 1 sign + 3 radix + 1 space
Base 10	19 digits + 1 sign + 1 space
Base 16	16 digits + 1 sign + 4 radix + 1 space
Base 16 group	19 digits + 4 radix + 3 spaces

- If the output log is specified as LIST format and the counter format is FIXED, the length of the counter fields is the same for all counters and equal to the longest one. See the format examples in the description of the PUT_RECORD subcommand.

- The default page width for a log with LEGIBLE_DATA format is 10,000 characters. The default page width for a log with LIST format is 132 characters.

Examples For an example of output produced by a GENERATE_LOG subcommand, see the PUT_RECORD subcommand description, as well as Examples of Using Statistics.

GENERATE_REPORT Subcommand

Purpose Produces a report based on specifications established by the output definition subcommands for the current utility session.

Format **GENERATE_REPORT** or
GENR
  INPUT=list of file
  OUTPUT=file
  DISPLAY_FORMAT=keyword
  STATUS=status variable

Parameters INPUT or I

Specifies the name or names of the input binary logs from which the report is to be generated. This parameter is required.

OUTPUT or *O*

Defines the name of the file to which the output is to be written. The default is $OUTPUT.

DISPLAY_FORMAT or *DF*

Defines the format in which you want the report generated. You can specify one of the following keywords; the default is LIST:

EXCEL or E

Output is formatted for input to the Excel[1] spreadsheet program.

LEGIBLE_DATA or LD

Program-readable format. Each selected statistic appears in a single row.

LIST or L

User-readable format. Data from a single selection is placed in rows having a width not exceeding 255 columns. All selection header information is displayed, as well as all counter and descriptive data fields for every occurrence of the statistic.

STATUS

Returns the completion status for this subcommand.

1. Excel is a product of Microsoft Corporation.

Remarks

- The following are the only ANALYZE_BINARY_LOG subcommands that work with the GENERATE_REPORT subcommand:

  PUT_FIELD
  PUT_FIELD_SUMMARY
  PUT_INTERVAL_FIELD
  PUT_STRING

- Calculated field values involving a multiplier other than 1 appear as real numbers displayed to three decimal places. If the multiplier is 1, the result is an integer.

- Files specified by the INPUT parameter are closed only when either of the following actions takes place:

  - You issue a QUIT subcommand.

  - You issue one of the following subcommands in which the INPUT parameter does not contain the name of the file you want closed:

    DISPLAY_LOGGED_STATISTICS
    GENERATE_LOG
    GENERATE_REPORT

- If the counter value is large, a sum overflow can occur in the summary portion of the field. If that happens, the utility issues an error message and all the summaries that depended on that sum are meaningless. The following summary calculation fields can experience a sum overflow:

  SUM
  MEAN or AVERAGE
  STANDARD_DEVIATION
  SUM_PER_SECOND

- The default page width for a report is 132 characters.

- If a numeric field is too short, the ANALYZE_BINARY_LOG fills it in with asterisks. If a text field, descriptive data field, or date/time field is too short, the text is truncated on the right.

Examples

For examples of output produced by a GENERATE_REPORT subcommand, see the PUT_FIELD, PUT_FIELD_SUMMARY, and PUT_INTERVAL_FIELD subcommand descriptions, the Examples of Using Statistics section, and the Site Analyst Examples online manual.

POP_PAGE_HEADER Subcommand

Purpose Specifies a page header entry that is not to appear in the remainder of a report.

Format **POP_PAGE_HEADER** or
POPPH
  POP_COUNT=keyword or *integer*
  PUT=name
  NUMBER=keyword or *integer*
  STATUS=status variable

Parameters *POP_COUNT* or *PC*

Defines the number of page header entries to be removed from the current page header stack. The definitions for the page header entries that are popped from the stack are not deleted from the list of output definition subcommands. You can specify an integer from 1 to $MAX_INTEGER, or you can specify the keyword ALL. The default is 1.

PUT or *P*

Defines the name of the report entry.

NUMBER or *N*

Defines where you want the current subcommand placed within the list of output definition subcommands. You can specify an integer from 1 to $MAX_INTEGER, or you can specify the keyword NEXT (or N). The default is NEXT. This means that the new output definition subcommand is placed after all current output definition subcommands. If you specify an integer, the new subcommand occupies that relative position. If you specify a relative position greater than the largest used ordinal, the subcommand is placed at the end of the list of subcommands.

STATUS

Returns the completion status for this subcommand.

Remarks

- This subcommand has the same characteristics as other subcommands that define PUT parameters. That is, the name of the report entry defined by the PUT parameter can be displayed and manipulated by the DISPLAY_PUT, CHANGE_PUT, and DELETE_PUT subcommands.
- This subcommand works only with the GENERATE_REPORT subcommand. It does not work with the GENERATE_LOG subcommand.

PUSH_PAGE_HEADER Subcommand

Purpose Specifies a page header entry to be inserted at the top of each page of a report.

Format **PUSH_PAGE_HEADER** or
PUSPH
PAGE_HEADER=keyword or *list of record*
PUT=name
NUMBER=keyword or *integer*
STATUS=status variable

Parameters *PAGE_HEADER* or *PH*

Defines one or more page header entry records. You can define a record or you can specify the keyword DEFAULT. Specifying DEFAULT causes the ANALYZE_BINARY_LOG utiltiy to use its standard page header. The default is DEFAULT. If you define a record, it has the following format:

(**string**, *start_column, column_width*)

string

Defines the string you want inserted in the report. You must specify 1 to 10,000 characters. This entry is required.

start_column

Defines the column in which the string begins. You can specify an integer from 1 to 10,000. You must define starting columns in such a way that columns do not overlap. The default for this field is the last position in the previous field plus 2. If there is no row label, the default start column for the first field is 1. If there is a row label, the default start column for the first field is the last position in the row label plus 1.

column_width

Defines the width of the column in which the string begins. You can specify an integer from 1 to 10,000. You must define column widths in such a way that columns do not overlap.

PUT or *P*

Defines the name of the report entry.

NUMBER or *N*

Defines where you want the current subcommand placed within the list of output definition subcommands. You can specify an integer from 1 to $MAX_INTEGER, or you can specify the keyword NEXT (or N). The default is NEXT. This means that the new output definiition subcommand is placed after all current output definition subcommands. If you specify an integer, the new subcommand occupies that relative position. If you specify a relative position greater than the largest used ordinal, the subcommand is placed at the end of the list of subcommands.

STATUS

Returns the completion status for this subcommand.

Remarks

- This subcommand has the same characteristics as other subcommands that define PUT parameters. That is, the name of the report entry defined by the PUT parameter can be diplayed and manipulated by the DISPLAY_PUT, CHANGE_PUT, and DELETE_PUT subcommands.
- Page headers are pushed onto a page header stack. The most recent page header is printed last.
- This subcommand works only with the GENERATE_RECORD subcommand. It does not work with the GENERATE_LOG subcommand.

Examples

The following example defines a page header that appears at the top of each page of a report:

```
push_page_header page_header=(('Statistics Analysis Report',1,30) ..
  ('Job End Data',40,30)) put=main_heading number=1
```

The following page header is printed:

```
Statistics Analysis Report           Job End Data
```

PUT_FIELD Subcommand

Purpose Defines report entries of one or more fields and their respective display options.

Format **PUT_FIELD** or
PUTF
FIELD=list of record
PUT=name
NUMBER=keyword or *integer*
STATUS=status variable

Parameters **FIELD** or **FIELDS** or **F**

Defines 1 to 25 records that describe fields that are to appear in the report. This parameter is required.

Each field can contain a counter, a descriptive data field, an elapsed time field, an elapsed time calculation field, a statistic header, or a string.

Each record has the following format:

(**field_name**, *display_option, start_column, column_width, header_1, header_2*)

field_name

Specifies the name of the field that will appear in the report. This entry is required.

display_option

Specifies a keyword indicating how data is to be displayed in the specified field. You can specify the following keywords:

ALL_OCCURRENCES or AO or DETAIL or D

Value of the field from each occurrence of the field's selection in the log. This results in one line of output for each occurrence of the field's selection in the log.

FIRST_OCCURRENCE or FO

Value of the field from the first occurrence of the field's selection in the log.

LAST_OCCURRENCE or LO

Value of the field from the last occurrence of the field's selection in the log.

COUNT or C

Number of occurrences of the field in the log.

SUM or S

Sum of the values of all occurrences of the field in the log.

MEAN or M or AVERAGE or AVG

Average as computed by dividing the SUM field by the COUNT field.

STANDARD_DEVIATION or SD

Normal standard deviation with a divisor of (n – 1).

MINIMUM or MIN

Minimum value of all occurrences of the field in the log.

MAXIMUM or MAX

Maximum value of all occurrences of the field in the log.

COUNT_PER_SECOND or CPS

Average as computed by dividing the COUNT field by the average time interval between occurrences of a field.

SUM_PER_SECOND or SPS

Average as computed by dividing the SUM field by the average time interval between occurrences of a field.

start_column

Defines the column in which the display_option begins. You can specify an integer from 1 to 10,000. You must define starting columns in such a way that columns do not overlap.

column_width

Defines the width of the column in which the display_option begins. You can specify an integer from 1 to 10,000. You must define column widths in such a way that columns do not overlap.

header_1

Defines a string that is displayed as the first line of that column header in the report. The string can be from 1 to 10,000 characters. By default, the field name is used as the column header.

header_2

Defines a string that is displayed as the second line of that column header in the report. The string can be from 1 to 10,000 characters.

PUT or *P*

Defines the name of the report entry.

NUMBER or *N*

Defines where you want the current subcommand placed within the list of output definition subcommands. You can specify an integer from 1 to $MAX_INTEGER, or you can specify the keyword NEXT (or N). The default is NEXT. This means that the new output definition subcommand is placed after all current output definition subcommands. If you specify an integer, the new subcommand occupies that relative position. If you specify a relative position greater than the largest used ordinal, the subcommand is placed at the end of the list of subcommands.

STATUS

Returns the completion status for this subcommand.

Remarks

- The purpose of this subcommand is to give you more control over the report format than is provided by the PUT_FIELD_SUMMARY subcommand.
- This subcommand works only with the GENERATE_REPORT subcommand. It does not work with the GENERATE_LOG subcommand.
- The statistical column numbers for fields whose contents are to be listed using this subcommand must be in ascending order. For example, you cannot list a field occupying columns 50 to 59 before you list a field occupying columns 40 to 49.
- When specifying column widths, be sure to reserve a column for the sign. For example, if you expect the DISPLAY_OPTION subfield to have a maximum of 10 digits, you should reserve 11 columns. This is true even if all numbers are positive.
- When using the DISPLAY_OPTION subfield, the keyword ALL_OCCURRENCES has the following requirements:
  - You cannot mix ALL_OCCURRENCES with other display options. If you choose the ALL_OCCURRENCES display option for one field of a PUT_FIELD subcommand, then you must use the ALL_OCCURRENCES display option for the rest of the fields.
  - All fields must belong to the same predecessor chain. When specifying fields belonging to different selections in the same PUT_FIELD subcommand, these fields must be part of the same chain of selections linked as job predecessors or task predecessors.

- The following table shows which display options are allowed for each of the listed field types.

Numeric Field Type	Display Options Allowed
Counter	All display options.
Number of counters	All display options
Descriptive data length	All display options
Elapsed time	All display options except SUM_PER_SECOND
Value per second	All display options except SUM_PER_SECOND
Occurrence per second	All display options except SUM_PER_SECOND
Text Field Type	**Display Options Allowed**
Descriptive data	ALL_OCCURRENCES FIRST_OCCURRENCE LAST_OCCURRENCE COUNT COUNT_PER_SECOND
Statistic code	ALL_OCCURRENCES FIRST_OCCURRENCE LAST_OCCURRENCE COUNT COUNT_PER_SECOND
Date time	ALL_OCCURRENCES FIRST_OCCURRENCE LAST_OCCURRENCE COUNT COUNT_PER_SECOND
System job name	ALL_OCCURRENCES FIRST_OCCURRENCE LAST_OCCURRENCE COUNT COUNT_PER_SECOND
Global task ID	ALL_OCCURRENCES FIRST_OCCURRENCE LAST_OCCURRENCE COUNT COUNT_PER_SECOND
String	ALL_OCCURRENCES FIRST_OCCURRENCE LAST_OCCURRENCE

Examples

In the following example, the data for the report comes from three selections: JOB_BEGIN, JOB_MODE, and JOB_END. These three selections create a chain of predecessors with JOB_END as the tail selection. This means for every statistic selected that meets the JOB_END selection criteria, there are corresponding statistics with the same system job name selected for the JOB_BEGIN and JOB_MODE selections. This allows the ANALYZE_BINARY_LOG utility to report fields from each statistic and match the fields based on the system job name. Since the ALL_OCCURRENCES display option is selected on the PUT_FIELD subcommand, when the utility selectes a statistic for JOB_END it records both JOB_END fields specified: JOB_JM_CPU and JOB_MM_CPU. The utility also records the USER_NAME and JOB_MODE fields from statistics in the chain of predecessors.

The tail selection, JOB_END, can be a predecessor to a selection that does not have fields in the report. In the example, the JOB_NAME selection does not have fields in the report but it is the link between the JOB_MODE selection and the JOB_BEGIN selection.

```
add_selection selection=job_begin statistic_code=jm0 time=08:00:00 .. 09:00:00
add_selection selection=job_name statistic_code=jm1 job_predecessor=job_begin
add_selection selection=job_mode statistic_code=jm2 job_predecessor=job_name
add_selection selection=job_end statistic_code=jm3 job_predecessor=job_mode
add_field field=user_name selection=job_begin descriptive_data=(1,31)
add_field field=job_mode selection=job_mode descriptive_data=(1,11)
add_field field=job_jm_cpu selection=job_end counter=(1, 0.000001)
add_field field=job_mm_cpu selection=job_end counter=(2, 0.000001)
put_field field=((user_name, all_occurrences,, 31, 'User Name') ..
              (job_mode, all_occurrences,, 11, 'Job Mode') ..
              (job_jm_cpu, all_occurrences,, 15, 'JM CPU') ..
              (job_mm_cpu, all_occurrences,, 15, 'MM CPU'))
generate_report input=log1 output=out
```

PUT_FIELD_SUMMARY Subcommand

Purpose Defines field summary rows that will appear in a report.

Format PUT_FIELD_SUMMARY or
PUTFS
 FIELD=list of record
 DISPLAY_OPTION=list of record
 DISPLAY_HEADERS=boolean
 ROW_LABEL_FORMAT=record
 PUT=name
 NUMBER=keyword or *integer*
 STATUS=status variable

Parameters FIELD or FIELDS or F

Specifies one or more records that name previously defined fields. This parameter is required. Each record has the following format:

(**field_name**, *row_label*)

field_name

Specifies the name of the previously defined field. This entry is required.

row_label

Defines the row label. This field must be expressed as a string. By default, the field_name value is used as the row label.

DISPLAY_OPTION or *DO*

Defines one or more records that describe how and where numeric data contained in a field is to be displayed. If the field defined by the FIELD parameter contains numeric data, you must specify the DISPLAY_OPTION parameter. If the field defined by the FIELD parameter contains a string, the DISPLAY_OPTION parameter is ignored.

The record has the following format:

(**calculation**, *start_column*, *column_width*)

The following are the default values for the entire parameter. If you specify any field within this parameter, none of the default values apply.

```
((count 32 10)
(sum 43 15)
(mean 59 15)
(standard_deviation 85 15)
(minimum 101 15)
(maximum 117 15))
```

calculation

Specifies a keyword indicating how you want numeric data represented in the report. This entry is required.

You can specify the following keywords:

COUNT or C

Number of occurrences of the field in the log.

SUM or S

Sum of the values of all occurrences of the field in the log.

MEAN or M or AVERAGE or AVG

Average as computed by dividing the SUM field by the COUNT field.

STANDARD_DEVIATION or SD

Normal standard deviation with a divisor of (n – 1).

MINIMUM or MIN

Minimum value of all occurrences of the field in the log.

MAXIMUM or MAX

Maximum value of all occurrences of the field in the log.

COUNT_PER_SECOND or CPS

Average as computed by dividing the COUNT field by the average time interval between occurrences of a field.

SUM_PER_SECOND or SPS

Average as computed by dividing the SUM field by the average time interval between occurrences of a field.

start_column

Defines the column in which the calculation value begins. You can specify an integer from 1 to 10,000. You must define starting columns in such a way that columns do not overlap. The default for this field is the last position in the previous field plus 2. If there is no row label, the default start column for the first field is 1. If there is a row label, the default start column for the first field is the last position in the row label plus 1.

column_width

Defines the width of the column in which the calculation value begins. You can specify an integer from 1 to 10,000. You must define column widths in such a way that columns do not overlap. The default for this field depends on what kind of calculation is performed:

- If the calculation entry is COUNT, the default is 10.
- If the calculation entry is SUM, MEAN, STANDARD_DEVIATION, MINIMUM, MAXIMUM, COUNT_PER_SECOND, or SUM_PER_SECOND, the default is 15.

DISPLAY_HEADERS or *DH*

Specifies a boolean value indicating whether column headers are displayed for the fields specified by this subcommand. The default is FALSE.

TRUE

Column headers are displayed.

FALSE

Column headers are not displayed.

ROW_LABEL_FORMAT or *RLF*

Defines a record describing the format of the row labels that are defined in the FIELD parameter. The record has the following format:

(*start_column, column_width*)

start_column

Defines the column in which the row label begins. You can specify an integer from 1 to 10,000. You must define starting columns in such a way that columns do not overlap. The default for this field is 1.

column_width

Defines the width of the row label column. You can specify an integer from 1 to 10,000. You must define column widths in such a way that columns do not overlap. The default for this field is 31.

PUT or *P*

Defines the name of the report entry. The report entry must not have been previously defined.

NUMBER or *N*

Defines where you want the current subcommand placed within the list of output definition subcommands. You can specify an integer from 1 to $MAX_INTEGER, or you can specify the keyword NEXT. The default is NEXT (or N). This means that the new output definition subcommand is placed after all current output definition subcommands. If you specify an integer, the new subcommand occupies that relative position. If you specify a relative position greater than the largest used ordinal, the subcommand is placed at the end of the list of subcommands.

STATUS

Returns the completion status for this subcommand.

Remarks

- This subcommand works only with the GENERATE_REPORT subcommand. It does not work with the GENERATE_LOG subcommand.
- Although this subcommand provides you with a quick way of examining the field data, it does not report descriptive data fields or strings.
- When specifying column widths, be sure to reserve a column for the sign. For example, if you expect the DISPLAY_OPTION subfield to have a maximum of 10 digits, you should reserve 11 columns. This is true even if all numbers are positive.

Examples

The following example defines the TASK_BEGIN and TASK_END selections and their associated fields and specifies field summary rows for the log report:

```
add_selection selection=task_begin statistic_code=pm0 time=08:00:00 .. 16:00:00
add_selection selection=task_end statistic_code=pm3 ..
  task_predecessor=task_begin descriptive_data='FCP$COMPILE_FORTRAN_SOURCE'
add_field field=cp_time_in_job_mode     selection=task_end counter=(1, 0.000001)
add_field field=cp_time_in_monitor_mode selection=task_end counter=(2, 0.000001)
add_field field=page_faults             selection=task_end counter=3
add_field field=page_ins                selection=task_end counter=4
add_field field=page_reclaims           selection=task_end counter=5
add_field field=page_assigns            selection=task_end counter=6
add_field field=maximum_job_working_set selection=task_end counter=7
add_field field=numberm_of_slowed_down  selection=task_end counter=8
put_field_summary put=task_statistic field=(cp_time_in_job_mode cp_time_in_monitor_mode ..
  page_faults page_ins page_reclaims page_assigns maximum_job_working_set ..
  number_of_slowed_down) display_option=((sum 25 7) (mean 33 10) ..
  (standard_deviation 44 10) (minimum 55 7) (maximum 63 8) ..
  (sum_per_second 72 8)) display_headers=true row_label_format=(1 23)
generate_report input=log1
```

The following report is generated:

```
                                           Standard                     Sum
                            Sum      Mean  Deviation  Minmum  Maximum   /Sec
CP_TIME_IN_JOB_MODE      63.370     1.056       .511    .578    2.193   .046
CP_TIME_IN_MONITOR_MODE  15.666      .261       .163    .137     .804   .011
PAGE_FAULTS                7591   126.517     73.545      75      366  5.564
PAGE_INS                   2188    36.467     61.798       1      193  1.604
PAGE_RECLAIMS               680    11.333     36.521       0      206   .498
PAGE_ASSIGNS               6556   109.267      7.932      99      128  4.805
MAXIMUM_JOB_WORKING_SET   10231   170.517     14.968     150      225  7.499
NUMBERM_OF_SLOWED_DOWN        0     0.000      0.000       0        0  0.000
```

PUT_INTERVAL_FIELD Subcommand

Purpose Defines report entries that are based on occurrences, or groups of occurrences, of a statistic containing the defined field.

Format **PUT_INTERVAL_FIELD** or
PUTIF
  FIELD=list of record
  REPORT_INTERVAL=integer
  ROW_LABEL=record
  PUT=name
  NUMBER=keyword or *integer*
  STATUS=status variable

Parameters **FIELD** or **FIELDS** or **F**

Defines 1 to 25 records that describe fields that are to appear in the report. The fields must already have been defined. This parameter is required. Each record has the following format:

(**field_name**, *display_option, start_column, column_width, header_1, header_2*)

field_name

Specifies the name of the field to be reported. This entry is required.

display_option

Specifies a keyword indicating how numeric data is to be displayed in the specified field. If the field specified by the field_name entry contains a string, the display_option entry is ignored. The default is SUM.

You can specify the following keywords:

FIRST_OCCURRENCE or FO

Value of the field from the first occurrence of the field's selection in the log.

COUNT or C

Number of occurrences of the field in the log.

SUM or S

Sum of the values of all occurrences of the field in the log.

MEAN or M or AVERAGE or AVG

Average as computed by dividing the SUM field by the COUNT field.

STANDARD_DEVIATION or SD

Normal standard deviation with a divisor of (n - 1).

MINIMUM or MIN

Minimum value of all occurrences of the field in the log.

MAXIMUM or MAX

Maximum value of all occurrences of the field in the log.

COUNT_PER_SECOND or CPS

Average as computed by dividing the COUNT field by the average time interval between occurrences of a field.

SUM_PER_SECOND or SPS

Average as computed by dividing the SUM field by the average time interval between occurrences of a field.

start_column

Defines the column in which the display_option entry begins. You can specify an integer from 1 to 10,000. You must define starting columns in such a way that columns do not overlap. The default for this field is the last position in the previous field plus 2. If there is no row label, the default start column for the first field is 1. If there is a row label, the default start column for the first field is the last position in the row label plus 1.

column_width

Defines the width of the column in which the display_option entry begins. You can specify an integer from 1 to 10,000. You must define column widths in such a way that columns do not overlap. The default for this field depends on the display_option entered:

- If the display_option entry is COUNT, the default is 10.
- If the display_option entry is SUM, MEAN, STANDARD_ DEVIATION, MINIMUM, MAXIMUM, COUNT_PER_SECOND, or SUM_PER_SECOND, the default is 15.

header_1

Defines a string that is displayed as the first line of that column header in the report. The string can be up to 10,000 characters.

header_2

Defines a string that is displayed as the second line of that column header in the report. The string can be up to 10,000 characters.

REPORT_INTERVAL or *RI*

Specifies the number of consecutive occurrences of a field that constitute an interval (a time interval if you are reporting on a periodic statistic, and an event interval if you are reporting on an event statistic). You can specify an integer from 1 to 10,000. Specifying a value of 1 causes all occurrences to be printed. Specifying a value of 10 causes 10 consecutive occurrences to be summarized for one report interval.

ROW_LABEL or *RL*

Defines a record indicating the row label information. The record has the following format:

(*label, start_column, column_width, date_time_format*)

label

Defines the row label. You can specify a string of characters or you can specify one of the following keywords; the default is TIME_RANGE:

START_TIME or ST

Beginning of the interval for the information collected. The beginning of the interval is the date and time of the first statistic in the previous interval. For a nonincremental counter, the full START_TIME value for the first interval is as follows:

```
1900-01-01 00:00:00.000.
```

For an incremental counter, the full START_TIME value for the first interval is the date and time of the base counter.

END_TIME or ET

End of the interval for the information collected. The end of the interval is the date and time of the first statistic in the current interval.

TIME_RANGE or TR

The START_TIME and END_TIME time range.

NONE or N

No row label appears.

start_column

Defines the column in which the row label begins. You can specify an integer from 1 to 10,000. You must define starting columns in such a way that columns do not overlap.

column_width

Defines the width of the column in which the row label begins. You can specify an integer from 1 to 10,000. You must define column widths in such a way that columns do not overlap.

date_time_format

Defines a string describing the format of the date and/or time appearing in the row label. The string must consist of a valid SCL date/time string as documented for the $DATE and $TIME functions in the NOS/VE Commands and Functions manual.

PUT or *P*

Defines the name of the report entry.

NUMBER or *N*

Defines where you want the current subcommand placed within the list of output definition subcommands. You can specify an integer from 1 to $MAX_INTEGER, or you can specify the keyword NEXT. The default is NEXT (or N). This means that the new output specification subcommand is placed after all current output specification subcommands. If you specify an integer, the new subcommand occupies that relative position. If you specify a relative position greater than the largest used ordinal, the subcommand is placed at the end of the list of subcommands.

STATUS

Returns the completion status for this subcommand.

Remarks

- This subcommand works only with the GENERATE_REPORT subcommand. It does not work with the GENERATE_LOG subcommand.
- Although you can use this subcommand to report on both periodic and event statistics, it is particularly useful for reporting on periodic statistics.
- For columns containing numeric data, both the headers and the data itself are right-justified. For columns containing text data (including strings and descriptive data fields), both the headers and the data itself are left-justified.
- If the PUT_INTERVAL_FIELD subcommand contains references to incremental and nonincremental counters, the first interval (the base interval for the incremental counter) does not appear in the report for any fields specified by the subcommand (including fields for nonincremental counters).
- The field name specified in the FIELD parameter can contain a counter, a descriptive data field, an elapsed time field, an elapsed time calculation field, a statistic header, or a string. The source of the descriptive data field is the descriptive data field of the logged statistic.
- If one of the fields specified by this subcommand is a descriptive data field and the report interval is greater than 1, the descriptive data field appearing in the report will be the first such field from the intervals included in that line.
- The statistical column numbers for fields whose contents are to be listed using this subcommand must be in ascending order. For example, you cannot list a field occupying columns 50 to 59 before you list a field occupying columns 40 to 49.
- When defining column widths, be sure to reserve a column for the sign. For example, if you expect the DISPLAY_OPTION subfield to have a maximum of 10 digits, you should reserve 11 columns. This is true even if all numbers are positive.

Examples

The following example defines the TASK_BEGIN and TASK_END selections for analyzing PM3 statistics occurring between 8 a.m. and 4 p.m. and specifies the report lines for that time interval:

```
add_selection selection=task_begin statistic_code=pm0 ..
  time=08:00:00 .. 16:00:00
add_selection selection=task_end statistic_code=pm3 task_predecessor=task_begin ..
  descriptive_data='FCP$COMPILE_FORTRAN_SOURCE'
add_field field=cp_time_in_job_mode selection=task_end counter=(1, 0.000001)
add_field field=cp_time_in_monitor_mode selection=task_end counter=(2, 0.000001)
add_field field=page_faults selection=task_end counter=3
add_field field=page_ins selection=task_end counter=4
add_field field=page_reclaims selection=task_end counter=5
add_field field=page_assigns selection=task_end counter=6
add_field field=maximum_job_working_set selection=task_end counter=7
add_field field=numberm_of_slowed_down selection=task_end counter=8
put_interval_field put=task_end_interval ..
  field=((cp_time_in_job_mode,, 14, 6, 'Job', 'Cpu')..
         (cp_time_in_monitor_mode,,, 8, 'Monitor', '  Cpu  ')..
         (page_faults,,, 5, 'Page ', 'Fault')..
         (page_ins,,, 4, 'Page', 'Ins ')..
         (page_reclaims,,, 4, 'Page', 'Rec.') ..
         (page_assigns,,, 6, ' Page ', 'Assign')   ..
         (maximum_job_working_set,,, 11, 'Maximum', 'Working Set') ..
         (numberm_of_slowed_down,,, 6, 'Slowed', ' Down ') ..
         (page_assigns, interval,, 8,, 'Interval') ..
         (page_assigns, elapsed_time_since_predecessor,, 8, 'Elapsed', 'Time')) ..
  row_label=(end_time 1  12  'MS')
generate_report input=log1
```

The following report is produced:

```
                 Job  Monitor Page Page Page   Page    Maximum Slowed           Elapsed
                 Cpu      Cpu Fault  Ins Rec. Assign Working Set   Down Interval     Time
11:55:31.234    .974     .324   147   61    5    111         163      0    0.000   10.921
11:55:31.245    .633     .582   248  173   10    102         225      0     .011   18.438
11:55:43.254    .951     .175    85    4    0    111         163      0   12.009   11.917
11:56:00.732    .969     .164    88    3    4    111         165      0   17.478   20.104
11:56:03.264    .951     .168    83    3    0    110         163      0    2.532   21.428
11:56:14.028    .612     .168    82   10    0    102         150      0   10.764   21.679
11:56:28.553    .972     .196    93    5    5    111         163      0   14.525   30.269
11:56:31.685    .609     .151    80    5    1    103         150      0    3.132   22.660
11:56:38.899    .952     .170    84    3    0    110         163      0    7.214   29.287
11:56:39.333    .974     .180    86    3    0    111         164      0     .434   29.149
11:56:43.870    .954     .170    84    4    0    110         163      0    4.537   25.931
11:56:45.046    .610     .147    77    4    0    103         150      0    1.176   16.134
11:58:04.083   1.033     .686   281  180   24    112         164      0   79.037   28.758
11:58:08.171   1.021     .186    98    1   15    112         164      0    4.088    4.153
11:58:24.838   1.023     .166    83    1    0    112         164      0   16.667    9.219
11:58:46.785   1.041     .238   131    1   44    113         164      0   21.947    4.224
11:58:50.279   1.030     .172    83    1    0    112         164      0    3.494    5.424
12:02:03.776    .754     .730   267  192    5    105         173      0  193.497   11.697
12:02:15.910    .733     .168    79    4    0    105         173      0   12.134    3.830
12:02:29.288    .732     .165    79    4    0    105         173      0   13.378    3.994
12:02:35.344    .734     .163    79    4    0    105         173      0    6.056    1.939
12:03:39.016    .745     .697   258  176   11    104         173      0   63.672   22.801
12:03:40.187    .735     .189    90    5   10    105         173      0    1.171    2.272
12:03:59.503    .736     .166    79    5    0    104         173      0   19.316    9.229
12:04:06.555    .731     .160    79    5    0    104         173      0    7.052    1.678
12:04:27.590    .929     .219    97   21    0    107         161      0   21.035    2.177
12:04:37.847   1.008     .178    86    4    4    108         163      0   10.257    6.236
12:04:39.128   1.000     .174    82    4    0    108         163      0    1.281    6.784
12:04:41.228    .931     .177    82    6    0    107         161      0    2.100    7.538
12:04:49.912    .931     .176    82    6    0    107         161      0    8.684    8.180
12:04:53.322   1.006     .184    83    4    0    108         163      0    3.410   10.664
12:04:57.653    .929     .170    83    6    0    108         161      0    4.331    9.827
12:05:12.357   1.005     .172    82    4    0    108         163      0   14.704    7.826
12:05:15.810   1.001     .169    82    4    0    108         163      0    3.453    9.935
12:05:24.172   2.171     .328   129   32    0    127         195      0    8.362   13.980
12:06:20.430    .957     .256   131   53    0    107         162      0   56.258   26.132
12:06:20.656    .968     .418   207  128    6    107         161      0     .226   39.822
12:06:20.941    .961     .407   197  113    6    107         161      0     .285   38.899
12:06:24.220    .949     .165    81    4    0    107         162      0    3.279   17.531
12:06:40.121   2.166     .327   130   30    0    128         195      0   15.901   40.132
12:06:40.226   2.166     .236   102    4    0    127         196      0     .105   34.773
12:06:40.926   2.165     .275   114   15    0    127         195      0     .700   38.309
12:07:00.432   2.193     .263   114    5   10    128         195      0   19.506   38.355
12:10:33.894    .730     .629   255  173   13    104         172      0  213.462   11.749
12:10:39.480    .728     .172    80    5    0    103         172      0    5.586    2.169
12:10:48.914    .717     .162    78    5    0    103         172      0    9.434    1.098
12:10:52.413    .725     .160    78    5    0    103         172      0    3.499    3.269
12:13:56.431    .729     .380   223  151    0    102         172      0  184.018    9.488
12:13:56.509    .736     .443   284  184   32    103         172      0     .078   10.524
12:13:56.594    .722     .158    78    6    0    102         172      0     .085    3.031
12:13:56.647    .727     .337   180  108    0    102         172      0     .053    7.661
12:16:41.067   2.131     .804   366  193   88    122         194      0  164.420   10.950
12:17:25.200   2.118     .332   277    9  178    122         194      0   44.133    5.017
12:17:27.168   2.113     .207   100    9    0    123         196      0    1.968    5.613
12:17:28.932   2.109     .202    99    9    0    122         196      0    1.764    4.518
12:17:36.419   2.106     .203    99    9    0    122         196      0    7.487    2.669
12:18:11.237    .583     .280   282    7  206     99         153      0   34.818    1.113
12:18:12.898    .580     .137    75    5    1     99         153      0    1.661    1.020
12:18:14.557    .578     .137    75    5    1     99         153      0    1.659     .911
12:18:15.631    .592     .149    75    5    1     99         153      0    1.074    1.022
```

PUT_NEW_PAGE Subcommand

Purpose Specifies a forced page break in a report.

Format **PUT_NEW_PAGE** or
PUTNP
USE_PAGE_HEADER=boolean
PUT=name
NUMBER=keyword or *integer*
STATUS=status variable

Parameters *USE_PAGE_HEADER* or *USE_PAGE_HEADERS* or *UPH*

Specifies a boolean value indicating whether page headers are displayed in the report when the page break occurs. The default is TRUE.

TRUE

Page headers are displayed.

FALSE

Page headers are not displayed. This includes both user-defined page headers and default page headers. However, this does not inhibit the displaying of page headers for the remainder of the report.

PUT or *P*

Defines the name of the report entry.

NUMBER or *N*

Defines where you want the current subcommand placed within the list of output definition subcommands. You can specify an integer from 1 to $MAX_INTEGER, or you can specify the keyword NEXT (or N). The default is NEXT. This means that the new output definiition subcommand is placed after all current output definition subcommands. If you specify an integer, the new subcommand occupies that relative position. If you specify a relative position greater than the largest used ordinal, the subcommand is placed at the end of the list of subcommands.

STATUS

Returns the completion status for this subcommand.

Remarks

- This subcommand has the same characteristics as other subcommands that define PUT parameters. That is, the name of the report entry defined by the PUT parameter can be displayed and manipulated by the DISPLAY_PUT, CHANGE_PUT, and DELETE_PUT subcommands.
- This subcommand works only with the GENERATE_REPORT subcommand. It does not work with the GENERATE_LOG subcommand.

PUT_RECORD Subcommand

Purpose

Defines one or more selections whose designated statistics are to be listed in the log produced by a GENERATE_LOG subcommand.

Format

PUT_RECORD or
PUT_RECORDS or
PUTR
  SELECTION=list of name
  COUNTER=keyword or *list of record*
  DESCRIPTIVE_DATA=string or *record*
  PUT=name
  NUMBER=keyword or *integer*
  STATUS=status variable

Parameters

SELECTION or **SELECTIONS** or **S**

Specifies the name or names of the selections to be reported. The selection must have been previously defined. This parameter is required.

COUNTER or *COUNTERS* or *C*

Specifies a keyword or a list of one or more records indicating which counter or counters of the statistics in the named selection will be displayed in the report. The keyword you can specify is NONE (no counters are printed).

The record has the following format:

(*counter_number, base*)

counter_number

Specifies one or more counters whose data is to be printed. You can specify an integer from 1 to 4,095. You can also specify the keyword ALL. In this case, all counters for the named selection are printed. The default is ALL.

base

Specifies the numeric base in which the counter or counters are printed. You can specify one of the following keywords; the default is B10:

BASE_2 or B2
BASE_8 or B8
BASE_10 or B10
BASE_16 or B16
BASE_16_GROUP or B16G

DESCRIPTIVE_DATA or *DD*

Defines a string or a record indicating the descriptive data field that is to appear as output with the named selection. You must specify this parameter if you do not specify the COUNTER parameter.

You can define a string of 0 to 4,095 characters, or you can define a record. If you define a record, it has the following format:

(*position, length, field_number, field_delimiter*)

position

Defines the starting position of the descriptive data field. You can specify an integer from 1 to 4,095. The default is 1.

length

Defines the length of the substring. You can specify an integer from 1 to 4,095, or you can specify the keyword ALL. The default is ALL.

field_number

Defines the field number for the substring. You can specify an integer from 1 to 4,095, or you can specify the keyword ALL. The default is ALL.

field_delimiter

Defines a string containing the character that delimits the subfield. The default is a comma (,).

PUT or *P*

Defines the name of the report entry. The report entry must not have been previously defined.

NUMBER or *N*

Defines where you want the current subcommand placed within the list of output definition subcommands. You can specify an integer from 1 to $MAX_INTEGER, or you can specify the keyword NEXT (or N). The default is NEXT. This means that the new output definition subcommand is placed after all current output definition subcommands. If you specify an integer, the new subcommand occupies that relative position. If you specify a relative position greater than the largest used ordinal, the subcommand is placed at the end of the list of subcommands.

STATUS

Returns the completion status for this subcommand.

Remarks

- This subcommand works only with the GENERATE_LOG subcommand. It does not work with the GENERATE_REPORT subcommand.

- Three output formats are available for information specified by this subcommand. The formats are specified with the DISPLAY_FORMAT parameter of the GENERATE_LOG subcommand. The following format options are available:

  LIST or L

  User-readable format. All selection header information is displayed, as well as all counter and descriptive data fields for every occurrence of the statistic.

  LEGIBLE_DATA or LD

  Program-readable format. All selected statistics appear in a single row.

  BINARY or B

  Output format is identical to that of the original binary log. This format is intended for users who want to save a log that is smaller than the original log.

- The following example shows how the output appears when DISPLAY_FORMAT=LEGIBLE_DATA and COUNTER_FORMAT=FIXED are specified on the GENERATE_LOG subcommand. In this and the next examples, the output normally appears as a single line. Due to space limitations on this page, it is divided into more than one line.

```
OS2       1988-09-06 08:02:11.274  $08550109AAA0000  00010-001   ..
6    7              176639592              382703482 ..
             4104                          72
5784                     132 0,1,  7
```

  The format of the output line is as follows. The xx value represents to the length of the date/time string. The yy value represents the length of the fixed counter.

```
row   column
      1                      statistic code           OS2
      10                     datetime                1988-09-06 08:02:11.
274
      12 + xx                system job name          $08550109AAA0000
      33 + xx                global task id           00010-001
      44 + xx                number of counters         6
      49 + xx                length of descriptive data  7
      52 + xx                counter 1                               176639592
      52 + xx + yy           counter 2                           382703482
      52 + xx + yy * 2       counter 3                                4104

      52 + xx + yy * (n - 1) counter n                                 132
      52 + xx + yy * n  + 1  descriptive data         0,1,  7
```

- The following example shows how the output appears when DISPLAY_FORMAT=LEGIBLE_DATA and COUNTER_FORMAT=VARIABLE are specified on the GENERATE_LOG subcommand. In this type of format, the counter values are separated by only a single character.

```
OS2      1988-09-06 08:02:11.274  $08550109AAA0000  00010-001
6    7 176639592 382703482 4104 72 5784 132 0,1,  7
```

- The following example shows how the output appears when DISPLAY_FORMAT=LIST and COUNTER_FORMAT=FIXED are specified on the GENERATE_LOG subcommand:

```
OS2       1988-09-06 08:02:11.274  $08550109AAA0000  00010-001    6    7
  1:            176639592           382703482               4104
  4:                   72                5784                132
'0,1,  7'
```

The format of the output line is as follows. The xx value represents to the length of the date/time string. The yy value represents the length of the fixed counter.

```
row   column
1      2                      statistic code             OS2
       11                     datetime                  1988-09-06 08:02:11.274
       13 + xx                system job name            $08550109AAA0000
       34 + xx                global task id             00010-001
       45 + xx                number of counters           6
       50 + xx                length of descriptive data   7
2      6                      counter 1                                176639592
       6 + yy                 counter 2                                382703482
       6 + yy * 2             counter 3                                     4104
3      6                      counter 4                                       72
       6 + yy                 counter 5                                     5784
       6 + yy * 2             counter 6                                      132
n      2                      descriptive data           '0,1,  7'
```

The number of counters displayed on each line is calculated as follows:

```
(page_width-5) / counter_length
```

- The base field in the COUNTER parameter for the PUT_RECORD subcommand is meaningless if the DATA_FORMAT parameter for the GENERATE_LOG subcommand is set to BINARY.

Examples The following example defines the TASK_BEGIN and TASK_END selections for analyzing the PM0 and PM3 statistics. The contents of the analysis of these statistics are then listed in a log.

```
add_selection selection=task_begin statistic_code=pm0 ..
  time=08:00:00 .. 16:00:00
add_selection selection=task_end statistic_code=pm3 ..
  task_predecessor=task_begin descriptive_data='FCP$COMPILE_FORTRAN_SOURCE'
put_record selection=task_begin
put_record selection=task_end
create_file file=$local.1
set_file_attribute file=$local.1 page_width=80
generate_log input=log1 output=$local.1 display_format=list ..
  counter_format=fixed
copy_file input=$local.1
```

The log appears as follows:

```
ANALYZE_BINARY_LOG                              07/01/88           PAGE  1
NOS/VE  1.4.1  10R                              11:09:20

Log scanned: :BGU.CLBG100.ANABL.LOG1

PM0       1988-05-17 11:51:17.440  $0830_0604_AAA_0000  00169-001    0    0

PM0       1988-05-17 11:52:42.829  $0830_0604_AAA_0000  00174-001    0    0

PM0       1988-05-17 11:55:12.807  $0830_0604_AAA_0032  00216-001    0    0

PM0       1988-05-17 11:55:20.313  $0830_0604_AAA_0033  00217-001    0    0

PM3       1988-05-17 11:55:31.234  $0830_0604_AAA_0033  00217-001    8   31
  1:                 974492               324039               147
  4:                     61                    5               111
  7:                    163                    0
'FCP$COMPILE_FORTRAN_SOURCE      '

PM3       1988-05-17 11:55:31.245  $0830_0604_AAA_0032  00216-001    8   31
  1:                 633200               581778               248
  4:                    173                   10               102
  7:                    225                    0
'FCP$COMPILE_FORTRAN_SOURCE      '

PM0       1988-05-17 11:55:31.337  $0830_0604_AAA_0035  00218-001    0    0

PM0       1988-05-17 11:55:40.628  $0830_0604_AAA_0043  00219-001    0    0

PM0       1988-05-17 11:55:41.836  $0830_0604_AAA_0045  00220-001    0    0

PM3       1988-05-17 11:55:43.254  $0830_0604_AAA_0035  00218-001    8   31
  1:                 950628               175054                85
  4:                      4                    0               111
  7:                    163                    0
'FCP$COMPILE_FORTRAN_SOURCE      '

PM0       1988-05-17 11:55:43.625  $0830_0604_AAA_0033  00221-001    0    0

PM0       1988-05-17 11:55:43.677  $0830_0604_AAA_0032  00222-001    0    0

PM0       1988-05-17 11:55:52.349  $0830_0604_AAA_0042  00223-001    0    0

PM0       1988-05-17 11:55:55.951  $0830_0604_AAA_0035  00224-001    0    0

PM0       1988-05-17 11:55:58.284  $0830_0604_AAA_0054  00225-001    0    0

PM3       1988-05-17 11:56:00.732  $0830_0604_AAA_0043  00219-001    8   31
  1:                 969106               164191                88
  4:                      3                    4               111
  7:                    165                    0
'FCP$COMPILE_FORTRAN_SOURCE      '

PM3       1988-05-17 11:56:03.264  $0830_0604_AAA_0045  00220-001    8   31
  1:                 951372               168383                83
  4:                      3                    0               110
  7:                    163                    0
'FCP$COMPILE_FORTRAN_SOURCE      '
```

PUT_STRING Subcommand

Purpose Defines strings of text to be inserted into a report.

Format **PUT_STRING** or
PUTS
STRING=list of record
PUT=name
NUMBER=keyword or *integer*
STATUS=status variable

Parameters **STRING** or **STRINGS** or **STR**

Defines a list of 1 to 11 string records to be inserted into a report. This parameter is required. The record has the following format:

(**string**, *start_column, column_width*)

string

Defines the string you want inserted in the report. You must specify 1 to 10,000 characters. This entry is required.

start_column

Defines the column in which the string begins. You can specify an integer from 1 to 10,000. You must define starting columns in such a way that columns do not overlap. The default for this field is the last position in the previous field plus 2. If there is no row label, the default start column for the first field is 1. If there is a row label, the default start column for the first field is the last position in the row label plus 1.

column_width

Defines the width of the column in which the string begins. You can specify an integer from 1 to 10,000. You must define column widths in such a way that columns do not overlap.

PUT or *P*

Defines the name of the report entry.

NUMBER or *N*

Defines where you want the current subcommand placed within the list of output definition subcommands. You can specify an integer from 1 to $MAX_INTEGER, or you can specify the keyword NEXT (or N). The default is NEXT. This means that the new output definiition subcommand is placed after all current output definition subcommands. If you specify an integer, the new subcommand occupies that relative position. If you specify a relative position greater than the largest used ordinal, the subcommand is placed at the end of the list of subcommands.

STATUS

Returns the completion status for this subcommand.

Remarks This subcommand works only with the GENERATE_REPORT subcommand. It does not work with the GENERATE_LOG subcommand.

Examples The following example defines a string that will be inserted in a report starting at column 45:

```
put_string put=report_header ..
  string=(('Daily Summary Report' 45))
generate_report input=log1
```

The following string appears in the report:

```
    Daily Summary Report
```

QUIT Subcommand

Purpose	Terminates the ANALYZE_BINARY_LOG utility.
Format	**QUIT** or **QUI** *STATUS=status variable*
Parameters	*STATUS* Returns the completion status for this subcommand.

Examples of Using Statistics

The following examples demonstrate useful activities that a performance analyst should know how to perform using the ANALYZE_BINARY_LOG utility. For all examples shown, you must activate the statistics that are emitted by NOS/VE. In addition, you must define an emission set for any periodic statistics (OS statistics and most NA statistics) you want analyzed. The first example shows how to do this. Most of the remaining examples illustrate how to produce certain kinds of reports.

Keeping Track of User Jobs with a Periodic Statistic

You can use the OS0 statistic to keep track of the number of user jobs in the system throughout the day. This periodic statistic has the following counters, among others:

Counter	Description
11	Records the number of interactive jobs.
12	Records the number of noninteractive jobs.

To use this statistic, follow these steps:

1. Because it is a periodic statistic, define how often you want it to be collected.
2. Activate the statistic at the start of system operation.
3. Terminate the binary log, thus copying it to a permanent file.
4. Direct the ANALYZE_BINARY_LOG utility to analyze the binary log and to produce a report showing the number of user jobs throughout the day.

Each step consists of the following:

1. At the start of system operations, use the MANAGE_PERIODIC_STATISTICS utility to define how often you want the statistic emitted. To define a period (interval) of 5 minutes, enter the following commands within a System Operator Utility session:

```
manage_periodic_statistics reserve_write_privilege=true
   add_statistic emission_set=s1 statistic=os0
   change_period emission_set=s1 period=5
   change_state emission_set=s1 state=enabled
   quit write_emission_sets=true
```

2. Activate the statistic by entering the following command within a System Operator Utility session:

```
activate_system_statistic statistic=os0
```

   This causes the OS0 statistic to be written to the global statistics log each time the statistic is emitted. (The global statistic log is the default for the LOG parameter of the ACTIVATE_SYSTEM_STATISTICS command. The parameters of this command are described earlier in this chapter.)

3. At the end of system operations, terminate the global statistics log to a permanent file that can be accessed from normal interactive jobs. To do this, execute the following command within a System Operator Utility session:

   ```
   terminate_log statistic :nve.sys_logs.stat_log.$next
   ```

   This terminates the statistics log and copies it to the permanent file you specify (represented in this example by :NVE.SYS_LOGS.STAT_LOG). We recommend you specify a file that is easily accessible to you (and only to you) for analysis at a terminal.

4. The following sample session generates a report indicating how many active and noninteractive user jobs were executed:

   ```
   anabl
     add_selection selection=job_memory_interval statistic_code=os0
     add_field field=interative_job_count selection=job_memory_interval ..
       counter=11
     add_field field=noninterative_job_count selection=job_memory_interval ..
       counter=12
     put_interval_field put=display_inter_job_count ..
       field=((interative_job_count,,20,20,'Number of','Interactive Jobs') ..
       (noninterative_job_count,,45,20,'Number of','Noninteractive Jobs')) ..
       row_label=(end_time, 1,15)
     generate_report input=$user.stats_log output=my_report
     quit
   ```

Producing Task/Program Usage Reports for Users

The next five examples show how, with the aid of the ANALYZE_BINARY_LOG utility, you can produce resource usage reports for user tasks.

As a precursor to each example, you need to perform the following activities:

1. Activate the PM3 (task end) statistic.
2. Run your tasks.
3. Deactivate the PM3 statistic.
4. Copy the job statistic log to a permanent file.

The following commands reflect these activities:

```
activate_job_statistics statistic=pm3
  .
  .  (Run tasks here)
  .
deactivate_job_statistics statistic=pm3
copy_file $local.$job_statistic_log $user.copy_of_log
```

Quick Report of Data for Each Occurrence of a Selected Record

The following example generates a log, in LIST format, for the TASK_END selection:

```
anabl
  add_selection selection=task_end statistic_code=pm3
  put_record selection=task_end
  generate_log input=$user.copy_of_log output=putr1_out display_format=list
  quit
```

The log appears as follows:

```
ANALYZE_BINARY_LOG  NOS/VE  1.4.1  10R  08/01/89 12:45:12      PAGE 1


Log scanned: :NVE3.JRM.ANABL_EXAMPLES.SAMPLE_LOG

PM3       1989-07-29.15:51:34  $0855_0109_AAP_5042  00313-029    9   31
  1: 2389362 609237 563 217 259 270 360 0 0
'PTP$DISPLAY_PROCEDURE_CALLS    '

PM3       1989-07-29.15:51:58  $0855_0109_AAP_5042  00429-028    9   31
  1: 2386008 566546 704 0 435 266 305 0 0
'PTP$DISPLAY_PROCEDURE_CALLS    '

PM3       1989-07-29.15:52:17  $0855_0109_AAP_5042  00499-030    9   31
  1: 2168544 533401 687 0 420 265 303 0 0
'PTP$DISPLAY_PROCEDURE_CALLS    '
```

Report Showing Each Occurrence of a Selected Record

The following example produces a simple summary that is easier to read than the output produced by the GENERATE_LOG subcommand. In addition, it provides more format control through the use of the PUT_INTERVAL_FIELD subcommand.

```
anabl
  add_selection selection=task_end statistic_code=pm3
  add_field selection=task_end field=te_jmcp_sec counter=(1, 1.0e-6)
  add_field selection=task_end field=te_mmcp_sec counter=(2, 1.0e-6)
  add_field selection=task_end field=te_name descriptive_data=(1,31)
  put_interval_field field=((te_name,,,,'Starting Procedure'),..
    (te_jmcp_sec,sum,,,'JM CP'), ..
    (te_mmcp_sec,sum,,,'MM CP'))     row_label=none
  generate_report input=sample_log output=putif2_out ..
    display_format=list
  quit
```

The output appears as follows:

```
ANALYZE_BINARY_LOG   NOS/VE  1.5.1  10R   08/01/89  13:47:43     PAGE 1

              Starting Procedure            JM CP           MM CP
 PTP$DISPLAY_PROCEDURE_CALLS                2.389            .609
 PTP$DISPLAY_PROCEDURE_CALLS                2.386            .567
 PTP$DISPLAY_PROCEDURE_CALLS                2.169            .533
```

Data Summary: Format 1

The following report summarizes the data rather than reporting each instance of the record:

```
anabl
  add_selection selection=task_end statistic_code=pm3
  add_field selection=task_end field=te_jmcp_sec counter=(1, 1.0e-6)
  add_field selection=task_end field=te_mmcp_sec counter=(2, 1.0e-6)
  put_field_summary field=(te_jmcp_sec,te_mmcp_sec) ..
    display_option=((count),(sum),(mean)) display_headers=true
  generate_report input=sample_log output=putfs3_out display_format=list
  quit
```

The output appears as follows:

```
ANALYZE_BINARY_LOG  NOS/VE  1.5.1  10R  08/01/89   08:42:37     PAGE  1

                                 Count            Sum            Mean
TE_JMCP_SEC                          3          6.944           2.315
TE_MMCP_SEC                          3          1.709            .570
```

Data Summary: Format 2

The following report reflects the format normally followed by a DBMS report writer. In contrast to the preceding example, each instance of the record is reported.

```
anabl
  add_selection selection=task_end statistic_code=pm3
  add_field selection=task_end field=te_jmcp_sec counter=(1, 1.0e-6)
  add_field selection=task_end field=te_mmcp_sec counter=(2, 1.0e-6)
  add_selection selection=labels
  add_field selection=labels field=count string='Count'
  add_field selection=labels field=sum string='Sum'
  add_field selection=labels field=mean string='Mean'
  put_string string=(('job cp',12,10),('mtr cp',24,10))
  put_string string=(('------',12,10),('------',24,10))
  put_field field=((count,,3,8),(te_jmcp_sec,count,12,10),..
    (te_mmcp_sec,count,24,10))
  put_field field=((sum,,3,8),(te_jmcp_sec,sum,12,10),..
    (te_mmcp_sec,sum,24,10))
  put_field field=((mean,,3,8),(te_jmcp_sec,mean,12,10),..
    (te_mmcp_sec,mean,24,10))
  generate_report input=sample_log output=putf3_out display_format=list
  quit
```

The output appears as follows:

```
ANALYZE_BINARY_LOG  NOS/VE  1.5.1  10R   08/01/89  09:04:36     PAGE  1

              job cp      mtr cp
              ------      ------
  Count            3           3
  Sum          6.944       1.709
  Mean         2.315        .570
```

Report Showing Both Detail and Summary Data

Since it provides both detail and summary information, the following example is a combination of the two previous examples:

```
anabl
  add_selection selection=task_end statistic_code=pm3
  add_field selection=task_end field=te_jmcp_sec counter=(1, 1.0e-6)
  add_field selection=task_end field=te_mmcp_sec counter=(2, 1.0e-6)
  add_field selection=task_end field=te_name descriptive_data=(1,31)
  add_selection selection=labels
  add_field selection=labels field=count string='Count'
  add_field selection=labels field=sum string='Sum'
  add_field selection=labels field=mean string='Mean'
  put_string string=' '
  put_string string='Task End Summary for Job'
  put_string string=' '
  put_string string=(('Start Procedure',3,31),..
    ('job cp',36,10),('mtr cp',48,10))
  put_string string=(('-------------------------------',..
    3,31),('------',36,10),('------',48,10))
  put_interval_field field=((te_name,,3,31),(te_jmcp_sec,sum,..
    36,10),(te_mmcp_sec,sum,48,10)) row_label=none
  put_string string=(('-------------------------------',3,31),..
    ('------',36,10),('------',48,10))
  put_field field=((sum,,3,31),(te_jmcp_sec,sum,36,10),..
    (te_mmcp_sec,sum,48,10))
  put_field field=((count,,3,31),(te_jmcp_sec,count,..
    36,10),(te_mmcp_sec,count,48,10))
  put_field field=((mean,,3,31),(te_jmcp_sec,mean,..
    36,10),(te_mmcp_sec,mean,48,10))
  generate_report input=sample_log output=putif3_out display_format=list
  quit
```

The report appears as follows:

```
ANALYZE_BINARY_LOG  NOS/VE  1.5.1  10R  08/01/89  09:52:30     PAGE 1


  Task End Summary for Job

                   Start Procedure      job cp      mtr cp
   -------------------------------      ------      ------
   PTP$DISPLAY_PROCEDURE_CALLS           2.389        .609
   PTP$DISPLAY_PROCEDURE_CALLS           2.386        .567
   PTP$DISPLAY_PROCEDURE_CALLS           2.169        .533
   -------------------------------      ------      ------
   Sum                                   6.944       1.709
   Count                                     3           3
   Mean                                  2.315        .570
```

Producing a Site Analyst Task Summary

A site analyst task summary can be obtained by activating the task end statistic (PM3) from the system job and producing a report using the ANALYZE_BINARY_LOG utility. The report is simply a copy of the user task end summary reports that are described on the preceding pages. A more complex example appears in the Site Analyst Examples online manual.

Producing a Site Analyst Job Summary

A site analyst job summary can be obtained by activating the job end statistic (JM3) from the system job and producing a report using the ANALYZE_BINARY_LOG utility. The report is simply a copy of the user task end summary reports that are described on the preceding pages except that references to the PM3 statistic code must be changed to JM3.

Producing Formatted Data as Input to Excel Spreadsheets

To generate output that can be used as input to an Excel spreadsheet program on a microcomputer, you can use a modified version of the second type of task usage report shown in the preceding section (Report Showing Each Occurrence of a Selected Record). The only change is that the DISPLAY_FORMAT parameter of the GENERATE_RECORD subcommand is set to EXCEL.

```
anabl
  add_selection selection=task_end statistic_code=pm3
  add_field selection=task_end field=te_jmcp_sec counter=(1, 1.0e-6)
  add_field selection=task_end field=te_mmcp_sec counter=(2, 1.0e-6)
  add_field selection=task_end field=te_name descriptive_data=(1,31)
  put_interval_field field=((te_name,,,,'Starting Procedure'),..
    (te_jmcp_sec,sum,,,'JM CP'), ..
    (te_mmcp_sec,sum,,,'MM CP')) row_label=none
  generate_report input=sample_log output=putif2_out ..
    display_format=excel
  quit
```

The output appears as follows. The field separators are actually tab characters (ASCII 09), not spaces as shown here. This causes each field in a row to appear in a separate cell of the spreadsheet program. Each row in this output is placed in a separate row in the spreadsheet program.

```
PTP$DISPLAY_PROCEDURE_CALLS 2.389 .609
PTP$DISPLAY_PROCEDURE_CALLS 2.386 .567
PTP$DISPLAY_PROCEDURE_CALLS 2.169 .533
```

Producing a Report of Periodic Statistics

Periodic statistics consist of operating system (OS) and NAM/VE (NA) statistics. The following example collects information on the use of the OS6 (CPU usage) statistic. Information found in counters 1 and 2 from 8 a.m. to 5 p.m. is collected. The values in the columns are the result of subtracting the values of consecutive occurrences of the counters.

```
anabl
  add_selection selection=cp_use statistic_code=os6 ..
    time=08:00:00..17:00:00
  add_field selection=cp_use field=job_mode_cp counter=(1, 1.0e-6, true)
  add_field selection=cp_use field=monitor_mode_cp counter=(2, 1.0e-6, true)
  put_interval_field field=((job_mode_cp,,,, 'Job Mode CP'),..
    (monitor_mode_cp,,,, 'Monitor Mode CP')) ..
    row_label=time_range
  generate_report input=input_log output=output_log
  quit
```

The report appears as follows:

```
ANALYZE_BINARY_LOG  NOS/VE  1.5.1  10R  08/09/89  12:10:37     PAGE 1
                                      Job Mode CP Monitor Mode CP
08:01:55 .. 08:06:55                      183.626          20.894
08:06:55 .. 08:11:55                      251.971          16.881
08:11:55 .. 08:16:55                      254.123          16.526
08:16:55 .. 08:21:55                      216.985          16.731
08:21:55 .. 08:26:55                      161.975          22.057
08:26:55 .. 08:31:55                         .214           9.767
08:31:55 .. 08:36:55                         .085          19.073
08:36:55 .. 08:41:54                      138.035          37.252
08:41:54 .. 08:46:55                      104.668          61.735
08:46:55 .. 08:51:55                       43.546          77.921
08:51:55 .. 08:56:54                        4.808          25.371
  .
  .
  .
16:31:55 .. 16:36:55                      109.863          51.101
16:36:55 .. 16:41:55                       80.692          33.934
16:41:55 .. 16:46:54                       13.457         108.452
16:46:54 .. 16:51:55                       47.189          67.371
16:51:55 .. 16:56:55                       94.250          49.442
```

Converting a Binary Log to ASCII

The following example converts the information in the ALL_RECORDS selection to ASCII format:

```
anabl
  add_selection selection=all_records
  put_record selection=all_records
  generate_log input=input_log output=output_log ..
    display_format=legible_data
  quit
```

Reducing a Binary Log

The following example produces a new binary log that is a subset of the original log:

```
anabl
  add_selection selection=job_end statistic_code=jm3
  add_selection selection=disk_stat statistic_code=os4
  put_record selection=(job_end,disk_stat)
  generate_log input=input_log output=output_log ..
    display_format=binary
  quit
```

Merging Binary Logs

The following example merges the JM3 and OS4 statistics from both the INPUT_LOG1 and INPUT_LOG2 logs into one log. The name of the output log is OUTPUT_LOG. All other statistics are ignored.

```
anabl
  add_selection selection=job_end statistic_code=jm3
  add_selection selection=disk_stat statistic_code=os4
  put_record selection=(job_end,disk_stat)
  generate_log input=(input_log1,input_log2) ..
    output=output_log display_format=binary
  quit
```

Comprehensive Online Example

A comprehensive example showing the use of the ANALYZE_BINARY_LOG utility appears in the Site Analyst Examples online manual.

Statistic Formats

The next sections describe the formats, descriptive data fields, and counters for these groups of statistics:

Statistic Identifier	Statistic Group
AV	Accounting
CB	COBOL compilation
CL	Command language
ES	NOS/VE editor
FC	FORTRAN Version 1 compilation
FV	FORTRAN Version 2 compilation
JM	Job management
LG	Logging
MV	Mail/VE Version 1 and Version 2
NA	NAM/VE
OD	Optical disk
OS	Operating system
PM	Program management
SF	NOS/VE security audit

The configuration management (CM) statistics are also described, although (except for CM5000) only in summary form. The complete descriptions of the CM statistics appear in a separate document. See Configuration Management Statistics later in this chapter.

Accounting Statistics

The following accounting statistics are available (these are event statistics):

Statistic Name	Description
AV6	Records when accounting began. This is a descriptive data field.
AV7	Records the total resources used by a job.
AV9	Records when an application began. This is a descriptive data field.
AV10	Records the total resources used by an application.
AV11	Records the unit measured activity for an application.
AV12	Marks the start of a PROCESS_STORAGE command run.
AV13	Records storage space data for each catalog.
AV14	Records storage space data for each permanent file cycle.
AV15	Records total catalog storage space data for each project member. A project member is a unique combination of family name, user name, account name, and project name.
AV16	Records total permanent file cycle storage space data for each project member.
AV17	Records total catalog storage space data for each user.
AV18	Records total permanent file cycle storage space data for each user.
AV19	Marks the end of a PROCESS_STORAGE command run.
AV20	Records failure occurrences of the ANALYZE_JOB_ACCOUNT_LOG command to update the resource usage file when it is called from the system epilog.
AV21	Records occurrences of interactive jobs being detached.
AV22	Records occurrences of detached interactive jobs being attached.
AV23	Records occurrences of the failure of an application to record application accounting information into a log due to a system type error.
AV24	Records total catalog storage space data for each account member. An account member is a unique combination of family name, user name, and account name.

Statistic Name	Description
AV25	Records total permanent file cycle storage space data for each account member.
AV26	Records occurrences of invalid login attempts.
AV27	Records the start of a job from a remote batch station.
AV28	Records the completion of printing of an output file.
AV29	Records the length of time an output file is queued.
AV30	Records a file queued for printing by a user.
AV31	Records a job queued for submission by a user.
AV32	Records a standard output file queued for printing.
AV33	Records completion of a permanent file transfer on the requesting mainframe.
AV34	Records completion of a permanent file transfer on the target mainframe.
AV35	Records completion of a queued file transfer on the originating mainframe.
AV36	Records completion of a queued file transfer on the destination mainframe.
AV37	Records connect time and bytes transferred for an interactive terminal session.

The statistics AV6, AV7, AV9, AV10, AV11, and AV37 (begin account, end account, begin application, end application, record unit measured activity, and record connect time and bytes transferred) are activated automatically as part of the NOS/VE deadstart process. The remaining accounting statistics must be enabled using the ACTIVATE_SYSTEM_STATISTICS or ACTIVATE_JOB_STATISTICS command before NOS/VE can begin logging information.

NOTE

Since statistics AV13 and AV14 (storage space data for each catalog and permanent file cycle) record large amounts of information that can fill up the binary logs, we recommend not activating these statistics unless you have a special need to account for catalogs and files on an individual basis.

The accounting statistics have the following descriptive data fields and counters defined. If a descriptive data field is not specified, none exists.

Statistic Name	Definition and Counter Descriptions
AV6	Begin accounting. This statistic is automatically enabled during NOS/VE deadstart. For each user who logs into the system, this statistic is issued to the accounting binary log as the user is being validated and before the system prolog is executed. This statistic does not have any defined counters. The descriptive data field has the following format:

```
family,user,account,project,
user job name,job class name
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
family	Name of the family under which the job was executing.
user	User's user name.
account	Name of the account under which the job was executing.
project	Name of the project under which the job was executing.
user job name	User-supplied name on the JOB, LOGIN or SUBMIT_JOB command.
job class name	Name of the job class in which the job was executing.

Statistic Name	Definition and Counter Descriptions
AV7	End accounting. This statistic is automatically enabled during NOS/VE deadstart. This statistic is issued to the accounting binary log before the system epilog is executed as part of user logout. The following counters are defined for this statistic:

Counter	Description
1	Total job SRU count.
2	Amount of CPU time, expressed in microseconds, spent in job mode.
3	Amount of CPU time, expressed in microseconds, spent in monitor mode.
4	Total number of page faults for the job.
5	Number of pages read in from disk.
6	Number of pages reclaimed for the job.
7	Number of new pages created for the job.
8	Total number of pages from the File Server.
9	Size, in pages, of the maximum working set used.

AV9

Records the beginning of an application. This statistic is automatically enabled during NOS/VE deadstart. There are no counters defined for this statistic. The descriptive data field has the following format:

```
application ID,nested application ID
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
application ID	Application identifier.
nested application ID	Concatenation of the ID of the calling application, a pound sign, and the ID of the called (nested) application. For example, if application ABC called application NES, the nested application ID would be ABC#NES. If no application nesting took place, the nested application ID is the same as the application ID.

Statistic Name	Definition and Counter Descriptions
AV10	Records the end of an application. This statistic is automatically enabled during NOS/VE deadstart. The descriptive data field has the following format:

```
application ID,nested application ID
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

The application ID and nested application ID fields are described under statistic AV9.

The following counters are defined for this statistic.

Counter	Description
1	Total SRUs used by the application.
2	Amount of CPU time, expressed in microseconds, spent in job mode.
3	Amount of CPU time, expressed in microseconds, spent in monitor mode.
4	Total number of page faults for the application.
5	Number of pages read in from disk.
6	Number of pages reclaimed for the application.
7	Number of new pages created for the application.

Statistic Name	Definition and Counter Descriptions
AV11	Records the unit-measured activity for the application. This statistic is automatically enabled during NOS/VE deadstart. The counters defined for this statistic vary with each application. The descriptive data field has the following format:

```
application ID,nested application ID
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
application ID	Application identifier.
nested application ID	Concatenation of the ID of the calling application, a pound sign, and the ID of the called (nested) application. For example, if application ABC called application NES, the nested application ID would be ABC#NES. If no application nesting took place, the nested application ID is the same as the application ID.

Statistic Name	Definition and Counter Descriptions
AV12	Marks the start of a PROCESS_STORAGE command run. There are no descriptive data fields or counters.
AV13	Records storage space data for each catalog. The descriptive data field has the following format:

```
family,user,account,project,catalog
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
family	Family of the storage devices containing the catalog.
user	User name under which the catalog resides.
account	Account to which catalog space is charged.
project	Project to which catalog space is charged.
catalog	Catalog name.

Statistic Name	Definition and Counter Descriptions
AV13 (Continued)	The following counters are defined for this statistic:

Counter	Description
1	Catalog size in bytes.
2	Number of files in the catalog.
3	Number of subcatalogs in the catalog.
4	Number of levels in the catalog hierarchy.
5	Allocated size of the catalog in bytes.
6	Reserved.
7	Reserved.

AV14

Records storage space data for each permanent file cycle. The descriptive data field has the following format:

```
family,user,account,project, ,file
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position. Since the file field entry has a cycle number, this entry can be 35 characters long.

Field	Description
family	Family of storage devices containing the file.
user	User name under which the file resides.
account	Account to which the file space is charged.
project	Project to which the file space is charged.
file	File name and cycle number formatted as: `file.cycle.`

Statistic Name	Definition and Counter Descriptions
AV14 (Continued)	The following counters are defined for this statistic:

Counter	Description
1	File cycle size in bytes.
2	Reserved.
3	File cycle's level in the catalog hierarchy; 1 is the master catalog.
4	Allocated size of the file cycle in bytes.
5	Size of a single archived copy of the file cycle in bytes.
6	Total size of all archived copies of the file cycle in bytes.

AV15

Records total catalog storage space data for each project member. A project member is a unique combination of family name, user name, account, and project. The descriptive data field has the following format:

```
family,user,account,project,
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
family	Family of storage devices containing the catalogs.
user	User name under which the catalogs reside.
account	Account to which catalog space is charged.
project	Project to which the catalog space is charged.

The following counters are defined for this statistic:

Counter	Description
1	Total size of project member catalogs in bytes.
2	Number of project member catalogs.
3	Total allocated size of project member catalogs in bytes.
4	Reserved.
5	Reserved.

Statistic Name	Definition and Counter Descriptions
AV16	Records total permanent file cycle storage space data for each project member. A project member is a unique combination of family name, user name, account, and project. The descriptive data field has the following format:

```
family,user,account,project,
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
family	Family of storage devices containing the file cycles.
user	User name under which the file cycles reside.
account	Account to which file cycle space is charged.
project	Project to which file cycle space is charged.

The following counters are defined for this statistic:

Counter	Description
1	Total size of project member file cycles in bytes.
2	Number of project member file cycles.
3	Total allocated size of project member file cycles in bytes.
4	Total size of archived copies of project member file cycles in bytes (excluding duplicate copies).
5	Total size of archived copies of project member file cycles in bytes (including duplicate copies).

Statistic Name	Definition and Counter Descriptions
AV17	Records total catalog storage space data for each user. The descriptive data field has the following format:

```
family,user, , ,
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
family	Family of storage devices containing the catalogs.
user	User name under which the catalogs reside.

The following counters are defined for this statistic:

Counter	Description
1	Total size of user catalogs in bytes.
2	Number of user catalogs.
3	Total allocated size of user catalogs in bytes.
4	Reserved.
5	Reserved.

Statistic Name	Definition and Counter Descriptions
AV18	Records total permanent file cycle storage space data for each user. The descriptive data field has the following format:

```
family,user, , ,
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
family	Family of storage devices containing the file cycles.
user	User name under which the file cycles reside.

The following counters are defined for this statistic:

Counter	Description
1	Total size of user file cycles in bytes.
2	Number of user file cycles.
3	Total allocated size of user file cycles in bytes.
4	Total size of archived copies of user file cycles in bytes (excluding duplicate copies).
5	Total size of archived copies of user file cycles in bytes (including duplicate copies).

Statistic Name	Definition and Counter Descriptions
AV19	Marks the end of a PROCESS_STORAGE command run. There are no descriptive data fields or counters.
AV20	Records failure occurrences of the ANALYZE_JOB_ACCOUNT_LOG command to update the resource usage file when it is called from the system epilog. The descriptive data field has the following format:

```
status
```

Field	Description
status	Error status parameters that can be used to reconstruct the error status occurring within ANALYZE_JOB_ACCOUNT_LOG. The maximum length of this field is 255 characters.

The following counter is defined for this statistic:

Counter	Description
1	Condition code of the error that occurred within ANALYZE_JOB_ACCOUNT_LOG.

AV21	Records occurrences of interactive jobs being detached. There are no descriptive data fields or counters.
AV22	Records occurrences of detached interactive jobs being attached. There are no descriptive data fields or counters.

Statistic Name	Definition and Counter Descriptions
AV23	Records occurrences of the failure of an application to record application accounting information into a log due to a system type error. The descriptive data field has the following format:

```
status
```

Field	Description
status	Error status parameters that can be used to reconstruct the error status occurring within the application.

The following counter is defined for this statistic:

Counter	Description
1	Condition code of the error that occurred within the application.

AV24

Records total catalog storage space data for each account member. An account member is a unique combination of family name, user name, and account name. The descriptive data field has the following format:

```
family,user,account,
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
family	Family of storage devices containing the catalogs.
user	User name under which the catalogs reside.
account	Account to which catalog space is charged.

The following counters are defined for this statistic:

Counter	Description
1	Total size of account member catalogs in bytes.
2	Number of account member catalogs.
3	Total allocated size of account member catalogs in bytes.
4	Reserved.
5	Reserved.

Statistic Name	Definition and Counter Descriptions
AV25	Records total permanent file cycle storage space data for each account member. The descriptive data field has the following format:

```
family,user,account,
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
family	Family of storage devices containing the file cycles.
user	User name under which the file cycles reside.
account	Account to which file cycle space is charged.

The following counters are defined for this statistic:

Counter	Description
1	Total size of account member file cycles in bytes.
2	Number of account member file cycles.
3	Total allocated size of account member file cycles in bytes.
4	Total size of archived copies of account member file cycles in bytes (excluding duplicate copies).
5	Total size of archived copies of account member file cycles in bytes (including duplicate copies).

Statistic Name	Definition and Counter Descriptions
AV26	Records occurrences of invalid login attempts. The descriptive data field has the following format:

```
family,user,terminal_name
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

An attempt to log in is invalid in these situations:

- You specify an incorrect user name.
- You specify an incorrect password.
- Your password has expired.

Field	Description
family	Family to which the login attempt was made.
user	User name under which the login attempt was made.
terminal_name	Name of the terminal at which the login attempt was made.

There are no counters for this statistic.

Statistic Name	Definition and Counter Descriptions
AV27	Records the start of a job from a remote batch station. The descriptive data field has the following format:

```
family,user,account,project,system job name,
user job name,DI system name,line name,line subtype,
line speed,station,device
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
family	Family name associated with the user who submitted the job.
user	User name of the user who submitted the job.
account	User's account name.
project	User's project name.
system job name	System-supplied name of the job.
user job name	User-supplied name of the job.
DI system name	Name of the DI from which the user is communicating.
line name	Name of the line over which the user is communicating.
line subtype	Subtype of the line over which the user is communicating.
line speed	Speed of the line over which the user is communicating.
station	Name of the station from which the user is communicating.
device	Name of the device from which the user is communicating.

Statistic Name	Definition and Counter Descriptions
AV27 (Continued)	The following counters are defined for this statistic:

Counter	Description
1	Size of the input file in bytes.
2	Connect time, expressed in seconds.
3	Number of lines.

AV28

Records the completion of printing of an output file. The descriptive data field has the following format:

```
family,user,account,project,system job name,
user file name,DI system name,line name,line subtype,
line speed,station,device,output class,forms code
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
family	Family name associated with the user who printed the file.
user	User name of the user who printed the file.
account	User's account name.
project	User's project name.
system job name	System-supplied name of the job that printed the file.
user file name	User-supplied name of the file.
DI system name	Name of the DI from which the file is printed.
line name	Name of the line over which the file is printed.
line subtype	Subtype of the line over which the file is printed.
line speed	Speed of the line over which the file is printed.

Statistic Name	Definition and Counter Descriptions

AV28 (Continued)

Counter	Description
station	Name of the station on which the file is printed.
device	Name of the device on which the file is printed.
output class	Class of the output file.
forms code	Forms code of the output file.

The following counters are defined for this statistic:

Counter	Description
1	Size of the output file in bytes.
2	Connect time in seconds.
3	Number of lines printed.

AV29

Records the length of time an output file is queued. The descriptive data field has the following format:

```
family,user,account,project,system job name,
user file name
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
family	Family name associated with the user who printed the file.
user	User name of the user who printed the file.
account	User's account name.
project	User's project name.
system job name	System-supplied name of the job that printed the file.
user file name	User-supplied name of the file.

Statistic Name	Definition and Counter Descriptions
AV29 (Continued)	The following counters are defined for this statistic:

Counter	Description
1	Size of the output file in bytes.
2	Time in queue, expressed in seconds.

Statistic Name	Definition and Counter Descriptions
AV30	Records a file queued for printing by a user. The descriptive data field has the following format:

```
user file name
```

Field	Description
user file name	User-supplied name (up to 31 characters) of the file that was queued for printing.

The following counter is defined for this statistic:

Counter	Description
1	Size of the output file in bytes.

Statistic Name	Definition and Counter Descriptions
AV31	Records a file queued for submission by a user. The descriptive data field has the following format:

```
system job name
```

Field	Description
system job name	System-supplied name (19 characters) of the job that was submitted.

The following counter is defined for this statistic:

Counter	Description
1	Size of the input file in bytes.

Statistic Name	Definition and Counter Descriptions
AV32	Records a standard output file queued for printing. The descriptive data field has the following format:

```
user file name
```

Field	Description
user file name	User-supplied name (up to 31 characters) of the standard output file (OUTPUT).

The following counter is defined for this statistic:

Counter	Description
1	Size of the standard output file in bytes.

AV33	Records the completion of a permanent file transfer on the requesting mainframe. The descriptive data field has the following format:

```
requesting mainframe system name,
target mainframe system name,command
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
requesting mainframe system name	System name of the requesting mainframe. This may be a physical identifier (PID), logical identifier (LID), or title.
target mainframe system name	System name of the target mainframe. This may be a physical identifier (PID), logical identifier (LID), or title.
command	String consisting of an explicit or implicit command.

Statistic Name	Definition and Counter Descriptions
AV33 (Continued)	The following counters are defined for this statistic:

Counter	Description
1	Size of the transferred file in bytes.
2	Number of bytes sent and received exclusive of the file transfer.
3	Connect time, expressed in seconds.

AV34

Records the completion of a permanent file transfer on the target mainframe. The descriptive data field has the following format:

```
requesting mainframe system name,
target mainframe system name,command
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
requesting mainframe system name	System name of the requesting mainframe. This may be a physical identifier (PID), logical identifier (LID), or title.
target mainframe system name	System name of the target mainframe. This may be a physical identifier (PID), logical identifier (LID), or title.
command	String consisting of an explicit or implicit command.

The following counters are defined for this statistic:

Counter	Description
1	Size of the transferred file in bytes.
2	Number of bytes sent and received exclusive of the file transfer.
3	Connect time, expressed in seconds.

Statistic Name	Definition and Counter Descriptions
AV35	Records the completion of a queued file transfer on the originating mainframe. The descriptive data field has the following format:

```
family,user,account,project,system job name,
user file name or user job name,
originating mainframe system name,
destination mainframe system name
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
family	Family name associated with the user who initiated the file transfer.
user	User name of the user who initiated the file transfer.
account	User's account name.
project	User's project name.
system job name	System-supplied name of the job that originated the queued file transfer.
user file name or user job name	User-supplied name of the file or job that was transferred.
originating mainframe system name	System name of the originating mainframe. This may be a physical identifier (PID), logical identifier (LID), or title.
destination mainframe system name	System name of the destination mainframe. This may be a physical identifier (PID), logical identifier (LID), or title.

The following counter is defined for this statistic:

Counter	Description
1	Size of the transferred file in bytes.

Statistic Name	Definition and Counter Descriptions
AV36	Records the completion of a queued file transfer on the destination mainframe. The descriptive data field has the following format:

```
family,user,account,project,system job name,
user file name or user job name,
originating mainframe system name,
destination mainframe system name
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
family	For an output file: family name associated with the user who initiated the file transfer. For an input file: family name under which the transferred job executed.
user	For an output file: user name of the user who initiated the file transfer. For an input file: user name under which the transferred job executed.
account	For an output file: account name of the user who initiated the file transfer. For an input file: account name under which the transferred job executed.
project	For an output file: project name of the user who initiated the file transfer. For an input file: project name under which the job executed.
system job name	For an output file: system-supplied name of the job that originated the queued file transfer. For an input file: system-supplied name of the job that is executed.
user file name or user job name	User-supplied name of the file or job that was transferred.
originating mainframe system name	System name of the originating mainframe. This may be a physical identifier (PID), logical identifier (LID), or title.
destination mainframe system name	system name of the destination mainframe. This may be a physical identifier (PID), logical identifier (LID), or title.

Statistic Name	Definition and Counter Descriptions
AV36 (Continued)	The following counter is defined for this statistic:

Counter	Description
1	Size of the transferred file in bytes.

AV37

Records connect time and number of bytes transferred for an interactive terminal connection. This statistic is automatically enabled during NOS/VE deadstart and is valid only for interactive users who are connected to NOS/VE through the NAM/VE application within a CDCNET configuration. The descriptive data field has the following format:

```
DI system name,line or trunk name,
line or trunk subtype,line speed,device
```

Each descriptive data field entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
DI system name	Name of the device interface from which the user is communicating.
line or trunk name	Name of the line or trunk over which the user is communicating.
line or trunk subtype	Subtype of the line or trunk over which the user is communicating.
line speed	Speed of the line over which the user is communicating.
device	Name of the device from which the user is communicating.

The following counters are defined for this statistic:

Counter	Description
1	Total number of characters of input.
2	Total number of characters of output.
3	Connect time, expressed in seconds.

COBOL Compilation Statistics

The following COBOL compilation statistics are available (these are event statistics):

Statistic Name	Description
CB0	COBOL compilation.
CB1	Reserved for future use.
CB2	Nonexecutable lines.
CB3	Source statement errors.
CB4	Options used.
CB5	Number of each type of source statement.

The COBOL compilation statistics have the following descriptive data fields and counters defined:

Statistic Name **Definition**

CB0 — COBOL compilation. This statistic is issued to the statistics file at the end of a successful COBOL compilation. The following counters are defined for this statistic:

Counter	Description
1	Time, in microseconds, the task spent in the compiler.
2	Monitor time, expressed in microseconds.
3	Number of lines compiled.
4	Number of statements in the COBOL source file.
5	Number of stacked compiles.

CB1 — Reserved for future use.

CB2 — Nonexecutable lines. The following counters are defined for this statistic:

Counter	Description
1	Number of blank lines
2	Number of comment lines.

Statistic Name	Description
CB3	Source statement errors. The following counters are defined for this statistic:

Counter	Description
1	Number of informational diagnostics.
2	Number of warning diagnostics.
3	Number of error diagnostics.
4	Number of fatal diagnostics.
5	Number of catastrophic diagnostics.
6	Number of FIPS (Federal Information Processing Standard) diagnostics.
7	Number of nonstandard diagnostics.

Statistic Name	Description
CB4	Options used. A string of characters is issued. There are no counters for this statistic.

Statistic Name	Description
CB5	Number of each type of source statement. The following counters are defined for this statistic:

Counter	Description
1	Number of ACCEPT statements.
2	Number of ADD statements.
3	Number of ALTER statements.
4	Number of CALL statements.
5	Number of CANCEL statements.
6	Number of CLOSE statements.
7	Number of COMPUTE statements.
8	Number of CONTINUE statements.
9	Number of DELETE statements.
10	Number of DISABLE statements.
11	Number of DISPLAY statements.
12	Number of DIVIDE statements.
13	Number of ENABLE statements.
14	Number of ENTER statements.
15	Number of EVALUATE statements.
16	Number of EXIT statements.
17	Number of GENERATE statements.
18	Number of GO TO statements.
19	Number of IF statements.
20	Number of INITIALIZE statements.
21	Number of INITIATE statements.
22	Number of INSPECT statements.
23	Number of MERGE statements.
24	Number of MOVE statements.

Statistic Name	Description
CB5 (Continued)	

Counter	Description
25	Number of MULTIPLY statements.
26	Number of OPEN statements.
27	Number of PERFORM statements.
28	Number of PURGE statements.
29	Number of READ statements.
30	Number of RECEIVE statements.
31	Number of RELEASE statements.
32	Number of RETURN statements.
33	Number of REWRITE statements.
34	Number of SEARCH statements.
35	Number of SEND statements.
36	Number of SET statements.
37	Number of SORT statements.
38	Number of START statements.
39	Number of STOP statements.
40	Number of STRING statements.
41	Number of SUBTRACT statements.
42	Number of SUPPRESS statements.
43	Number of TERMINATE statements.
44	Number of UNSTRING statements.
45	Number of USE statements.
46	Number of WRITE statements.

Command Language Statistics

The following command language statistics are available (these are event statistics):

Statistic Name	Description
CL0	Display message command.
CL1	Command name.
CL2	Command resources.

The statistics CL1 and CL2 generate a very large number of entries. They should not be activated during normal operations and should be used during test operations and only at the direction of Control Data software support analysts.

NOTE

The CL2 statistic is a conditional statistic due to the large amount of data collected and generated when it is activated.

To activate the statistic, complete these steps:

1. Enter, from the system console:

```
SET_SYSTEM_ATTRIBUTE COMMAND_STATISTICS_ENABLED 1
```

2. Enter either an ACTIVATE_SYSTEM_STATISTICS or an ACTIVATE_JOB_STATISTICS command.

To turn the statistic off, complete these steps:

1. Enter:

```
SET_SYSTEM_ATTRIBUTE COMMAND_STATISTICS_ENABLED 0
```

2. Enter either a DEACTIVATE_SYSTEM_STATISTICS or a DEACTIVATE_JOB_STATISTICS command.

The command language statistics have the following descriptive data fields:

Statistic Name	Definition and Counter Descriptions
CL0	Display message command. The descriptive data field contains the string that was specified in the SCL command that generated the display message.
CL1	Command name. The descriptive data field contains an SCL command name in uppercase letters.
CL2	Command resources. This statistic is issued at the end of command processing. The data emitted does not include resources consumed by tasks other than the one from which the command is issued. The descriptive data field contains an SCL command name in uppercase letters. The following counters are also defined:

Counter	Description
1	Amount of CPU time, expressed in microseconds, spent in job mode.
2	Amount of CPU time, expressed in microseconds, spent in monitor mode.
3	Total number of page faults for the job.
4	Number of pages read in from disk for the job.
5	Number of pages reclaimed for the job.
6	Number of new pages created for the job.

Editor Statistics

The following NOS/VE editor statistics are available (these are event statistics):

Statistic Name	Description
ES0	Editor begin.
ES1	Editor end.

The editor statistics have the following descriptive data fields and counters defined:

Statistic Name	Definition
ES0	Editor begin. This statistic is issued at the beginnning of an editor session. No descriptive data field or counters are defined for this statistic.
ES1	Editor end. This statistic is issued at the end of an editor session. There is no descriptive data field for this statistic. The following counters are defined:

Counter	Description
1	Amount of CPU time, expressed in microseconds, spent in job mode.
2	Amount of CPU time, expressed in microseconds, spent in monitor mode.
3	Total number of page faults for the job.
4	Number of pages read in from disk for the job.
5	Number of pages reclaimed for the job.
6	Number of new pages created for the job.
7	Number of pages that were read from the file server.

FORTRAN Version 1 Compilation Statistics

The following FORTRAN Version 1 compilation statistics are available (these are event statistics):

Statistic Name	Description
FC0	FORTRAN Version 1 compilation.
FC1	FORTRAN Version 1 program unit.

The FORTRAN Version 1 compilation statistics have the following descriptive data fields and counters defined:

Statistic Name	Definition
FC0	FORTRAN Version 1 compilation. This statistic is issued to the statistics file at the end of a successful FORTRAN Version 1 compilation. The following counters are defined for this statistic:

Counter	Description
1	Time, expressed in microseconds, the task spent in the compiler.
2	Monitor time, expressed in microseconds.
3	Number of lines compiled (including blank lines and comment lines).
4	Number of statements in the FORTRAN Version 1 source file.
5	Number of program units.

FC1	FORTRAN Version 1 program unit. This statistic is issued to the statistics file each time a FORTRAN Version 1 program unit completes compilation. The descriptive data field has the following format:

```
program unit name
```

Field	Description
program unit name	Name of the FORTRAN Version 1 program unit.

Statistic Name	Definition and Counter Descriptions
FC1 (Continued)	The following counters are defined for this statistic:

Counter	Description
1	Task time, expressed in microseconds, for the program unit.
2	Monitor time, expressed in microseconds, for the program unit.
3	Number of lines in the program unit (including blank lines and comment lines).
4	Number of statements in the program unit.

FORTRAN Version 2 Compilation Statistics

The following FORTRAN Version 2 compilation statistics are available (these are event statistics):

Statistic Name	Description
FV0	FORTRAN Version 2 compilation.
FV1	FORTRAN Version 2 program unit.

The FORTRAN Version 2 compilation statistics have the following descriptive data fields and counters defined:

Statistic Name — **Definition**

FV0

FORTRAN Version 2 compilation. This statistic is issued to the statistics file at the end of a successful FORTRAN Version 2 compilation. The following counters are defined for this statistic:

Counter	Description
1	Time, expressed in microseconds, the task spent in the compiler.
2	Monitor time, expressed in microseconds.
3	Number of lines compiled (including blank lines and comment lines).
4	Number of statements in the FORTRAN Version 2 source file.
5	Number of program units.

FV1

FORTRAN Version 2 program unit. This statistic is issued to the statistics file each time a FORTRAN Version 2 program unit completes compilation. The descriptive data field has the following format:

```
program unit name
```

Field	Description
program unit name	Name of the FORTRAN Version 2 program unit.

Statistic Name	Definition and Counter Descriptions
FV1 (Continued)	The following counters are defined for this statistic:

Counter	Description
1	Task time, expressed in microseconds, for the program unit.
2	Monitor time, expressed in microseconds, for the program unit.
3	Number of lines in the program unit (including blank lines and comment lines).
4	Number of statements in the program unit.

Job Management Statistics

The following job management statistics are available (these are event statistics):

Statistic Name	Description
JM0	User name.
JM1	User job name.
JM2	Job mode.
JM3	Job end.
JM4	Job queueing started.
JM5	Job queueing aborted.
JM6	Output queueing started.
JM7	Output queueing aborted.
JM8	Job forwarding started.
JM9	Output forwarding started.
JM10	Job initiated.
JM11	Job terminated.
JM12	Print plot initiated.
JM13	Print plot terminated.
JM14	Submit job executed.
JM15	Print plot file executed.
JM16	Job history message.
JM17	Reserved for future use.
JM18	Change output attributes.

The following descriptive data fields and counters are defined for the job management statistics:

Statistic Name	Definition
JM0	User name. No counters are defined for this statistic. The descriptive data field is 31 characters long and contains the user name to which the job belongs.
JM1	Job name. No counters are defined for this statistic. The descriptive data field is 31 characters long and contains the user-supplied job name.
JM2	Job mode. No counters are defined for this statistic. The descriptive data field is 11 characters long and contains one of the following strings: INTERACTIVE The job is interactive mode. BATCH The job is batch mode.
JM3	Job end. This statistic is issued at the end of a user's job. There is no descriptive data field for this statistic. The following counters are defined:

Counter	Description
1	Amount of CPU time, expressed in microseconds, spent in job mode.
2	Amount of CPU time, expressed in microseconds, spent in monitor mode.
3	Total number of page faults for the job.
4	Number of pages read in from disk for the job.
5	Number of pages reclaimed for the job.
6	Number of new pages created for the job.
7	Maximum size (in pages) of the job working set actually attained during execution of the job.
8	Number of pages that were read from the file server.

Statistic Name	Description
JM4	Job queuing started. No counters are defined for this statistic. The descriptive data field has the following format:

```
system job name,user job name,login family,
login user,control family,control user,station
```

Field	Description
system job name	System-supplied name of job. Starting position is column 1; field is 19 characters long.
user job name	User-supplied name of job. Starting position is column 20; field is 31 characters long.
login family	Family name of the login user. Starting position is column 51; field is 31 characters long.
login user	User name of the login user. Starting position is column 82; field is 31 characters long.
control family	Family name of the control user for the job. Starting position is column 113; field is 31 characters long.
control user	User name of the control user for the job. Starting position is column 144; field is 31 characters long.
station	Station name for output from job. Starting position is column 175; field is 31 characters long.

Statistic Name	Description
JM5	Job queuing aborted. No counters are defined for this statistic. The descriptive data field has the following format:

```
system job name,reason
```

Field	Description
system job name	System-supplied name of job. Starting position is column 1; field is 19 characters long.
reason	Condition identifier for the job's aborting. Starting position is column 20; field is 31 characters long.

Statistic Name	Description
JM6	Output queuing started. No counters are defined for this statistic. The descriptive data field has the following format:

```
system job name,login family,login user,
system file name,control family,control user,station
```

Field	Description
system job name	System-supplied name of job. Starting position is column 1; field is 19 characters long.
login family	Family name of the login user. Starting position is column 20; field is 31 characters long.
login user	User name of the login user. Starting position is column 51; field is 31 characters long.
system file name	System-supplied name of output file. Starting position is column 82; field is 19 characters long.
control family	Family name of the control user for the job. Starting position is column 101; field is 31 characters long.
control user	User name of the control user for the job. Starting position is column 132; field is 31 characters long.
station	Station for output from job. Starting position is column 163; field is 31 characters long.

Statistic Name **Description**

JM7

Output queuing aborted. No counters are defined for this statistic. The descriptive data field has the following format:

```
system job name,system file name,reason
```

Field	Description
system job name	System-supplied name of job. Starting position is column 1; field is 19 characters long.
system file name	System-supplied name of output file. Starting position is column 20; field is 19 characters long.
reason	Condition identifier for the output's aborting. Starting position is column 39; field is 31 characters long.

JM8

Job forwarding started. No counters are defined for this statistic. The descriptive data field has the following format:

```
system job name
```

Field	Description
system job name	System-supplied name of job. Starting position is column 1; field is 19 characters long.

JM9

Output forwarding started. No counters are defined for this statistic. The descriptive data field has the following format:

```
system job name,system file name
```

Field	Description
system job name	System-supplied name of job. Starting position is column 1; field is 19 characters long.
system file name	System-supplied name of output file. Starting position is column 20; field is 31 characters long.

JM10

Job initiated. No counters are defined for this statistic.

The descriptive data field has the following format:

```
system job name
```

Field	Description
system job name	System-supplied name of job. Starting position is column 1; field is 19 characters long.

Statistic Name	Description
JM11	Job terminated. No counters are defined for this statistic. The descriptive data field has the following format:

```
system job name,system file name,output disposition,reason
```

Field	Description
system job name	System-supplied name of job. Starting position is column 1; field is 19 characters long.
system file name	System-supplied name of output file. Starting position is column 20; field is 31 characters long.
output disposition	Manner in which the output generated by the job is to be disposed. Starting position is column 39; field is 31 characters long.
reason	Condition identifier for termination. Starting position is column 70; field is 31 characters long.

JM12	Print plot initiated. No counters are defined for this statistic. The descriptive data field has the following format:

```
system job name,system file name
```

Field	Description
system job name	System-supplied name of job that generated the file. Starting position is column 1; field is 19 characters long.
system file name	System-supplied name of output file. Starting position is column 20; field is 31 characters long.

JM13	Print plot terminated. No counters are defined for this statistic. The descriptive data field has the following format:

```
system job name,system file name
```

Field	Description
system job name	System-supplied name of job whose output was terminated. Starting position is column 1; field is 19 characters long.
system file name	System-supplied name of output file. Starting position is column 20; field is 31 characters long.

Statistic Name	Description
JM14	Submit job executed. No counters are defined for this statistic. The descriptive data field has the following format:

```
system job name,job destination,
job destination usage,user job name
```

Field	Description
system job name	System-supplied name of job. Starting position is column 1; field is 19 characters long.
job destination	Identification of the system where the job is to execute. Starting position is column 20; field is 31 characters long.
job destination usage	Application to be used to forward the job to a remote system for execution. Starting position is column 51; field is 31 characters long.
user job name	User-supplied name of job. Starting position is column 82; field is 31 characters long.

Statistic Name	Description
JM15	Print plot file executed. No counters are defined for this statistic. The descriptive data field has the following format:

```
system job name,system file name,output destination,
output destination usage,user file name
```

Field	Description
system job name	System-supplied name of job issuing command. Starting position is column 1; field is 19 characters long.
system file name	System-supplied name of output file. Starting position is column 20; field is 31 characters long.
output destination	Where the output file is to be sent for printing. Starting position is column 39; field is 31 characters long.
output destination usage	Type of CDCNET print station where the output file will be printed or the name of the application used to forward the output file to a remote system. Starting position is column 70; field is 31 characters long.
user file name	User-supplied name of the file being printed. Starting position is column 101; field is 31 characters long.

Statistic Name	Description
JM16	Job history message. No counters are defined for this statistic. The descriptive data field has the following format:

```
system job name,message
```

Field	Description
system job name	System-supplied name of job. Starting position is column 1; field is 19 characters long.
message	Message issued by job to history log. Starting position is column 20; field is 256 characters long.

Statistic Name	Description
JM17	Reserved for future use.
JM18	Change output attributes. No counters are defined for this statistic. The descriptive data field has the following format:

```
system job name,system file name,control family,control user,
output destination,output destination usage,station
```

Field	Description
system job name	System-supplied name of the job that generated the output file. Starting position is column 1; field is 19 characters long.
system file name	Name of file whose attributes are being changed. Starting position is column 20; field is 31 characters long.
control family	Family name of the control user for the file. Starting position is column 39; field is 31 characters long.
control user	User name of the control user for the file. Starting position is column 70; field is 31 characters long.
output destination	Where the output is to be sent for printing. Starting position is column 101; field is 31 characters long.
output destination usage	Type of CDCNET print station where the output file will be printed or the name of the application used to forward the output file to a remote system. Starting position is column 132; field is 31 characters long.
station	Station name for the output file. Starting position is column 163; field is 31 characters long.

Logging Statistics

The following logging statistics are available (these are event statistics):

Statistic Name	Description
LG0	Start of log segment.
LG1	End of log segment.

The following descriptive data fields are defined for the logging statistics:

Statistic Name	Definition
LG0	Start of log segment. When a TERMINATE_LOG command is entered to end a segment of log entries and begin a new segment, the LG0 statistic is entered as the first statistic of the new segment. There is no way to disable this statistic.

The descriptive data field has the following format:

```
log termination identifier
```

Field	Description
log termination identifier	Unique, system-supplied name. The unique name assigned to an LG0 statistic is the same name assigned to the LG1 statistic that closed the previous log segment. The use of matching name identifiers facilitates the correct sequencing of log segments.

The LG0 statistic has no counters.

LG1	End of log segment. Entered as the last statistic in a log segment being terminated as a result of a TERMINATE_LOG command. There is no way to disable this statistic.

The descriptive data field has the following format:

```
log termination identifier
```

Field	Description
log termination identifier	Unique, system-supplied name. The name supplied will match the unique name assigned to the LG0 statistic that begins the new log segment.

The LG1 statistic has no counters.

Mail/VE Version 1 Accounting Statistics

The following Mail/VE Version 1 accounting statistics are available (these are event statistics):

Statistic Name	Description
MV0	Sending charge.
MV1	Storage charge.

The following descriptive data fields and counters are defined for the Mail/VE Version 1 accounting statistics:

Statistic Name	Definition
MV0	Sending charge issued when a letter is sent. The descriptive data field has the following format:

```
family name,user name
```

Field	Description
family name	Family name associated with the sending charge.
user name	User name associated with the sending charge.

The following counters are defined:

Counter	Description
1	Number of characters in the letter.
2	Number of mailboxes to which the letter was sent.
3	Number of families associated with mailboxes to which the letter was sent.

Statistic Name	Definition
MV1	Storage charge issued when a letter is deleted from a mailbox. The descriptive data field has the following format:

```
family name,user name
```

Field	Description
family name	Family name associated with the storage charge.
user name	User name associated with the storage charge.

The following counters are defined:

Counter	Description
1	Number of characters in the letter.
2	Number of milliseconds this letter has been retained for this mailbox since the last Mail/VE Version 1 statistic was issued for this letter.
3	Number of mailboxes currently associated with this letter.

Mail/VE Version 2 Accounting Statistics

The following Mail/VE Version 2 accounting statistics are available (these are event statistics):

Statistic Name	Description
MV500	Sending charge issued when letter is sent via the Mail/VE user interface.
MV501	Information that is collected when a letter is read.
MV502	Information that is collected when a letter is placed in a mailbox.
MV503	Session length.
MV504	Sending charge issued by message transfer agent.

The following descriptive data fields and counters are defined for the Mail/VE Version 2 accounting statistics:

Statistic Name	Description
MV500	Sending charge issued when a letter is sent via the Mail/VE user interface. The descriptive data field has the following format:

```
user name,family name,mailbox,message identifier
```

Field	Description
user name	Sender's user name.
family name	Sender's family name.
mailbox	Sender's mailbox address.
message identifier	Message identifier consisting of the country name, administrative domain name, and private domain name.

No counters are defined for this statistic.

Statistic Name	Description
MV501	Information that is collected when a letter is read. The descriptive data field has the following format:

```
user name,family name,mailbox,message identifier
```

Field	Description
user name	Reader's user name.
family name	Reader's family name.
mailbox	Reader's mailbox address.
message identifier	Message identifier consisting of the country name, administrative domain name, and private domain name.

No counters are defined for this statistic.

Statistic Name	Description
MV502	Information that is collected when a letter is placed in a mailbox. The descriptive data field has the following format:

```
user name,family name,mailbox,message identifier
```

Field	Description
user name	Recipient's user name.
family name	Recipient's family name.
mailbox	Recipient's mailbox address.
message identifier	Message identifier consisting of the country name, administrative domain name, and private domain name.

No counters are defined for this statistic.

Statistic Name	Description
MV503	Session length. The descriptive data field has the following format:

```
user name,family name
```

Field	Description
user name	User's user name.
family name	User's family name.

The following counter is defined:

Counter	Description
1	Duration of session, in seconds.

Statistic Name	Description
MV504	Sending charge issued by a message transfer agent. The descriptive data field has the following format:

```
message identifier, destination title, message size
```

Field	Description
message identifier	Message identifier consisting of the country name, administrative domain name, and private domain name.
destination title	Name of the letter's destination.
message size	Size of the message in bytes.

No counters are defined for this statistic.

NAM/VE Statistics

The following NAM/VE statistics are available. With the exception of NA10, these are periodic statistics.

Statistic Name	Description
NA0	Amount of work performed by each NAM/VE layer and network management entity.
NA1	Amount of work performed by the NAM/VE Intranet layer.
NA2	Amount of work performed by each NAM/VE layer and network management entity.
NA3	Amount of work performed by the NAM/VE channel connection (CC) entity.
NA10	Statistic emitted when the number of times the maximum number of login attempts (over NAM/VE) has been exceeded. This is an event statistic.

Like the OS statistics, the NA statistics (except for NA10) are designed for regular periodic emission. Before it can be logged, an NA statistic must be assigned to an emission set with the MANAGE_PERIODIC_STATISTICS utility. For more information about periodic statistics, refer to MANAGE_PERIODIC_STATISTICS Utility earlier in this chapter.

The following data fields and counters are defined for the NA statistics. If a descriptive data field is not specified, none exists.

Statistic Name	Description
NA0	Amount of work performed by each NAM/VE layer and network management entity. The following counters are defined:

Counter	Description
1	Number of Internet broadcast service requests. A broadcast means that a message is sent to all Control Data Network Architecture (CDNA) systems on all known networks. Incremental counter.
2	Number of protocol data units (PDUs) Internet has received from Intranet. Incremental counter.
3	Number of Internet requests to send PDUs. Incremental counter.
4	Number of PDUs relayed between network solutions by Internet. Incremental counter.
5	Number of PDUs Internet has routed within the local system. Incremental counter.

Statistic Name	Description	
NA0 (Continued)	**Counter**	**Description**
	6	Number of transport connections that have been created. Incremental counter.
	7	Number of transport connections that are active. Snapshot counter.
	8	Number of transport connections that are not active and are in the process of terminating. Snapshot counter.
	9	Number of transport connections that have terminated. Incremental counter.
	10	Number of data packets received by transport. Incremental counter.
	11	Number of data packets sent by transport. Incremental counter.
	12	Number of data packets discarded by transport. Incremental counter.
	13	Number of duplicate data packets received by transport. Incremental counter.
	14	Number of expedited data packets received by transport. Incremental counter.
	15	Number of expedited data packets sent by transport. Incremental counter.
	16	Number of expedited data packets discarded by transport. Incremental counter.
	17	Number of duplicate data packets received by transport. Incremental counter.
	18	Number of acknowledgment requests received by transport. An acknowledgment is a confirmation for previously sent data or expedited data. Incremental counter.
	19	Number of acknowledgment requests sent by transport. Incremental counter.
	20	Number of acknowledgments discarded by transport. Incremental counter.
	21	Number of probe packets received by transport. A probe packet is used by transport to determine if its communicating peer is still active. Incremental counter.

Statistic Name	Description
NA0 (Continued)	

Counter	Description
22	Number of probe packets sent by transport. Incremental counter.
23	Number of probe packets discarded by transport. Incremental counter.
24	Number of times previously sent data has been retransmitted by transport. Incremental counter.
25	Number of error packets received by transport. An error packet is generated when data is received for an inactive connection. Incremental counter.
26	Number of error packets sent by transport. Incremental counter.
27	Number of session synchronize requests received. Incremental counter.
28	Number of session synchronize requests sent. Incremental counter.
29	Number of session interrupt requests received. Incremental counter.
30	Number of session interrupt requests sent. Incremental counter.
31	Number of duplicate routing information data units (RIDUs) received. A duplicate RIDU has the same sequence number as a previous RIDU received from the same system. Incremental counter.
32	Number of RIDUs received from other multihomed systems. Incremental counter.
33	Number of RIDUs sent by the local system. Incremental counter.
34	Number of RIDUs aged out (that is, the RIDU is deleted after its lifetime has expired). Incremental counter.
35	Number of times the routing table is recomputed due to changes in the directly connected network. Incremental counter.
36	Number of times the routing table is recomputed due to changes in the remote networks. Incremental counter.

Statistic Name	Description
NA0 (Continued)	

Counter	Description
37	Number of times the routing table has been partially updated. Incremental counter.
38	Current number of registered titles in the NAM/VE directory. Snapshot counter.
39	Current number of entries in the translation cache. Title translations received from other NOS/VE and CDCNET systems over the network are stored in the translation cache. Snapshot counter.
40	Current number of active directory searches. Snapshot counter.
41	Number of initiated directory searches. Incremental counter.
42	Number of get title translation requests that returned a title translation. Incremental counter.
43	Number of title translations found in the directory of the local NOS/VE system that satisfy title translation requests. Incremental counter.
44	Number of title translations found in the translation cache of the local NOS/VE system that satisfy title translation requests. Incremental counter.
45	Number of title translations received that were broadcast from other NOS/VE systems in the network. Incremental counter.
46	Number of title translations broadcast to all other NOS/VE systems in the network. Incremental counter.
47	Number of title translations received that were sent specifically to this NOS/VE system over the network. Incremental counter.
48	Number of title translations sent to NOS/VE and CDCNET systems because they satisfy translation requests received from those systems. Incremental counter.
49	Number of title translation requests broadcast to all other NOS/VE and CDCNET systems in the network. Incremental counter.
50	Number of title translation requests received from other NOS/VE and CDCNET systems in the network. Incremental counter.

Statistic Name	Description	
NA0 (Continued)	**Counter**	**Description**
	51	Current number of active file access management entity connections. Snapshot counter.
	52	Number of files accessed by the file access management entity. Incremental counter.
	53	Number of instances the buffer manager descriptor pool became empty. Incremental counter.
	54	Number of small buffer manager containers allocated. Incremental counter.
	55	Number of large buffer manager containers allocated. Incremental counter.
	56	Number of small buffer manager containers freed. Incremental counter.
	57	Number of large buffer manager containers freed. Incremental counter.
	58	Number of instances the small container pool of the PP buffer pool became empty. Incremental counter.
	59	Number of instances the large container pool of the PP buffer pool became empty. Incremental counter.
	60	Number of instances the small container pool of the PP buffer pool became replenished. Incremental counter.
	61	Number of instances the large container pool of the PP buffer pool became replenished. Incremental counter.

Statistic Name	Description
NA1	Amount of work performed by the NAM/VE Intranet layer. These statistics are separately collected and recorded for each network solution. The descriptive data field has the following format:

```
network identifier
```

Field	Description
network identifier	Network identifier of the network solution for which the information was collected.

The following counters are defined:

Counter	Description
1	Number of multicasts Intranet has received. A multicast is a message sent to all Control Data Network Architecture (CDNA) systems on a single network. Incremental counter.
2	Number of multicasts Intranet has sent. Incremental counter.
3	Average size of the protocol data units (PDUs) Intranet has received. Incremental counter.
4	Total number of PDUs Intranet has received. Incremental counter.
5	Number of received PDUs that were discarded by Intranet. Incremental counter.
6	Average size of the Intranet PDUs sent. Incremental counter.
7	Total number of PDUs sent by Intranet. Incremental counter.
8	Number of PDUs that have been discarded by Intranet. Incremental counter.
9	Current number of Intranet PDUs that are queued to be sent. Snapshot counter.
10	Average number of noncontiguous memory locations used for outgoing Intranet PDUs. Incremental counter.

Statistic Name	Description
NA2	Amount of work performed by each NAM/VE layer and network management entity. This statistic relates only to OSI (open system interconnection). The following counters are defined for this statistic:

Counter	Description
1	Number of channel connection (CC) requests that are broadcast. Incremental counter.
2	Current total number of channel connections over all communications devices. Snapshot counter.
3	Current total number of priority channel connections over all communications devices. A priority connection is defined as a connection having a consistent level of service. Snapshot counter.
4	Current total number of link access agent (LAA) service access points (SAPs) that are open. Snapshot counter.
5	Number of LAA protocol data units (PDUs) that are received. Incremental counter.
6	Number of LAA PDUs sent. Incremental counter.
7	Total number of bytes of data received by LAA. Incremental counter.
8	Total number of bytes of data sent by LAA. Incremental counter.
9	Number of broadcasts sent by the network access agent (NAA). Incremental counter.
10	Number of NAA PDUs received. Incremental counter.
11	Number of NAA PDUs sent. Incremental counter.
12	Total number of bytes of data received by NAA. Incremental counter.
13	Total number of bytes of data sent by NAA. Incremental counter.
14	Number of times a specific route to a Control Data Network Architecture (CDNA) destination could not be determined by the system management access agent (SMAA). Incremental counter.
15	Number of select device requests made with CDNA addresses. Incremental counter.
16	Number of select device requests made with non-CDNA addresses. Incremental counter.

Statistic Name	Description
NA2 (Continued)	

Counter	Description
17	Number of times specific routes to non-CDNA destinations could not be determined. Incremental counter.
18	Number of times queries were issued to determine a route to a given destination. Incremental counter.
19	Number of times the OSI subnetwork attributes were updated. Incremental counter.
20	Number of data PDUs received by the transport access agent (TAA). Incremental counter.
21	Number of data PDUs sent by the TAA. Incremental counter.
22	Number of expedited data PDUs received by the TAA. Expedited data is data that is not subject to flow control. Incremental counter.
23	Number of expedited data PDUs sent by the TAA. Incremental counter.
24	Total number of bytes received by the TAA. Incremental counter.
25	Total number of bytes sent by the TAA. Incremental counter.

Statistic Name	Description
NA3	Amount of work performed by the NAM/VE channel connection (CC) entity. This statistic relates only to OSI (open system interconnection). The data for this statistic is separately collected and recorded for each communications device. The descriptive data field has the following format:

```
network identifier
```

Field	Description
network identifier	Network identifier (up to 10 characters) for each connected communications device for which the information was collected.

The following counters are defined:

Counter	Description
1	Number of credit request protocol data units (PDUs) received. A credit PDU received contains the number of messages that the communicating peer will allow the local CC to send. Incremental counter.
2	Number of credit request PDUs sent to the communicating peer by the local CC. A credit PDU sent contains the number of messages that the CC is currently willing to accept from the communicating peer. Incremental counter.
3	Current number of normal connections to a particular device. Snapshot counter.
4	Current number of priority connections to a particular device. A priority connection is defined as a connection having a consistent level of service. Snapshot counter.
5	Number of times the link to a particular device has been reset. Incremental counter.
6	Number of times a duplicate connect request has been received from the communicating peer. Incremental counter.
7	Current number of send data PDUs queued on a normal connection for a particular device. Snapshot counter.
8	Number of PDUs processed out of order. Incremental counter.

Statistic Name	Description	
NA3 (Continued)	**Counter**	**Description**
	9	Average size of data PDUs received on priority connections. Incremental counter.
	10	Total number of data PDUs received on priority connections. Incremental counter.
	11	Number of expedited data PDUs received on priority connections. Expedited data is data sent by CC that is not subject to flow control. Incremental counter.
	12	Number of data PDUs received on priority connections but discarded. Incremental counter.
	13	Average size of data PDUs sent on priority connections. Incremental counter.
	14	Total number of data PDUs sent on priority connections. Incremental counter.
	15	Number of expedited data PDUs sent on priority connections. Incremental counter.
	16	Number of expedited data PDUs sent on priority connections but discarded. Incremental counter.
	17	Number of priority data PDUs that are queued on the channel connection waiting for the peer to give additional credits. Snapshot counter.
	18	Average size of data PDUs received. Incremental counter.
	19	Number of data PDUs received. Incremental counter.
	20	Number of PDUs received by CC but discarded. Incremental counter.
	21	Number of expedited PDUs received. Incremental counter.
	22	Average size of a CC data PDU sent. Incremental counter.
	23	Total number of CC data PDUs sent. Incremental counter.
	24	Number of expedited data PDUs sent. Incremental counter.
	25	Number of data PDUs that could not be sent and were discarded. Incremental counter.

Statistic Name	Description
NA10	Statistic emitted when the maximum number of user login attempts over NAM/VE has been exceeded. The descriptive data field has the following format. Each entry consists of a maximum of 31 characters, has trailing blanks removed, and is separated from the next entry by a comma.

```
user name,family name,terminal name
```

Field	Description
user name	Login user name.
family name	Family name under which the user logged in.
terminal name	System-supplied name of the terminal.

The following counter is defined for this statistic:

Counter	Description
1	Value of the NAM attribute MAXIMUM_LOGIN_ATTEMPTS. Snapshot counter.

Optical Disk Accounting Statistics

The following descriptive data fields and counters are defined for the optical disk accounting statistics:

Statistic Name	Description
OD10	Information that is collected when a file section is closed. The descriptive data field has the following format:

```
mf.controller.unit,vsn,file,version,owner,group,partition
```

Field	Description
mf	Mainframe identifier.
controller	Optical disk controller name.
unit	Optical disk unit name.
vsn	Volume serial number of the volume containing the file.
file	Recorded file name of the file.
version	Version number of the file.
owner	Owner identification of the file.
group	Group identification of the file.
partition	Volume partition containing the file.

The following counters are defined for this statistic:

Counter	Description
1	Number of bytes read. (Not implemented yet).
2	Number of bytes written. (Not implemented yet).
3	Number of sectors read.
4	Number of sectors written.
5	Total number of requests to the PP.
6	Number of bytes per sector.
7	Wallclock seconds. (Not implemented yet).

Operating System Statistics

The following operating statistics are available (these are periodic statistics):

Statistic Name	Description
OS0	Job memory interval.
OS1	Paging monitor interval.
OS2	PP usage.
OS3	Path usage.
OS4	Disk unit usage.
OS5	Disk unit space available.
OS6	CPU usage.
OS7	Service class usage.
OS8	Job class usage.

Like most NA statistics, the OS statistics are designed for regular periodic emission. Before it can be logged, an OS statistic must be assigned to an emission set with the MANAGE_PERIODIC_STATISTICS utility. For more information about periodic statistics, refer to MANAGE_PERIODIC_STATISTICS Utility earlier in this chapter.

The following data fields and counters are defined for the OS statistics. If a descriptive data field is not specified, none exists.

Statistic Name	Description
OS0	Job memory interval. The values for the counters of these statistics reflect just the data for the current time interval. The following counters are defined for this statistic:

Counter	Description
1	Number of free pages. Snapshot counter.
2	Number of available pages. Snapshot counter.
3	Number of available-modified pages. Snapshot counter.
4	Number of mainframe-wired pages. Snapshot counter.
5	Number of pages in all the shared queues. Snapshot counter.
6	Number of fixed pages. Snapshot counter.
7	Number of pages with an I/O error. Snapshot counter.
8	Number of job working set pages. Snapshot counter.

Statistic Name	Description
OS0 (Continued)	

Counter	Description
9	Number of swapped jobs. Snapshot counter.
10	Number of ready tasks. Snapshot counter.
11	Number of interactive mode jobs. Snapshot counter.
12	Number of noninteractive jobs. Snapshot counter.
13	Number of pages in the task service shared queue. Snapshot counter.
14	Number of pages in the executable file shared queue. Snapshot counter.
15	Number of pages in the nonexecutable file shared queue. Snapshot counter.
16	Number of pages in the device file shared queue. Snapshot counter.
17	Number of pages in the file server shared queue. Snapshot counter.
18	Number of pages in the other shared queue. Snapshot counter.

Statistic Name	Description
OS1	Paging monitor interval. The following counters are defined for this statistic:

Counter	Description
1	Number of page faults from the available queue. Incremental counter.
2	Number of page faults from the available-modified queue. Incremental counter.
3	Number of page faults from disk. Incremental counter.
4	Number of new pages. Incremental counter.
5	Number of page faults for pages already being brought into memory. Incremental counter.
6	Number of page fault I/O requests rejected because I/O queues were full. Incremental counter.
7	Number of occurrences of forced aggressive aging. Incremental counter.
8	Number of times aggressive aging caused the shared queue to be aged. Incremental counter.
9	Number of times aggressive aging caused the job queue to be aged. Incremental counter.
10	Number of times where aggressive aging failed. Incremental counter.
11	Number of occurrences of write aged page. Incremental counter.
12	Number of cycle monitor requests. Incremental counter.
13	Average duration of monitor time servicing cycle requests, expressed in microseconds. Snapshot counter.
14	Number of delay monitor requests. Incremental counter.
15	Average duration of monitor time servicing delay requests, expressed in microseconds. Snapshot counter.
16	Number of wait monitor requests. Incremental counter.
17	Average duration of monitor time servicing wait requests, expressed in microseconds. Snapshot counter.

Statistic Name	Description
OS1 (Continued)	

Counter	Description
18	Number of write-modified monitor requests. Incremental counter.
19	Average duration of monitor time servicing write modified pages.
20	Number of times page streaming prestream mode was initiated. Incremental counter.
21	Number of times page streaming mode was initiated. Incremental counter.
22	Number of times page streaming prestream mode was terminated. Incremental counter.
23	Number of times page streaming mode was terminated by random page faults. Incremental counter.
24	Number of pages read in prestream mode. Incremental counter.
25	Number of pages read in page streaming mode. Incremental counter.
26	Number of times a page streaming task was too slow to keep up with the disk. Incremental counter. When a segment being read from disk in page streaming mode, the system reads ahead several pages. Normally the task will catch up with the disk and reference one of those pages before the I/O is complete. If not, this counter is incremented.
27	Number of times a page streaming fault occurred within the same transfer unit as the last page streaming fault. Incremental counter.
28	Number of pages freed behind. Incremental counter. To control the size of the job working set, pages of a segment being read sequentially are sometimes freed immediately. These pages are behind the current read position. Hence the term *free behind*.
29	Number of times page streaming mode continued after a random page fault.

Statistic Name	Description
OS2	PP usage. The descriptive data field has the following format:

```
iou,concurrence,channel
```

Each descriptive data field entry has trailing blanks removed and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
iou	IOU number on which the PP is located.
concurrence	Integer indicating whether the PP is concurrent (1) or noncurrent (0).
channel	Physical channel number to which the PP is connected.

The following counters are defined for this statistic; all counter values are incremental:

Counter	Description
1	Data transfer time for the PP, expressed in microseconds. Incremental counter.
2	Seek and latency time for the PP, expressed in microseconds. Incremental counter.
3	Number of read requests that streamed. Incremental counter.
4	Number of read requests that failed to stream. Incremental counter.
5	Number of write requests that streamed. Incremental counter.
6	Number of write requests that failed to stream. Incremental counter.

Statistic Name	Description
OS3	Path usage. The descriptive data field has the following format:

```
iou,concurrence,channel,port equipment,type
```

Each descriptive data field entry has trailing blanks removed and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
iou	IOU number on which the PP is located.
concurrence	Integer indicating whether the PP is concurrent (1) or noncurrent (0).
channel	Physical channel number to which the PP is connected.
port	Port to which the path is configured. One of the following values appears: N (unspecified; all nonconcurrent channels); U (unspecified; concurrent channel but does not use port); A (port A); B (port B.
equipment	Equipment number of the controller configured to the path.
type	Type of controller configured to the path.

The following counters are defined for the OS3 statistic:

Counter	Description
1	Number of read requests. Incremental counter.
2	Number of bytes read. Incremental counter.
3	Number of write requests. Incremental counter.
4	Number of bytes written (data and preset). Incremental counter.
5	Total request queue time, expressed in microseconds. Incremental counter.
6	Intermediate errors. Incremental counter.
7	Recovered errors. Incremental counter.
8	Unrecovered errors. Incremental counter.
9	Number of bytes per MAU (minimum addressable unit). Snapshot counter.

Statistic Name	Description
OS4	Disk unit usage. The descriptive data field has the following format:

```
rvsn,type
```

Each descriptive data field entry has trailing blanks removed and is separated from the next entry by a comma. The entries are positional; that is, a comma followed by a blank space means there is no descriptive data field entry for that position.

Field	Description
rvsn	Recorded VSN for the disk unit.
type	Type of disk unit.

The following counters are defined for this statistic:

Counter	Description
1	Number of read requests. Incremental counter.
2	Read queue time, expressed in microseconds. Incremental counter.
3	Number of bytes read. Incremental counter.
4	Number of write requests. Incremental counter.
5	Write queue time, expressed in microseconds. Incremental counter.
6	Number of bytes written as data. Incremental counter.
7	Number of bytes written as data and preset. Incremental counter.
8	Number of swapin requests. Incremental counter.
9	Swapin queue time, expressed in microseconds. Incremental counter.
10	Number of bytes swapped in. Incremental counter.
11	Number of swapout requests. Incremental counter.
12	Swapout queue time, expressed in microseconds. Incremental counter.
13	Number of bytes swapped out as data. Incremental counter.

Statistic Name	Description
OS4 (Continued)	

Counter	Description
14	Number of bytes swapped out as data and preset. Incremental counter.
15	Number of read requests that streamed. Incremental counter.
16	Number of read requests that failed to stream. Incremental counter.
17	Number of write requests that streamed. Incremental counter.
18	Number of write requests that failed to stream. Incremental counter.
19	Number of requests that caused skipped cylinders. Incremental counter.
20	Total number of cylinders skipped. Incremental counter.
21	Number of intermediate errors. Incremental counter.
22	Number of recovered errors. Incremental counter.
23	Number of unrecovered errors. Incremental counter.
24	Number of bytes per MAU (minimum addressable unit). Snapshot counter.

Statistic Name	Description
OS5	Disk unit space available. The descriptive data field has the following format:

```
rvsn
```

Field	Description
rvsn	Recorded VSN for the disk unit.

The following counter is defined for this statistic:

Counter	Description
1	Number of bytes of available space on the disk unit. Snapshot counter.

Statistic Name	Description
OS6	CPU usage. The following counters are defined for this statistic. Job mode CPU time and monitor mode CPU time include the CPU time assigned to all jobs, including the system job.

Counter	Description
1	NOS/VE idle time, in microseconds, when no I/O was taking place. Incremental counter.
2	NOS/VE idle time, in microseconds, when I/O was taking place. Incremental counter.
3	Job mode CPU time, in microseconds. Incremental counter.
4	Monitor mode CPU time, in microseconds. Incremental counter.
5	Total time, in microseconds, spent in NOS or NOS/BE. This counter contains a zero value if the system is not dual state. Incremental counter.
6	Time, in microseconds, spent in NOS or NOS/BE because NOS/VE was idle. This is a subset of counter 5. Incremental counter.

Statistic Name	Description
OS7	Service class usage. In each emission period, this statistic is issued for each service class in the system's active scheduling profile. The descriptive data field has the following format:

```
name
```

Field	Description
name	Name of the service class.

Each counter contains data accumulated for all jobs in the service class named in the counter's descriptive data field. The following counters are defined for this statistic:

Counter	Description
1	Total job mode CPU time. Incremental counter.
2	Total monitor mode CPU time. Incremental counter.
3	Total pages added to job working sets because of page faults. Incremental counter.
4	Total pages added to job working sets because of page faults from disk jobs. Incremental counter.
5	Total pages added to job working sets because of page faults from the available or available modified page queues. Incremental counter.
6	Total new pages added to job working sets because of page faults. Incremental counter.
7	Current number of active jobs. Snapshot counter.
8	Current number of jobs in the swapin queue. Snapshot counter.
9	Number of long wait swapouts. Incremental counter.
10	Number of swapouts forced by the job scheduler because of lower priority. Incremental counter.
11	Number of swapins delayed because of insufficient available memory. Incremental counter.
12	Number of swapins delayed because of no available active job list ordinals. Incremental counter.
13	Total pages swapped out. Incremental counter.

Statistic Name	Description
OS7 (Continued)	

Counter	Description
14	Total time spent as active jobs in memory. Incremental counter.
15	Total time spent as swapin candidates. Incremental counter.
16	Total swapins. Incremental counter.
17	Total time spent in think state. Incremental counter.
18	Number of times jobs were in think state. Incremental counter.

OS8

Job class usage. In each emission period, this statistic is issued for each job class in the system's active scheduling profile. The descriptive data field has the following format:

```
name
```

Field	Description
name	Name of the job class.

Each counter contains data accumulated for all jobs in the job class named in the counter's descriptive data field. The following counters are defined for this statistic:

Counter	Description
1	Total number of jobs in the input queue. Snapshot counter.
2	Total number of jobs completed. Incremental counter.

Program Management Statistics

The following program management statistics are available (these are event statistics):

Statistic Name	Description
PM0	Task begin.
PM1	Task starting procedure.
PM2	Task name.
PM3	Task end.
PM4	Loader begin.
PM5	Loader end.
PM6	Task end exception.

These statistics have the following descriptive data fields and counters defined:

Statistic Name **Definition**

PM0

Task begin. This statistic does not have a descriptive data field or any counters defined.

PM1

Task starting procedure. This statistic is issued each time the task starting procedure is executed. The descriptive data field has the following format:

```
starting procedure name
```

Field	Description
starting procedure name	Name of the starting procedure for the task.

No counters are defined for this statistic.

PM2

Task name. This statistic is issued each time a task name is encountered. The descriptive data field has the following format:

```
task name
```

Field	Description
task name	Name of the task.

No counters are defined for this statistic.

Statistic Name	Statistic Definition
PM3	Task end. This statistic is issued at the completion of each task. The descriptive data field has the following format:

```
task name
```

Field	Description
task name	Name of the task.

The following counters are defined:

Counter	Description
1	CPU time, expressed in microseconds, in job mode. Snapshot counter.
2	CPU time, expressed in microseconds, in monitor mode. Snapshot counter.
3	Number of page faults. Snapshot counter.
4	Number of pages read from disk. Snapshot counter.
5	Number of pages reclaimed. Snapshot counter.
6	Number of new pages assigned. Snapshot counter.
7	Maximum size (in pages) of the job working set actually attained during execution of the task. Snapshot counter.
8	Number of times the task was slowed down due to excessive paging while the job working set was at its maximum size. Snapshot counter.
9	Number of pages that were read from the file server. Snapshot counter.

Statistic Name	Statistic Definition
PM4	Loader begin. This statistic is issued each time the loader task begins execution. No descriptive data fields or counters are defined for this statistic.

Statistic Name	Statistic Definition
PM5	Loader end. This statistic is issued each time the loader task completes successfully. The descriptive data field has the following format:

```
task name
```

Field	Description
task name	Name of the task.

The following counters are defined:

Counter	Description
1	CPU time, expressed in microseconds, in job mode. Snapshot counter.
2	CPU time, expressed in microseconds, in monitor mode. Snapshot counter.
3	Number of page faults. Snapshot counter.
4	Number of pages read from disk. Snapshot counter.
5	Number of pages reclaimed. Snapshot counter.
6	Number of new pages assigned. Snapshot counter.

Statistic Name	Statistic Definition
PM6	Task end exception. This statistic is issued only at the completion of a task in which an exception has occurred. The only exception causing this statistic to be issued is when the task slowed down at least once because of excessive paging while the job working set was at its maximum size. The descriptive data field has the following format:

```
system job name,user job name,task name
```

Field	Description
system job name	System-supplied name of the job.
user job name	User-supplied name of the job.
task name	Name of the task.

The following counters are defined:

Counter	Description
1	Maximum size (in pages) of the job working set actually attained during execution of the task. Snapshot counter.
2	Number of times the task was slowed down because of excessive paging while the job working set was at its maximum size. Snapshot counter.

NOS/VE Security Audit Statistics

NOS/VE monitors security-related operations to provide an audit trail. The security audit trail includes a date/time stamp, the name of the job performing the operation, the identity of the user, the terminal name (if interactive), and the success or failure of the operation. This information is used by a security administrator to analyze security-related issues. Sites can choose the operations they want to audit based on selection criteria. Sites can also prevent anyone from disabling auditing once it has been initiated.

For NOS/VE security audit statistics, the statistic product identification is SF. These statistics are described in the NOS/VE Security Administration manual.

Configuration Management Statistics

Software within NOS/VE monitors the performance of disk and tape units belonging to NOS/VE during normal system operations. When error conditions are encountered (both recoverable and unrecoverable), statistics are entered in the system's engineering log. The configuration management statistics are automatically activated during NOS/VE deadstart. You do not need to activate or deactivate them.

As explained at the beginning of this chapter, each statistic has two parts: a descriptive data field and a counter portion. The descriptive data field is an ASCII string that describes the elements in use at the time of the failure and that usually gives some indication of failure severity and failure isolation.

The counter portion is an array of integer counter words containing binary information on the physical path, general status, detailed status, and so forth. By convention in the configuration management disk and tape statistics, bit 0 (leftmost bit) of each single-valued counter word indicates the presence or absence of information in the remainder of the word. If bit 0 is set, you can ignore the contents of the word. If bit 0 is not set, then the counter word contains failure data. (Single-valued counter words contain only one value; that is, they do not contain hardware status in a packed form.)

For more information about disk failure reporting and recovery, see The NOS/VE System Performance and Maintenance manual, Volume 2.

CM Statistics Text File

For configuration management statistics, the statistic product identification is CM. These event statistics are fully documented in text file $SYSTEM.MANUALS.LINE_PRINTER_MANUALS.CM_STATISTICS. To view the contents of the file, you can use an editor, such as the EDIT_FILE utility.

To print a copy of the file containing the CM statistics, enter:

```
/print_file f=$system.manuals.line_printer_manuals.cm_statistics
```

Summary of the CM Statistics

Within the CM statistics, the following statistic code subranges are defined (for more information on statistic codes, refer to NOS/VE Statistics: An Overview near the beginning of this chapter):

Statistic Code Range	Purpose
CM0000 to CM0999	Configuration control.
CM1000 to CM1999	Mainframe usage and failure data.
CM4000 to CM4999	Disk subsystem usage and failure data.
CM5000 to CM5999	Tape subsystem usage and failure data.
CM6000 to CM6999	External processor subsystem usage and failure data.
CM7000 to CM7999	Network device usage and failure data.

The following is a list of the statistics in numerical order:

Statistic Name	Description
Configuration Control:	
CM0	Identifies the elements used to initialize the system and the type of initialization performed.
CM1	Identifies the mainframe for which a system initialization has occurred.
CM2	Identifies the central processor on the mainframe being initialized.
CM3	Identifies the central memory on the mainframe being initialized.
CM4	Identifies an I/O unit on the mainframe being initialized.
CM5	Identifies a channel on the mainframe being initialized.
CM6	Identifies a peripheral in the active configuration.
CM50	Records a message stating that a system termination command has been processed.
CM51	Records a message stating that a system continuation command has been processed.
CM100	Identifies a peripheral storage device that has been initialized.
CM200	Records the change of state of a system hardware element.

Statistic Name	Description
Mainframe Usage and Failure:	
CM1100	Records the failure data captured by the system following corrected or uncorrected CPU errors.
CM1101	Records the failure data captured by the system following corrected or uncorrected central memory errors.
CM1102	Records the failure data captured by the system following corrected or uncorrected I/O unit errors.
CM1103	Records the failure data captured by the system following detection of a changing environment or of a power warning.
CM1104	Records the failure data stored in the mainframe element counters buffer.
CM1105	Records the failure data stored in the SECDED ID table.
CM1106	Records the beginning of a new hour.
CM1107	Records the failure data captured by the system following corrected or uncorrected page map errors; applies only to CYBER 930 mainframes.
CM1108	Records the failure data captured by the system following 6xx (DFT-or SCI-detected) critical error conditions.
CM1109	Records the failure data captured by the system following noncritical 4xx (bad requests to DFT) or 5xx (packet transmission errors) critical error conditions.
CM1110	Records disk errors from requests made to DFT.
CM1800	Records a variety of informative messages.
CM1900	Records usage of the Diagnostic Virtual System Utility (DVS).

Statistic Name	Description
Disk:	
CM4000	Records the amount of work done by a channel/controller combination.
CM4001	Records the amount of work done by a disk unit for a logging interval.
CM4002	Signals the end of the logging of disk path and disk usage data.
CM4100	Records the failure data captured by the system when accessing a 7154 disk subsystem.
CM4101	Records the failure data captured by the system when accessing a 7155-1x disk subsystem.
CM4102	Records the failure data captured by the system when accessing a 10395-11 disk controller.
CM4103	Records the failure data captured by the system when accessing a FA7B4-D disk controller.
CM4104	Records the failure data captured by the system when accessing a 7165-2x disk controller.
CM4105	Records the failure data captured by the system when accessing an 887-1 disk subsystem.
CM4106	Records the failure data captured by the system when accessing a 9836-1 or 9853-x disk subsystem.
Tape:	
CM5000	Records the amount of work done by a tape unit for a particular tape mount. This statistic is documented in the next section.
CM5100	Records the failure data captured by the system when accessing a 7021-3x tape subsystem.
CM5101	Records the failure data captured by the system when accessing a 7221-1 tape subsystem.
CM5102	Records the failure data captured by the system when accessing a 698-1x tape subsystem.
CM5103	Records the failure data captured by the system when accessing a 5698-1x tape subsystem.
CM5104	Records the failure data captured by the system when accessing a CTS/CCC subsystem.

Statistic Name	Description
External Processor:	
CM6100	Records the failure data captured by the system when accessing a 65354-1x MAP V subsystem.
Network Device:	
CM7000	Records the failure data captured by the system when attempting to access an integrated communications adapter (ICA).
CM7001	Records the failure data captured by the system when attempting to access a mainframe device interface (MDI).
CM7002	Records the usage data captured by the system when accessing an MDI.
CM7100	Records the failure data captured by the system resulting from an ICA transient hardware failure.
CM7101	Indicates that a nonfatal threshold value, set by the SET_ICA_PARAMETERS hardware function, has been reached.
CM7102	Indicates that an unsupported version of the microcode for the ICA board has just been loaded.
CM7103	Records information on the last ICA reset.
CM7104	Indicates that the EEPROM rewrite limit has been reached.
CM7105	Records statistical information captured by the system on ICA software activities.
CM7200	Records information captured by the system when a NAD PP driver encounters an abnormal condition.
CM7202	Records statistical information captured by the system on NAD operations.
CM7302	Records the failure data captured by the system when accessing a STORNET memory device. The File Server's PP reports initial failure data after the request retry count has been exhausted. This log entry provides the initial and final failure data for the I/O failure.
CM7303	Records the failure data captured by the system when accessing an ESM-II memory device. The File Server's PP reports initial failure data after the request retry count has been exhausted. This log entry provides the initial and final failure data for the I/O failure.
CM7400	Records the amount of work performed by a particular disk unit. This information is used by maintenance personnel to measure the failure data per amount of service performed.

CM 5000 Statistic

This statistic records tape usage. This information is used to measure the failure rate per amount of service performed. To aid in the isolation of failure, the external VSN of the tape volume is included in this statistic. This statistic is recorded each time a tape reel assigned to a job is dismounted by the system. Each channel/controller path used to access the unit is individually reported by a unique occurrence of this statistic.

The descriptive data field has the following format:

```
mainframe.IOUn.PPn.CHn.element.unit*vsn
```

Field	Description
mainframe	Records the name of the mainframe in the form $SYSTEM_ mmmm_ssss, where mmmm is the model number of central processor 0 (CP0) and ssss is the serial number of that processor.
IOUn	Identifies the I/O unit associated with the channel over which the usage occurred. The value n can be 0 or 1.
PPn	Records the PP that performed the I/O operation being reported. The value n is the PP number. If this was a concurrent PP, this field is CPPn.
CHn	Records the channel number over which the I/O request was processed. The value n is the channel number. If this was a concurrent channel, this field is CCHnp. The value p is the port (A or B) through which the tape device was accessed.
element	Records the element name of a 7021-3x controller, a 7221-1 adapter, or a 698-xx tape controller whose work is being reported.
unit	Records the element name of the tape device whose work is being reported.
vsn	Records the external VSN of the tape volume whose activity is being recorded.

The following counters are defined for the CM5000 statistic:

Counter	Description
1	Tape unit density selection in effect during the last read or write operation. The following codes are used: 1 (NRZI, 800 cpi) 2 (PE, 1600 cpi) 3 (GCR, 6250 cpi)
2	Total number of tape blocks written to tape by the job using this physical path. This count includes blocks that were written during failure recovery.
3	Total number of tape blocks read from tape by the job using this physical path. This count includes blocks which were read during failure recovery.
4	Total number of I/O requests (write and read requests) made by the job for this path to the tape unit.

Glossary A

Appendixes

Glossary A

A

Active Job

Job that is in memory and that has an active job list ordinal assigned to it. Compare with Initiated Job.

Application Scheduling Attribute

Job scheduling value that applies only to jobs that call certain site-selected application programs.

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

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.

D

Dispatching Priority

Readiness of a task for CPU dispatching. Specified as part of the DISPATCHING_CONTROL scheduling attribute, dispatching priorities are numbered P1 to P10. Within each priority, ready tasks are kept in a linked, ordered list. A ready task is linked into one of the dispatching control table (DCT) queues based on the task's dispatching priority.

Dynamic Dispatching

Task dispatching according to an algorithm that allows periodic adjustment of a job's dispatching priority. See also Dispatching Priority.

E

Estimated Ready Time

Current time plus the last calculated think time.

Estimated Think Time

Estimated length of time it takes the user to enter the next piece of input. Applies only to interactive jobs.

F

File Server

NOS/VE product that transfers requests and data between mainframes connected by STORNET or ESM-II extended memory devices.

G

Global Log

System-wide log that records statistics for all jobs on the system. See also Statistic.

Group Control Attribute

Value that helps establish boundaries that control how jobs are run on a specified mainframe or in a specified job class or service class.

Group Definition Attribute

Value that helps establish names or abbreviations for scheduling profile entities such as job classes or service classes.

Group Limit Attribute

Value that provides maximum limits for jobs assigned to that job class or service class.

Group Membership Attribute

Value that helps control assignment of a job to a mainframe or a job class.

Group Statistic Attribute

Value that provides access to statistical information about a job scheduling entity.

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.

Initiated Job

Job that has started execution but has not finished. It can be in one of two states: active or swapped-out. Compare with Active Job.

J

Job Category

Logical grouping of jobs that is based on characteristics important to the site.

Job Class

Name that defines a set of attributes assigned to a job. These attributes control the operation of the job during its input and initiation phases. For instance, the job class determines when a particular job is initiated. The default job classes used by NOS/VE are SYSTEM, MAINTENANCE, BATCH, INTERACTIVE, and UNASSIGNED.

Job Class Attribute

Job scheduling value that helps govern the input and initiation phases of job processing.

Job Working Set

Working set that consists of the pages a job is currently using. Pages that a job shares with other jobs are not part of a job working set. See also Shared Working Set.

L

Load Leveling

Automatic distribution of jobs between mainframes connected by the NOS/VE File Server. See also File Server.

Local Log

Log that records statistics emitted by a single job.

Long Wait

State entered by a job that expects to be inactive for a relatively long time and indicates this to the system. A typical long wait is when an interactive job has issued the interactive prompt to the terminal and is waiting for user-input.

M

Maintenance Attribute

System attribute used for purposes other than performance (such as system debugging).

Major Time Slice

Value that specifies, as a multiplier of the CPU_QUANTUM_TIME scheduling attribute, the amount of time a task can execute before losing its priority.

Memory Attribute

Value that defines limits and timed intervals related to memory management.

Minor Time Slice

Value that specifies, as a multiplier of the CPU_QUANTUM_TIME scheduling attribute, the maximum amount of time a task can use the CPU before a call is made to the CPU dispatcher to select a different task of equal or higher priority.

N

Nondynamic Dispatching

Task dispatching at a constant level.

O

Output Class Attribute

Job scheduling value that governs the disposition of job output files.

P

Page

Allocatable unit of real memory.

Page Aging

Process of determining the length of time since a page was last referenced. If a page has not been referenced within system-defined time intervals, a counter known as the page's age is incremented. If the age exceeds certain thresholds, the page is removed from the task's address space. NOS/VE employs a least-recently-used, page-aging algorithm.

Page Assign

Page assign occurs when the system responds to a page fault by assigning the task a page that did not previously exist on disk or in memory. In this case, the system assigns the new page to the task's address space in memory. See Page Fault, and compare with Page Reclaim and Page-In.

Page Caching

Removing pages from memory into a storage area that allows for more rapid access by the CPU than is possible in normal storage.

Page Fault

Hardware interrupt that occurs when a referenced page is not found in the system page table. This happens when a task references a page that is not in the task's address space. In response to a page fault, the system attempts to place the requested page in the task's address space. See Page Assign, Page-In, and Page Reclaim.

Page-In

Page-in occurs when the system responds to a page fault by reading the requested page from disk and transferring it into the task's address space in memory. See Page Fault, and compare with Page Reclaim and Page Assign.

Page Reclaim

Page reclaim occurs when the system responds to a page fault by obtaining the requested page from the available queue rather than by reading the page in from disk. The requested page is then assigned to the task's address space in memory. See Page Fault, and compare with Page Assign and Page-In.

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.

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.

Preemption

Swapping a job out of memory to make way for another job of higher priority.

Q

Queue Age Field

Number of seconds that a job has been in the input queue.

R

Reassignable Page

Page in the free or available page queue. A page in the available modified queue is not reassignable.

S

Scheduling Control Attribute

Job scheduling value that defines a processing characteristic applicable to all jobs regardless of their job class, service class, or job category. Examples are page aging scan times and memory limitations.

Service Class

Name that defines a set of attributes assigned to a job. These attributes govern the execution phase of job processing (including job swapping and determining dispatching priorities). The default service classes used by NOS/VE are SYSTEM, MAINTENANCE, BATCH, INTERACTIVE, and UNASSIGNED.

Service Class Attribute

Job scheduling value that helps govern the execution phase of job processing, including job swapping and determination of dispatching priorities.

Shared Queue

Queue containing shared working set pages based on the type of segment being shared. The six shared queues are task services (system code), executable files, nonexecutable files, device files, file server, and other (reserved for future use).

Shared Queue Attribute

Value that helps define the aging interval and size of a shared queue. Together, these attributes define how NOS/VE ages the pages of the shared working set. See also Shared Queue.

Shared Working Set

Working set consisting of pages that are shared by more than one job. Each page of a shared working set is assigned to one of six shared queues: task services (system code), executable files, nonexecutable files, device files, file server, and other (reserved for future use).

Statistic

Unit of data emitted in connection with the Statistics Facility. Typically, statistics record information about accounting, performance evaluation, and configuration management.

Swapping

Movement of jobs from central memory to disk or from disk to central memory. Swapping is done to maintain good system performance.

System Administrator

A user assigned a system administration capability.

See also System Job.

System Attribute

Value that allows users to select system options such as operational modes, table sizes, and file sizes. System attributes are specified on the SET_SYSTEM_ATTRIBUTE system core command.

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.

T

Target Memory

Value that specifies a level of free and available pages that the job scheduler attempts to keep in real memory.

Thrashing

Condition in which too much paging occurs in a system, causing system performance to degrade.

Thrashing Level

Value that specifies the minimum number of free and available pages needed in real memory to prevent system thrashing. This value is specified by the SCHEDULING_MEMORY_LEVELS scheduling attribute.

Transfer Unit

Segment division containing 16,384 bytes.

U

Used Bit

Flag that is set to indicate that a page has been used since the last time it was aged.

Utility

See Command Utility.

V

Virtual Address

Address of a location in virtual memory. The system hardware determines the physical location referenced by the virtual address.

Virtual Memory

Memory access mode using virtual addresses. Each task uses the same virtual address range; the system associates the virtual address range referenced by a task to the physical memory assigned to the task.

W

Working Set

Set of memory pages currently being used by the job. This set changes over time and is maintained by NOS/VE. See also Job Working Set and Shared Working Set.

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 NOS/VE System Usage manual provides 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 (table B-1 supplies the online titles). 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 Initialization Package (CIP) CYBER 180 Model 810, 830, 815, 825; CYBER 810A, 830A Computer Systems Reference Manual	60000417	
CYBER Initialization Package (CIP) CYBER 180 Model 835, 845, 855; CYBER 840, 850, 860 Computer Systems with IOU AB115A Reference Manual	60000418	
CYBER Initialization Package (CIP) CYBER 180 Model 845, 855; CYBER 840, 850, 860 with IOU AT478/AT481A; CYBER 840A, 850A, 860A, 870A, 990, 990E, 995E Computer Systems Reference Manual	60000419	
CYBER Initialization Package (CIP) CYBER 960, 994 Computer Systems Reference Manual	60000420	
CYBER Initialization Package (CIP) CYBER 962, 992 Computer Systems Reference Manual	60000421	
CYBER Initialization Package (CIP) CYBER 170 Model 865, 875; Non-Model 8XX/9XX Series Computer Systems Reference Manual	60000422	
Desktop/VE Host Utilities Usage	60463918	
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 2: Maintenance Usage	60463925	
NOS/VE User Validation Usage	60464513	
NOS/VE File Archiving Usage	60463944	
NOS/VE File Server For STORNET and ESM-II Usage	60000190	
NOS/VE Security Administration Usage	60463945	
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	
CYBIL for NOS/VE File Management Usage	60464114	EXAMPLES
CYBIL for NOS/VE System Interface Usage	60464115	EXAMPLES
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 Analysis Handbook	60459300	
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	
FORTRAN for NOS/VE Topics for FORTRAN Programmers Usage	60485916	
HPA/VE Reference	60461930	

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

NOS/VE Processing of Job Mode Software Errors D

NOS/VE Processing of Job Mode Software Errors D

Tasks running in job mode occasionally cause errors that are detected either by the hardware or the NOS/VE monitor. The action taken when one of these errors occurs is controlled by various system attributes set during NOS/VE deadstart with the SET_SYSTEM_ATTRIBUTE command.

Types of Errors

There are four types of job mode software errors:

- Broken task
- Monitor condition register (MCR) faults
- Unknown system requests
- Segment access faults

Broken Task

A broken task is a task in which the trap mechanism is not able to function correctly. NOS/VE monitor sends a broken task fault to the task and attempts to repair the trap mechanism. The specific cases of a broken task are:

Error Condition	Description
System error	Job mode software has declared the task is broken. This is a special case of a broken task.
Monitor fault buffer full	Job mode errors are occurring but are not being processed because the monitor fault buffer is full.
Traps disabled	A job mode error occurred while traps were disabled (the trap enable flag at bit 14 of word 2 in the job process state exchange package was not set).
Invalid A0	The task's dynamic space pointer (address register A0 of the job process state exchange package) was not valid.
UCR/MCR traps disabled	A user condition register/monitor condition register (UCR/MCR) error occurred while traps were disabled (the trap enable flag at bit 14 of word 2 in the job process state exchange package was not set).

The normal job mode action for a broken task is to exit.

MCR Fault

An MCR fault indicates that job mode caused a hardware-detected MCR fault. The normal job mode action for an MCR fault is to cause a condition fault to occur.

Unknown System Request

This type of error indicates that while the task was in job mode, the task issued a monitor request that was either invalid or had ring attributes different from the ring attributes NOS/VE monitor expected. The normal job mode action is to exit the task.

Segment Access Faults

Segment access faults indicate that one of the following errors has occurred during job mode:

- Page fault for an address greater than the end-of-information on a read only file or segment.
- A disk read error.

These errors either originate in NOS/VE monitor or cause the hardware to change from job mode to monitor mode. Depending on the values of system attributes, NOS/VE monitor halts or sends the error back to job mode as a monitor fault condition. At this point, the system core debugger can be activated. The normal job mode action is to cause a condition fault to occur.

System Attributes for Error Processing

The following system attributes are related to NOS/VE error processing. You can set and display them using the system core commands SET_SYSTEM_ATTRIBUTE and DISPLAY_SYSTEM_ATTRIBUTE. For more information about these commands and attributes, see chapter 2, Adjusting System Attributes.

DUMP_WHEN_DEBUG

When the system core debugger is called by a fault at or below SYSTEM_DEBUG_RING and the DUMP_WHEN_DEBUG attribute is specified as TRUE, the debugger automatically creates a dump of the task (see the description of the system core debugger FORMAT command). When the dump is complete, normal fault action takes place.

HALT_ON_HUNG_TASK

If this attribute is specified as 1, the occurrence of a hung task causes NOS/VE monitor to halt the system. If the attribute is specified as 0 (the default), the system sends a signal to the task to "hang" itself; that is, to go into an infinite wait in which it does nothing. Jobs with hung tasks are marked on the operator CP display by the characters *H in the status field. A hung task also results if any error occurs in job mode ring 1.

HALTRING

If a broken task or MCR fault occurs at or below the value specified for the HALTRING attribute (P register ring number), NOS/VE monitor halts the system. Broken tasks occurring above the HALTRING value cause a monitor fault to be sent back to job mode.

SYSTEM_DEBUG_RING

If an error (broken task, MCR fault, unknown system request, or segment access fault) occurs at or below the value of this attribute (P register ring number), the system core debugger is called within the task. At that point in time, the task environment can be examined by using system core debugger commands.

SYSTEM_DEBUG_SEGMENT

If an error occurs which meets the criteria defined by SYSTEM_DEBUG_RING, the system compares the P register segment number with the current value of SYSTEM_DEBUG_SEGMENT to determine if the system core debugger should be called. The use of this attribute allows you to limit calls to the system core debugger to operating system segments only. The debugger is called if either of the following is true:

- The value of SYSTEM_DEBUG_SEGMENT is 0 (the default).
- The P register segment number is equal to or less than the value of SYSTEM_DEBUG_SEGMENT.

SYSTEM_ERROR_HANG_COUNT

This attribute specifies the number of broken task errors allowed to occur in a given task before that task is considered hung.

Index

Comments (continued from other side)

Please fold on dotted line;
seal edges with tape only.

FOLD

FOLD

FOLD

CONTROL DATA
Technical Publications
ARH219
4201 N. Lexington Avenue
Arden Hills, MN 55126-9983

NOS/VE System Performance and Maintenance, Volume 1 **60463915 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 2, are followed by II. 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