NetBatch Manual

Abstract

This manual describes NetBatch, an HP NonStop™ software product that automates job scheduling, startup, and management on HP NonStop S-series system and HP Integrity NonStop NS-series system. The manual contains a product overview, software installation instructions, and planning, setup, and usage guidelines for NetBatch objects. The manual also contains descriptions of NetBatch commands and object attributes.

Product Version

NetBatch D30.00

NetBatch H01

Supported Release Version Updates (RVUs)

This publication supports D20.00 and all subsequent D-series RVUs, G02.00 and all subsequent G-series RVUs, and H06.03 and all subsequent H-series RVUs until otherwise indicated by its replacement publication.

Part Number	Published
522460-004	January 2007

Document History

Part Number	Product Version	Published
142530	NetBatch D30	August 1998
522460-001	NetBatch D30	February 2002
522460-002	NetBatch D30	May 2002
522460-003	NetBatch D30	September 2005
522460-004	NetBatch D30, H01	January 2007

NetBatch Manual

Glossary | Index | Figures | Tables

What's New in This Manual xiii
- Manual Information xiii
- New and Changed Information xiii

About This Manual xvii
- Audience xvii
- Organization xvii
- Further Reading xix
- Notation Conventions xx
- Abbreviations xxiv

1. NetBatch Introduction

Product Overview 1-1
- NetBatch Jobs 1-1
- Executor Programs 1-2
- Network Capabilities 1-2
- CPU Assignment 1-2
- ASSIGNs, DEFINEs, and PARAMs 1-3
- Operator Interface 1-3
- Scheduling Options 1-3
- Scheduler Logging 1-3
- Programmatic Interfaces 1-3
- Spooler Jobs 1-3

Core Components 1-4
- NETBATCH 1-4
- BATCHCOM 1-4
- BATCHCAL 1-4
- NBEXEC 1-4
- NB^JOB^SUBMIT 1-5

Scheduler Functions 1-6
- CPU Assignment 1-6
- Job Storage 1-7

1. NetBatch Introduction (continued)

Job Scheduling and Startup 1-8
Job Tracking, Control, and Termination 1-9
NetBatch High-PIN Capabilities 1-12

2. Software Installation

Installing NetBatch Software—Overview 2-1
NetBatch Product Files 2-1
Installation Prerequisites 2-2
Installation Procedure 2-3
Step 1: Log On as the Super ID 2-3
Step 2: Shut Down NETBATCH and Stop Its Processes 2-4
Step 3: Receive NetBatch Product Files 2-4
Step 4: Export and Print the NetBatch Softdoc 2-4
Step 5: Perform Pre-installation Tasks 2-4
Step 6: Perform Installation-Related Tasks 2-4
Step 7: Perform Post-installation Tasks 2-5
Step 8: Restart Schedulers 2-7
Migrating a Scheduler Database 2-7
Step 1: Log on as the Super ID (255,255). 2-7
Step 2: Shut Down the Scheduler 2-7
Step 3: Copy the Database Files to the Target Location 2-8
Step 4: Secure the Migrated JOB File to Local Super ID 2-8
Step 5: Update Alternate Key File References 2-9
Step 6: Warm Start Scheduler 2-9
Step 7: Verify and Alter Attributes 2-9

3. Scheduler Planning, Configuration, and Management

Planning Schedulers 3-1
Task 1. Compile Job Information 3-1
Task 2. Determine Scheduler Numbers and Attributes 3-2
Task 3. Establish Class and Executor Configurations 3-6
Starting Schedulers 3-9
Overview 3-9
Running NETBATCH 3-10
Cold Starting a Scheduler 3-15
Warm Starting a Scheduler 3-18
Warm-Start Events 3-21
Configuring Schedulers 3-23

3. Scheduler Planning, Configuration, and Management (continued)

Scheduler Configuration Commands 3-23
Scheduler Configuration Procedure 3-23
Displaying Scheduler Configuration 3-25
Automating Scheduler Configuration 3-26
Managing Schedulers 3-29
Determining Whether a Scheduler Is Running 3-29
Displaying Scheduler, Executor, and Class Status 3-29
Altering Executor and Class Attributes 3-33
Stopping and Restarting Executors 3-34
Deleting Executors and Classes 3-35
Dealing With Scheduler Log Files 3-38
Enabling and Disabling EMS Event-Message Generation 3-44
Stopping a Scheduler 3-45

4. Job Planning, Submission, and Management

Planning Jobs 4-1
Planning Procedure 4-1
Creating a Job Input File 4-7
Using Completion Codes to Test Process Termination Status 4-7
Identifying Processes of a Job 4-10
ZBAT:JOBINFO 4-12
Submitting Jobs 4-13
Submitting a Job With the SUBMIT JOB Command 4-13
Specifying a Job's Class and Selection Priority 4-19
Specifying a Job's Executor Program and the Program's Run Options and Parameters 4-20
Specifying a Job's Input File, Defaults, and PURGE-IN-FILE Attribute 4-21
Specifying a Job's Output File, Maximum Print Lines or Maximum Print Pages, and Log File 4-22
Scheduling a Job 4-23
Specifying a Job's Hold Characteristics 4-26
Controlling a Job's Behavior on Process Failure 4-28
Specifying a Job's Tape Drives Requirements 4-30
Specifying a Job's Dependencies 4-31
Specifying a Job's ASSIGNs, DEFINEs, and PARAMs 4-34
Specifying a Job's Description 4-40
Managing Jobs 4-42

4. Job Planning, Submission, and Management (continued)

Displaying Job Status 4-42
Altering Job Attributes 4-44
Overriding Job Dependencies, Timing Attributes, and Selection Priority 4-44
Suspending and Reactivating Job Processes 4-45
Stopping and Deleting Jobs 4-46
Dealing With Job Output 4-47

5. Run Calendar Generation and Display

Example 5-1
Running BATCHCAL 5-2
BATCHCAL's High PIN Capabilities 5-4
Entering Source Data 5-5
Entering Source Data in a File Versus Entering It Interactively 5-5
Formatting Source Data 5-6
Using BATCHCAL Commands 5-13
== Command 5-13
EXIT Command 5-13
FC Command 5-13
HELP Command 5-14
Generating a Run Calendar 5-14
Generating a Run Calendar From an EDIT Source File 5-14
Generating a Run Calendar During an Interactive BATCHCAL Session 5-15
Displaying Run Times 5-16
Displaying Run Times in List Form 5-17
Displaying Run Times in Chart Form 5-19
Reformatting an Old Calendar File 5-22

6. Commands

Running BATCHCOM 6-1
Considerations 6-2
BATCHCOM's High PIN Capabilities 6-6
Keywords 6-7
Keyword Types 6-7
Keyword Abbreviations 6-9
Keyword and Command Aliases 6-11
Command Reference Summary 6-11
Command to Run NETBATCH 6-11
Command to Run BATCHCOM 6-11

6. Commands (continued)

Attachment-Set Commands 6-12
Class Commands 6-13
Executor Commands 6-14
Job Commands 6-15
Scheduler Commands 6-20
Other Commands 6-22
Command Security 6-23
Command Descriptions 6-28
Wild-Card Characters 6-29
ABORT SCHEDULER Command 6-30
ACTIVATE JOB Command 6-32
ADD ATTACHMENT-SET Command 6-34
ADD CLASS Command 6-40
ADD EXECUTOR Command 6-42
ADD SCHEDULER Command 6-46
ALLOW ERRORS Command 6-51
ALTER ATTACHMENT-SET Command 6-52
ALTER CLASS Command 6-58
ALTER EXECUTOR Command 6-59
ALTER JOB Command 6-62
ALTER SCHEDULER Command 6-67
ASSUME ATTACHMENT-SET Command 6-69
ASSUME CLASS Command 6-70
ASSUME EXECUTOR Command 6-71
ASSUME JOB Command 6-72
ASSUME SCHEDULER Command 6-73
CHANGEUSER Command 6-74
COMMENT Command 6-76
DELETE ATTACHMENT-SET Command 6-77
DELETE CLASS Command 6-81
DELETE EXECUTOR Command 6-82
DELETE JOB Command 6-84
DISPLAY-SPI Command 6-86
EXIT Command 6-92
FC Command 6-92
HELP Command 6-95
HISTORY Command 6-98
INFO ATTACHMENT-SET Command 6-99

6. Commands (continued)

INFO CLASS Command 6-103
INFO EXECUTOR Command 6-104
INFO JOB Command 6-106
INFO SCHEDULER Command 6-110
OBEY Command 6-112
OPEN Command 6-113
RELEASE-WAITON Command 6-115
REPORT JOB Command 6-116
RESET ATTACHMENT-SET Command 6-117
RESET CLASS Command 6-119
RESET EXECUTOR Command 6-120
RESET JOB Command 6-121
RESET SCHEDULER Command 6-122
RUN Command 6-124
RUNNEXT JOB Command 6-125
RUNNOW JOB Command 6-128
SET ATTACHMENT-SET Command 6-130
SET CLASS Command 6-133
SET EXECUTOR Command 6-134
SET JOB Command 6-136
SET SCHEDULER Command 6-140
SHOW ATTACHMENT-SET Command 6-142
SHOW CLASS Command 6-145
SHOW EXECUTOR Command 6-146
SHOW JOB Command 6-147
SHOW SCHEDULER Command 6-150
SHUTDOWN SCHEDULER Command 6-152
START EXECUTOR Command 6-155
START SCHEDULER Command 6-157
STATUS ATTACHMENT-SET Command 6-160
STATUS EXECUTOR Command 6-163
STATUS JOB Command 6-165
STATUS SCHEDULER Command 6-173
STATUS-HISTORY Command 6-175
STOP EXECUTOR Command 6-178
STOP JOB Command 6-180
SUBMIT JOB Command 6-183
SUSPEND JOB Command 6-190

6. Commands (continued)

SWITCHCPU SCHEDULER Command 6-192
SWITCHLOG SCHEDULER Command 6-193
SYSTEM Command 6-195
VOLUME Command 6-196
! Command 6-198
? Command 6-200

7. Attributes

Attribute Reference Summary 7-1
Attachment-Set Attributes 7-1
Class Attribute 7-1
Executor Attributes 7-2
Job Attributes 7-2
Scheduler Attributes 7-6
Attribute Descriptions 7-8
AFTER Job Attribute 7-9
ASSIGN Attachment-Set Attribute 7-12
AT Job Attribute 7-15
AT-ALLOWED Scheduler Attribute 7-18
ATTACHMENT-SET Job Attribute 7-19
BACKUPCPU Scheduler Attribute 7-23
CALENDAR Job Attribute 7-25
CATCHUP Scheduler Attribute 7-28
CLASS Executor Attribute 7-30
CLASS Job Attribute 7-32
CPU Executor Attribute 7-34
DEFAULT-CLASS Scheduler Attribute 7-36
DEFAULT-EXECUTOR-PROGRAM Scheduler Attribute 7-38
DEFAULT-HIGHPIN Scheduler Attribute 7-39
DEFAULT-MAXPRINTLINES Scheduler Attribute 7-40
DEFAULT-MAXPRINTPAGES Scheduler Attribute 7-41
DEFAULT-OUT Scheduler Attribute 7-42
DEFAULT-PRI Scheduler Attribute 7-43
DEFAULT-SELPRI Scheduler Attribute 7-44
DEFAULT-STALL Scheduler Attribute 7-45
DEFAULT-STOP-ON-ABEND Scheduler Attribute 7-46
DEFINE Attachment-Set Attribute 7-47
DESCRIPTION Job Attribute 7-52

7. Attributes (continued)

EMS Scheduler Attribute 7-54
EVERY Job Attribute 7-55
EXECUTOR-PROGRAM Job Attribute 7-58
EXTSWAP Job Attribute 7-60
HIGHPIN Job Attribute 7-61
HOLD Job Attribute 7-62
HOLDAFTER Job Attribute 7-63
IFFAILS Job Attribute 7-64
IN Job Attribute 7-66
INITIATION Class Attribute 7-68
INITIATION Scheduler Attribute 7-69
JOB-LOG Job Attribute 7-70
JOBID-ZERO Job Attribute 7-73
LIB Job Attribute 7-74
LIMIT Job Attribute 7-75
LOCALNAMES Scheduler Attribute 7-77
MAX-CONCURRENT-JOBS Scheduler Attribute 7-79
MAX-PRI Scheduler Attribute 7-81
MAXPRINTLINES Job Attribute 7-82
MAXPRINTPAGES Job Attribute 7-83
MEM Job Attribute 7-85
NAME Job Attribute 7-86
OUT Job Attribute 7-87
PARAM Attachment-Set Attribute 7-89
PFS Job Attribute 7-91
PRI Job Attribute 7-92
PURGE-IN-FILE Job Attribute 7-93
RESTART Job Attribute 7-95
RUND Job Attribute 7-97
SAVEABEND Job Attribute 7-98
SECURITY Attachment-Set Attribute 7-99
SELPRI Job Attribute 7-101
STALL Job Attribute 7-102
STARTUP Job Attribute 7-104
STOP-ON-ABEND Job Attribute 7-105
SUBMIT-ALLOWED Scheduler Attribute 7-107
SWAP Job Attribute 7-108
TAPEDRIVES Job Attribute 7-109

7. Attributes (continued)

TAPEDRIVES Scheduler Attribute 7-110
TEMPORARY Attachment-Set Attribute 7-112
TERM Job Attribute 7-114
VOLUME Job Attribute 7-116
WAIT Job Attribute 7-118
WAITON Job Attribute 7-119

A. Messages

Message Types A-2
Error Messages A-2
Informational Messages A-2
Warning Messages A-2
Message Descriptions A-3

B. NBEXEC

Introducing NBEXEC B-1
TACL Alternative B-1
Application Design Tool B-1
Improved Resource Usage B-1
Automatic Restart B-1
Log Files B-2
NBEXEC Processing B-2
High PIN Capabilities B-2
Command and Variable Reference Summary B-3
Command Identifiers B-3
NBEXEC-Defined Variables B-4
NBEXEC Syntax Summary B-5
Command to Run NBEXEC B-5
Control-file ($) Commands B-6
Command-Interpreter (:) Commands B-7
Other Commands B-10
NBEXEC-Defined Logical Variables B-10
NBEXEC-Defined String Variables B-10

C. National Language Support

Changing BATCHCOM Keywords and Messages C-1
Step 1: Log On as the Super ID C-1
Step 2: Make a Backup Copy of BATCHLIB C-1
Step 3: Extract Keywords and Messages From BATCHLIB C-1
Step 4: Change Keywords and Messages in EDIT Source File C-2
Step 5: Convert EDIT Source File to TAL Source File C-6
Step 6: Compile TAL Source File C-6
Step 7: Bind Compiled Object Into BATCHLIB C-6
Step 8: Assign Updated BATCHLIB File to BATCHCOM C-7
Step 9: Add DEFINE =_ZBAT_NLS to the TACL Environment C-7

Glossary

Index

Figures

Figure 1-1. A NetBatch Job 1-2
Figure 1-2. NetBatch Environment 1-5
Figure 1-3. Sample Job, Class, Executor, and CPU Relationships 1-7
Figure 1-4. Scheduling Algorithm 1-10
Figure 2-1. NetBatch Software Installation Overview 2-3
Figure 2-2. NBFLAGS Bit Settings 2-5
Figure 3-1. Example of a Job Table for Scheduler Planning 3-2
Figure 3-2. Example of a Scheduler Planning Table 3-5
Figure 3-3. Example of a Class and Executor Planning Table 3-8
Figure 3-4. Example of ALTER EXECUTOR Command 3-34
Figure 3-5. Example of DELETE EXECUTOR Command 3-36
Figure 3-6. Example of DELETE CLASS Command 3-37
Figure 3-7. Example of a Scheduler Log File 3-38
Figure 4-1. Sample Programs and Run Environments Diagram 4-3
Figure 4-2. Sample Job Dependencies Diagram 4-4
Figure 4-3. Sample Jobs Planning Table 4-6
Figure 4-4. Sample Input File 4-7
Figure 4-5. Sample #CASE and #IF Constructs for Completion Code Testing 4-9
Figure 4-6. Sample Job Input File Containing #CASE Statements 4-9
Figure 4-7. Example of GMOMJOBID Propagation 4-10
Figure 4-8. Example of Dissociated Processes 4-11
Figure 4-9. Decision Chart for Setting IFFAILS, RESTART, STALL, and STOP-ON-ABEND Attributes 4-29

Figures (continued)

Figure 4-10. Sample #IF Statements for ZBAT:RELEASE Error 4-33
Figure 4-11. Sample Job Log File 4-48
Figure 4-12. Example of Executor-Program Output 4-49
Figure 4-13. Example of Nonexecutor-Program Output 4-49
Figure 6-1. Examples of Attribute Defaulting-Attachment Set 6-36
Figure 6-2. Example of Attribute Defaulting-Class 6-40
Figure 6-3. Example of Attribute Defaulting-Executor 6-43
Figure 6-4. Example of Attribute Defaulting-Scheduler 6-48
Figure 6-5. Status JOB..., DETAIL Command Display for Jobs That Are Not Executing, Over Limit, or Suspended 6-168
Figure 6-6. JOB..., DETAIL C ommand- Display for Jobs That Are Not Executing, Over Limit, or Suspended 6-169
Figure 6-7. Example of Attribute Defaulting-Job 6-186

Tables

Table 1-1. Scheduling Algorithm Description 1-11
Table 1-2. NetBatch D20 High PIN Capabilities 1-12
Table 2-1. NetBatch Subvolumes 2-1
Table 2-2. NetBatch Product Files 2-1
Table 2-3. NetBatch Installation Prerequisites 2-2
Table 3-1. Scheduler Attributes 3-4
Table 3-2. High PIN Capabilities in NETBATCH 3-14
Table 3-3. Scheduler Log File Keywords 3-41
Table 3-4. Scheduler Log File Logging Formats 3-43
Table 4-1. Sample Job Category Table 4-2
Table 4-2. Sample Job-Recovery Strategies 4-5
Table 4-3. Completion Codes 4-8
Table 4-4. Job Attributes 4-15
Table 4-5. Job States on Process Failure 4-28
Table 4-6. Attachment-Set Attributes 4-36
Table 4-7. Job State 4-42
Table 5-1. High PIN Capabilities in BATCHCAL 5-4
Table 5-2. Displaying Run Times in List Form 5-17
Table 5-3. Displaying Run Times in Chart Form 5-19
Table 5-4. Reformatting an Old Calendar File 5-22
Table 6-1. High PIN Capabilities in BATCHCOM 6-6
Table 6-2. Command Security 6-24
Table 6-3. Attachment-Set Defaults 6-35

Tables (continued)

Table 6-4. Executor Defaults 6-43
Table 6-5. Scheduler Defaults 6-47
Table 6-6. Files Created or Initialized by ADD SCHEDULER Command 6-48
Table 6-7. Executor States 6-164
Table 6-8. Job States 6-169
Table 6-9. Job Defaults 6-185
Table 7-1. DEFINE Types (Classes) 7-48
Table 7-2. DEFINE Attributes 7-48
Table 7-3. Attachment-Set Access 7-100

What's New in This Manual

Manual Information

Abstract

This manual describes NetBatch, an HP NonStop™ software product that automates job scheduling, startup, and management on HP NonStop S-series system and HP Integrity NonStop NS-series system. The manual contains a product overview, software installation instructions, and planning, setup, and usage guidelines for NetBatch objects. The manual also contains descriptions of NetBatch commands and object attributes.

Product Version

NetBatch D30.00

NetBatch H01

Supported Release Version Updates (RVUs)

This publication supports D20.00 and all subsequent D-series RVUs, G02.00 and all subsequent G-series RVUs, and H06.03 and all subsequent H-series RVUs until otherwise indicated by its replacement publication.

Part Number	Published
522460-004	January 2007

Document History

Part Number	Product Version	Published
142530	NetBatch D30	August 1998
522460-001	NetBatch D30	February 2002
522460-002	NetBatch D30	May 2002
522460-003	NetBatch D30	September 2005
522460-004	NetBatch D30, H01	January 2007

New and Changed Information

Changes to the G06.30 manual:

- Updated the range for the WAITON job attribute on pages 1-3, 4-18, 4-31, 7-5, 7-119, and A-34.
- Added the description and examples for logging information greater than 132 bytes under Interpreting the Contents of a Scheduler Log File on page 3-40

- Changed the range of job numbers for systems running G-series RVUs on pages 4-13, 6-187, A-23, and Glossary-5.
- Changed the consideration on DST explaining how next-runtime is calculated after DST on page 6-188.
- Added the error message 2110-E on page A-25.

Changes to the G06.27 Manual

- Updated the TACL RUN command with the `REPORT` and `jobrun-database-subvol` attributes under Running NETBATCH on page 3-10.
- Added the description for:
  - REPORT-ON on page 3-12
  - jobrun-database-subvol on page 3-13
- Added the synopsis and syntax summary of the REPORT JOB under Job Commands on page 6-17.
- Added the new command, REPORT JOB, under Command Security 6-26.
- Added the description for REPORT JOB Command on page 6-116.

Changes to the G06.16 Manual

- Updated Section 2, Software Installation, to discuss use of DSM/SCM instead of Install.
- Added a consideration regarding when the scheduler INITIATION attribute is OFF to RUNNOW JOB Command on page 6-128
- Updated the descriptions of the READY and RUNNOW job states in Table 6-8, Job States, on page 6-169
- Added scheduler INITIATION attribute information to the considerations in SUBMIT JOB Command on page 6-183
- Added the message text to warning 0548-W on page A-12
- Added message 0549-I on page A-12

Changes to the G06.15 Manual

- Added new events in Section 3, Scheduler Planning, Configuration, and Management
- Changed the use of the RUNNOW command in Section 7, Attributes
- Changed the scheduler behavior when a job with the EVERY attribute is held off in Section 7, Attributes

- Changed the BATCHCAL default
- Added a new error to Appendix A, Messages

About This Manual

This manual describes NetBatch, an HP NonStop software product that automates job scheduling, startup, and management on HP NonStop systems. The manual contains a product overview, software installation instructions, and planning, setup, and usage guidelines for NetBatch objects. The manual also contains descriptions of NetBatch commands and object attributes.

Audience

The intended audience for this manual includes NetBatch users who do any of:

- Plan, generate, maintain, and control use of the NetBatch product
- Design, write, and test jobs
- Schedule jobs, and set up and maintain job dependencies
- Submit jobs, and monitor and manage batch runs

Audience prerequisites include:

- A good working knowledge of the HP NonStop operating system and basic system operations
- Previous exposure to batch processing software

Organization

Section	Description
Section 1, NetBatch Introduction	Contains an overview of the NetBatch product, and lists and describes core NetBatch components. It also describes functions of the NetBatch scheduler and explains the scheduling algorithm.
Section 2, Software Installation	Explains NetBatch software installation, and lists and describes NetBatch product files. It also describes how to migrate a scheduler database from one location to another.
Section 3, Scheduler Planning, Configuration, and Management	Documents a scheduler planning process and explains how to cold start, warm start, and configure schedulers. It also describes how to perform scheduler-management tasks.
Section 4, Job Planning, Submission, and Management	Documents a job planning process and explains how to submit a job to a scheduler. It also describes how to perform job management tasks.
Section 5, Run Calendar Generation and Display	Introduces BATCHCAL run calendars and explains their use in job scheduling. It also lists the procedures for generating run calendars, displaying run times, and reformatting old calendar files to the current format.

Section	Description
Section 6, Commands	Describes and gives examples of the syntax, operation, and results of all BATCHCOM commands.
Section 7, Attributes	Describes and gives examples of the syntax, operation, and results of all attachment-set, class, executor, job, and scheduler attributes.
Appendix A, Messages	Lists NetBatch error, informational, and warning messages, and gives cause, effect, and recovery information for those messages.
Appendix B, NBEXEC	Presents a management-level synopsis of NBEXEC, the NetBatch NonStop executor program. It also supplies a quick-reference guide to NBEXEC command and variable syntax.
Appendix C, National Language Support	Explains how to change BATCHCOM keywords and messages to suit your operational environment.

This chart indicates the intended audience for each section and appendix:

	For users who...			
Section/Appendix	**Plan, generate, maintain, and control the NetBatch system**	**Design, write, and test jobs**	**Schedule jobs, and set up and maintain job dependencies**	**Submit jobs, and monitor and manage batch runs**
Section 1, NetBatch Introduction	X	X	X	X
Section 2, Software Installation	X			
Section 3, Scheduler Planning, Configuration, and Management	X	X		X
Section 4, Job Planning, Submission, and Management		X	X	X
Section 5, Run Calendar Generation and Display		X	X	X
Section 6, Commands	X	X	X	X
Section 7, Attributes	X	X	X	X
Appendix A, Messages	X	X	X	X
Appendix B, NBEXEC		X		
Appendix C, National Language Support	X			

Further Reading

This manual contains references to:

Manual	Description
Debug Manual	Describes the Guardian debug facility on TNS and TNS/R systems
DSM Template Services Manual	Describes Distributed Systems Management (DSM) Template Services, which support the representation of SPI buffers in display text
DSM/SCM User's Guide	Describes how to plan for and install a new software release or software product revision (SPR) using DSM/SCM. It describes the overall installation process, the usage of DSM/SCM, and the specific tasks required to install system software
EDIT User's Guide and Reference Manual	Describes the syntax of EDIT commands, explains how to create and use EDIT files, and describes EDIT versus page-mode editing
EMS Manual	Describes EMS, a collection of processes, tools, and interfaces that provide event-message collection and distribution in the DSM environment
File Utility Program (FUP) Reference Manual	Describes the syntax of all FUP commands and includes explanations of FUP error messages
Guardian Procedure Calls Reference Manual	Describes all system procedure calls
Guardian Procedure Errors and Messages Manual	Describes system procedure error codes and error lists, system messages, traps, and the trap error list
Guardian Programmer's Guide	Describes how to use the Guardian procedure call interface to access system services from an application program
Guardian User's Guide	Describes basic operating-system tasks
Inspect Manual	Describes the Inspect interactive symbolic debugger for TNS/R and TNS/E systems. The manual is intended for system and application programmers.
NetBatch Management Programming Manual	Describes the Distributed Systems Management (DSM) programmatic interfaces (commands, responses, and event messages in Subsystem Programmatic Interface (SPI) format) to the NetBatch subsystem
NetBatch-Plus Reference Manual	Describes NetBatch-Plus, a Pathway application that provides a screen-driven NetBatch interface
Safeguard Reference Manual	Describes the Safeguard distributed security management facility and the syntax of the commands of the SAFECOM command interpreter

Manual	Description
SPI Programming Manual	Describes the Subsystem Programmatic Interface and tells how to use it in management applications and subsystems you write
System Generation Manual	Describes software installation and system generation
TACL Programming Guide	Presents examples of HP Tandem Advanced Command Language (TACL) macros and routines
TACL Reference Manual	Describes the syntax, operation, and results of all TACL commands, functions, built-in functions, and built-in variables

Notation Conventions

Hypertext Links

Blue underline is used to indicate a hypertext link within text. By clicking a passage of text with a blue underline, you are taken to the location described. For example:

> This requirement is described under Backup DAM Volumes and Physical Disk Drives on page 3-2.

General Syntax Notation

This list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS. Uppercase letters indicate keywords and reserved words; enter these items exactly as shown. Items not enclosed in brackets are required. For example:

```
MAXATTACH
```

lowercase italic letters. Lowercase italic letters indicate variable items that you supply. Items not enclosed in brackets are required. For example:

file-name

computer type. `Computer type` letters within text indicate C and Open System Services (OSS) keywords and reserved words; enter these items exactly as shown. Items not enclosed in brackets are required. For example:

```
myfile.c
```

italic computer type. *`Italic computer type`* letters within text indicate C and Open System Services (OSS) variable items that you supply. Items not enclosed in brackets are required. For example:

pathname

[ ] Brackets. Brackets enclose optional syntax items. For example:

```
TERM [\system-name.]$terminal-name

INT[ERRUPTS]
```

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list may be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
FC [ num  ]
   [ -num ]
   [ text ]

K [ X | D ] address
```

{ } Braces. A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list may be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LISTOPENS PROCESS { $appl-mgr-name }
                  { $process-name  }

ALLOWSU { ON | OFF }
```

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
INSPECT { OFF | ON | SAVEABEND }
```

… Ellipsis. An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
M address [ , new-value ]...

[ - ] {0|1|2|3|4|5|6|7|8|9}...
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
"s-char..."
```

Punctuation. Parentheses, commas, semicolons, and other symbols not previously described must be entered as shown. For example:

```
error := NEXTFILENAME ( file-name ) ;

LISTOPENS SU $process-name.#su-name
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must enter as shown. For example:

```
"[" repetition-constant-list "]"
```

Item Spacing. Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
CALL STEPMOM ( process-id ) ;
```

If there is no space between two items, spaces are not permitted. In these examples, there are no spaces permitted between the period and any other items:

```
$process-name.#su-name
```

Line Spacing. If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
ALTER [ / OUT file-spec / ] LINE

   [ , attribute-spec ]...
```

!i and !o. In procedure calls, the !i notation follows an input parameter (one that passes data to the called procedure); the !o notation follows an output parameter (one that returns data to the calling program). For example:

```
CALL CHECKRESIZESEGMENT (  segment-id                    !i
                         , error        ) ;              !o
```

!i,o. In procedure calls, the !i,o notation follows an input/output parameter (one that both passes data to the called procedure and returns data to the calling program). For example:

```
error := COMPRESSEDIT ( filenum ) ;                      !i,o
```

!i:i. In procedure calls, the !i:i notation follows an input string parameter that has a corresponding parameter specifying the length of the string in bytes. For example:

```
error := FILENAME_COMPARE_ (  filename1:length           !i:i
                            , filename2:length ) ;       !i:i
```

!o:i. In procedure calls, the !o:i notation follows an output buffer parameter that has a corresponding input parameter specifying the maximum length of the output buffer in bytes. For example:

```
error := FILE_GETINFO_ (  filenum                        !i
                        , [ filename:maxlen ] ) ;        !o:i
```

Notation for Messages

list summarizes the notation conventions for the presentation of displayed messages in this manual.

Bold Text. Bold text in an example indicates user input entered at the terminal. For example:

```
ENTER RUN CODE
?123
CODE RECEIVED:      123.00
```

The user must press the Return key after typing the input.

Nonitalic text. Nonitalic letters, numbers, and punctuation indicate text that is displayed or returned exactly as shown. For example:

```
Backup Up.
```

lowercase italic letters. Lowercase italic letters indicate variable items whose values are displayed or returned. For example:

p-register

process-name

[ ] Brackets. Brackets enclose items that are sometimes, but not always, displayed. For example:

Event number = *number* [Subject = *first-subject-value*]

A group of items enclosed in brackets is a list of all possible items that can be displayed, of which one or none might actually be displayed. The items in the list might be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

proc-name trapped [in SQL | in SQL file system]

{ } Braces. A group of items enclosed in braces is a list of all possible items that can be displayed, of which one is actually displayed. The items in the list might be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

obj-type obj-name state changed to *state*, caused by
{ Object | Operator | Service }

process-name State changed from *old-objstate* to *objstate*
{ Operator Request. }
{ Unknown. }

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

Transfer status: { OK | Failed }

% Percent Sign. A percent sign precedes a number that is not in decimal notation. The % notation precedes an octal number. The %B notation precedes a binary number. The %H notation precedes a hexadecimal number. For example:

```
%005400

%B101111

%H2F

P=%p-register E=%e-register
```

Change Bar Notation

Change bars are used to indicate substantive differences between this edition of the manual and the preceding edition. Change bars are vertical rules placed in the right margin of changed portions of text, figures, tables, examples, and so on. Change bars highlight new or revised information. For example:

> The message types specified in the REPORT clause are different in the COBOL environment and the Common Run-Time Environment (CRE).
>
> The CRE has many new message types and some new message type codes forold message types. In the CRE, the message type SYSTEM includes all messages except LOGICAL-CLOSE and LOGICAL-OPEN.

Abbreviations

The glossary of this manual includes abbreviations.

1 NetBatch Introduction

This section introduces the NetBatch product:

Topic	Page
Product Overview	1-1
Core Components	1-4
Scheduler Functions	1-6
NetBatch High-PIN Capabilities	1-12

Product Overview

NetBatch is an HP NonStop software product that lets your organization automate job scheduling, startup, and management on NonStop systems. The product:

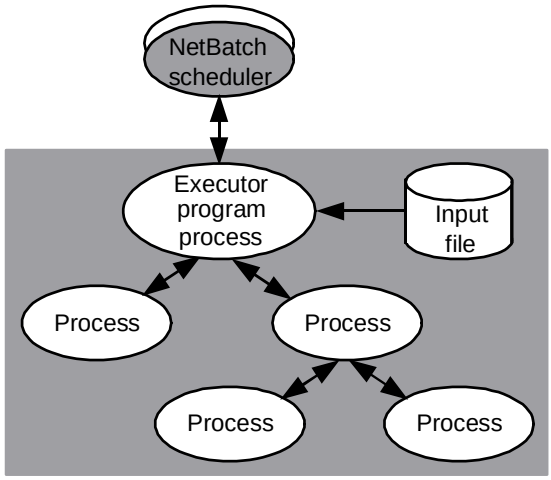
- Complements the systems' online transaction processing (OLTP) capabilities by providing automated batch processing facilities on those systems
- Increases job throughput by enabling job distribution among the systems' CPUs
- Frees staff for other work by reducing the need for user intervention in jobs

NetBatch Jobs

A NetBatch job is a process or sequence of processes that performs specified tasks. For example, a job could start an online Pathway application, summarize and post end-of-period results, or back up data files.

All NetBatch jobs have an executor program and, depending on the program, an input file. The input file contains commands executed by the executor program, which the NetBatch scheduler starts as the job's initial process. The executor-program process can in turn start other processes. Figure 1-1 on page 1-2 illustrates this concept.

Figure 1-1. A NetBatch Job



Executor Programs

A NetBatch job's executor program can be any user-written or NonStop program. This flexibility in choosing an executor program has two main benefits:

- Programmers can select the most appropriate executor programs for jobs and not have to learn a special job control language.
- Existing jobs need little or no conversion before you move them under NetBatch control.

Executor programs commonly used for NetBatch jobs are:

- The TACL program offers the full capabilities of the NonStop system command language.
- NBEXEC is an executor program tailored for batch processing. Supplied with NetBatch software, NBEXEC is an easy-to-learn alternative to the TACL program for job programming. NBEXEC also simplifies MIS Batch to NetBatch job migration.

Network Capabilities

Jobs can run on any node in a network. Also, you can control NetBatch schedulers on different nodes from a central site, reducing operation demands at remote locations.

CPU Assignment

The NetBatch scheduler routes jobs to user-specified CPUs and provides mechanisms (classes and executors) for controlling the routing process. These capabilities let you distribute the job workload according to CPU availability and OLTP demands.

ASSIGNs, DEFINEs, and PARAMs

The NetBatch scheduler propagates ASSIGNs, DEFINEs, and PARAMs to executor-program processes through user-defined entities called attachment sets. A single set can supply ASSIGNs, DEFINEs, and PARAMs to many jobs, thus simplifying job setup.

Operator Interface

The NetBatch operator interface (BATCHCOM) lets you inquire about and manipulate the scheduler, the scheduler's executors and classes, and attachment sets and jobs. The interface comes with a comprehensive online help facility that contains information about topics such as command syntax.

Scheduling Options

The NetBatch scheduler offers a variety of scheduling options that let you specify when jobs are to start. The scheduler can run a job at or after a specified date and time, or delay execution for a specified period after submission. Jobs can run automatically at regular, specified intervals or at times specified by a run calendar. Execution of a job also can depend on the completion of up to 50 other jobs.

Scheduler Logging

The NetBatch scheduler records in a log file details of events that occur while the scheduler is running. You can see the log file, for example, when troubleshooting the scheduler. In addition to maintaining this log file, the scheduler creates log files for the jobs it starts. The scheduler uses these log files to record details of events that occur during job execution.

Programmatic Interfaces

The NetBatch subsystem has programmatic interfaces (commands, responses, and event messages in Subsystem Programmatic Interface [SPI] format) that let your applications:

- Send commands to and receive responses from the subsystem in the form of tokenized messages
- Retrieve Event Management Service (EMS) events sent by the subsystem in the form of tokenized messages

For details of the interfaces, see the *NetBatch Management Programming Manual*.

Spooler Jobs

The spooler subsystems support the NetBatch product through their ability to link output of NetBatch jobs into batches of spooler jobs. This lets you manipulate the batched jobs in the spoolers as single entities instead of as individual jobs.

Core Components

The NetBatch product has five core components:

- A scheduler program (NETBATCH)
- A command interpreter (BATCHCOM)
- A calendar generation program (BATCHCAL)
- An executor program (NBEXEC)
- A TAL procedure (NB^JOB^SUBMIT)

Figure 1-2 on page 1-5 shows the components in a sample NetBatch environment.

NETBATCH

NETBATCH is the program-file ID of the NetBatch scheduler. The scheduler is a process-pair server that stores job records in its database. It schedules and starts the jobs, tracks and controls their execution, and records details of their termination. It also controls, by means of classes and executors, the distribution of jobs among CPUs in your system. The scheduler's command interface is BATCHCOM. For more information on the scheduler, see Scheduler Functions on page 1-6.

BATCHCOM

BATCHCOM is the program-file ID of the NetBatch command interpreter. BATCHCOM enables interactive and noninteractive manipulation of the scheduler, the scheduler's executors and classes, and attachment sets and jobs. For more information on BATCHCOM, see Section 6, Commands.

BATCHCAL

BATCHCAL is the program-file ID of the NetBatch calendar program. The program allows you to generate a calendar file containing a series of dates and times called run times. You can schedule a job to run automatically at those times by using the CALENDAR attribute to assign the file to the job. BATCHCAL also can display run times in a calendar file and reformat an old calendar file to the current format. For more information on BATCHCAL, see Section 5, Run Calendar Generation and Display.

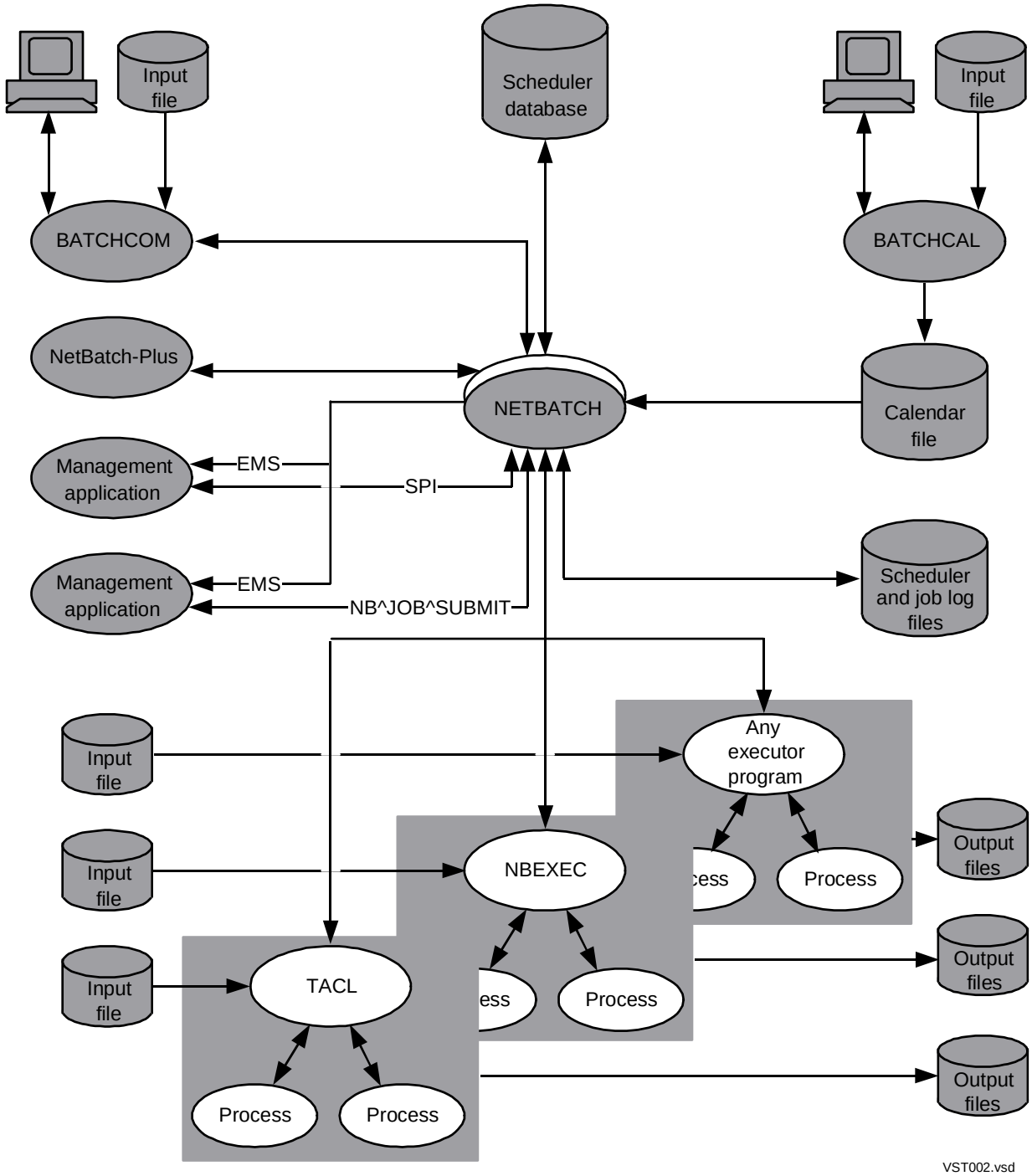
NBEXEC

NBEXEC is the program-file ID of the NetBatch executor program. Compatible with BPROC (the batch execution process of obsolete product MIS Batch), NBEXEC executes control file commands, supplies data to started processes, and logs process output. NBEXEC can run as a NonStop process pair and offers a simple-but-powerful job control language that includes error testing and job-recovery facilities. For more information on NBEXEC, see Appendix B, NBEXEC.

NB^JOB^SUBMIT

NB^JOB^SUBMIT is a TAL procedure defined in the NetBatch library file BATCHLIB. The procedure enables programmatic submission and alteration of jobs from user-written programs. For more information about NB^JOB^SUBMIT, see the *NetBatch Management Programming Manual*.

Figure 1-2. NetBatch Environment



Scheduler Functions

The scheduler is a process-pair server whose functions are to:

- Store job records in its database
- Schedule and start jobs
- Track and control job execution
- Record job termination details

This subsection expands this definition after describing how the scheduler assigns and controls the flow of jobs to CPUs.

CPU Assignment

The scheduler assigns jobs to CPUs by means of its classes and executors. By manipulating these classes and executors, you can control the flow of jobs to the CPUs.

Classes

A class is a logical entity in the scheduler. Its purpose is to group jobs and to control their flow to executors and thereby to the executors' CPUs.

- The CLASS job attribute assigns a job to a class.
- The INITIATION class attribute controls job flow from the class to executors. When set to ON, the attribute enables the scheduler to select jobs from the class. When set to OFF, the attribute prevents the scheduler from scanning the class and so prevents the class's jobs from running.

Classes are the job queuing mechanisms of the scheduler. You can assign a class to multiple executors to give its jobs opportunities to execute in different CPUs.

To add classes to the scheduler and display, and to alter their attributes, use the BATCHCOM commands ADD CLASS, INFO CLASS, and ALTER CLASS. To delete classes, use the BATCHCOM command DELETE CLASS.

Executors

Like a class, an executor is a logical entity in the scheduler. Its purpose is to link jobs by way of their classes to a CPU. This link enables the scheduler to start, in the specified CPU, the initial process (the executor program) of each job.

- The CLASS executor attribute specifies the executor's classes.
- The CPU executor attribute specifies the executor's CPU.

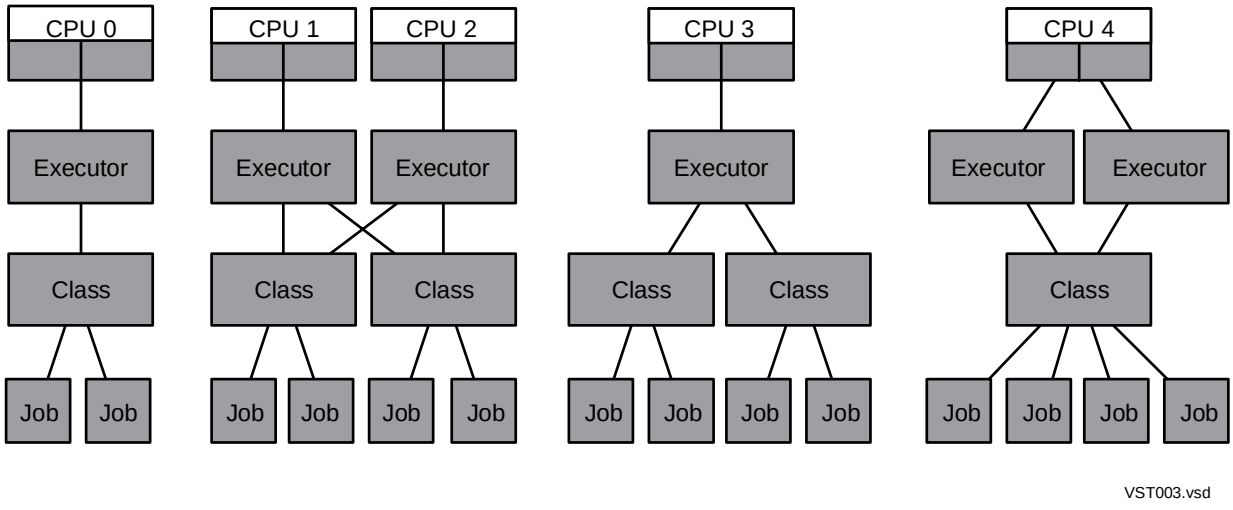
Executors act as gateways between classes and CPUs. When started, an executor lets one job at a time from the classes run in its CPU. No other jobs can use the executor

until the job finishes. (Stopping an executor prevents jobs from using it to access its CPU.) The number of started executors determines how many jobs can run together.

For example, a scheduler with 10 started executors can run up to 10 jobs concurrently. To add executors to a scheduler and display and alter their attributes, use the BATCHCOM commands ADD EXECUTOR, INFO EXECUTOR, and ALTER EXECUTOR. To start, stop, and delete executors, use the BATCHCOM commands START EXECUTOR, STOP EXECUTOR, and DELETE EXECUTOR.

Figure 1-3 illustrates the relationships between jobs, classes, executors, and CPUs by showing the class and executor configuration on a sample node.

Figure 1-3. Sample Job, Class, Executor, and CPU Relationships



Job Storage

The scheduler records in its database jobs submitted to it by the BATCHCOM command SUBMIT JOB. (Creation of the scheduler's database occurs by way of the BATCHCOM command ADD SCHEDULER during a scheduler cold start.) Each job record contains the job's attributes, which are:

Attribute	Function
CLASS, SELPRI	Specify a job's class and selection priority
EXECUTOR-PROGRAM, EXTSWAP,HIGHPIN, JOBID-ZERO, LIB, MEM,NAME, PFS, PRI, RUND, SAVEABEND,SWAP, TERM, STARTUP	Specify a job's executor program and the program's run options and parameters
IN, PURGE-IN-FILE, VOLUME	Specify a job's input file and volume and security defaults, and determine whether the scheduler purges the input file on job deletion
OUT, JOB-LOG, MAXPRINTLINES, MAXPRINTPAGES	Specify a job's output file, log file, and maximum print lines or maximum print pages

Attribute	Function (continued)
AFTER, AT, CALENDAR, EVERY, WAIT, LIMIT	Schedule a job and specify a time limit
HOLD, HOLDAFTER	Specify a job's hold characteristics
IFFAILS, RESTART, STALL, STOP-ON-ABEND	Control a job's behavior on process failure
TAPEDRIVES	Specifies a job's tape drives requirements
WAITON	Specifies a job's dependencies
ATTACHMENT-SET	Specifies a job's ASSIGNs, DEFINEs, and PARAMs
DESCRIPTION	Describes a job

The scheduler lets you display and alter a job's attributes by using the BATCHCOM commands INFO JOB and ALTER JOB. To delete jobs, use the BATCHCOM command DELETE JOB.

Job Scheduling and Startup

The scheduler uses a job's timing attributes to determine when the job should run. This table describes the functions of the timing attributes:

Attribute	Function
AFTER	Specifies a time after which a job becomes eligible for execution
AT	Specifies a time after which the scheduler runs the job
CALENDAR	Assigns a BATCHCOL-generated run calendar to the job
EVERY	Specifies a regular interval at which the scheduler automatically runs a job
WAIT	Prevents the scheduler from executing a job for a specified period after submission

AFTER, AT, and WAIT are mutually exclusive, as are CALENDAR and EVERY.

On satisfaction of the timing attributes, the scheduler starts the job (that is, runs its executor program) if all these conditions exist:

- The job's class has the attribute INITIATION ON.
- No other jobs in the job's class have a higher selection priority. Jobs with a higher selection priority are:
  - Jobs with dependencies, timing attributes, and SELPRI attributes overridden by the BATCHCOM commands RUNNOW JOB or RUNNEXT JOB
  - Jobs with a higher SELPRI attribute
  - Jobs with the same SELPRI attribute but an earlier submission time
- An executor is available to the job's class. (The scheduler creates a temporary executor for the job if the job has the AT attribute. The scheduler deletes the temporary executor after the job finishes.)

- Tape drives are available to the job (if the job has the TAPEDRIVES attribute).
- The job's master jobs have released it (if the job has the WAITON attribute).
- The job has the attribute HOLD OFF.

Figure 1-4 on page 1-10 illustrates and describes the algorithm used by the scheduler to select and start jobs.

Attachment Sets

An attachment set is a group of ASSIGNs, DEFINEs, and PARAMs you can link to jobs by the ATTACHMENT-SET attribute. When the scheduler starts the jobs, it propagates the set's ASSIGNs, DEFINEs and PARAMs to the jobs' executor-program processes.

To add attachment sets and display and alter their attributes, use the BATCHCOM commands ADD ATTACHMENT-SET, INFO ATTACHMENT-SET, and ALTER ATTACHMENT-SET. To delete attachment sets, use the BATCHCOM command DELETE ATTACHMENT-SET.

Job Tracking, Control, and Termination

When a scheduler starts a job with the attribute JOBID-ZERO OFF, it tags the job's executor-program process with a unique identifier called a GMOMJOBID. This identifier contains the scheduler's process name (that is, the name of the job's ancestor process, or GMOM) and the job's number (JOBID). The executor-program process then propagates the GMOMJOBID to its child processes. Thus, the GMOMJOBID identifies processes related to the job. Moreover, it lets the scheduler receive creation and completion information about the processes.

- Creation information lets the scheduler suspend, reactivate, and stop the processes.
- Completion information lets the scheduler accrue the processes' CPU time for inclusion in scheduler-generated EMS messages.

To view the processing states of jobs, use the BATCHCOM command STATUS JOB. To suspend and reactivate processes of running jobs, use the BATCHCOM commands SUSPEND JOB and ACTIVATE JOB. To terminate processes of running or suspended jobs, use the BATCHCOM command STOP JOB.

Figure 1-4. Scheduling Algorithm

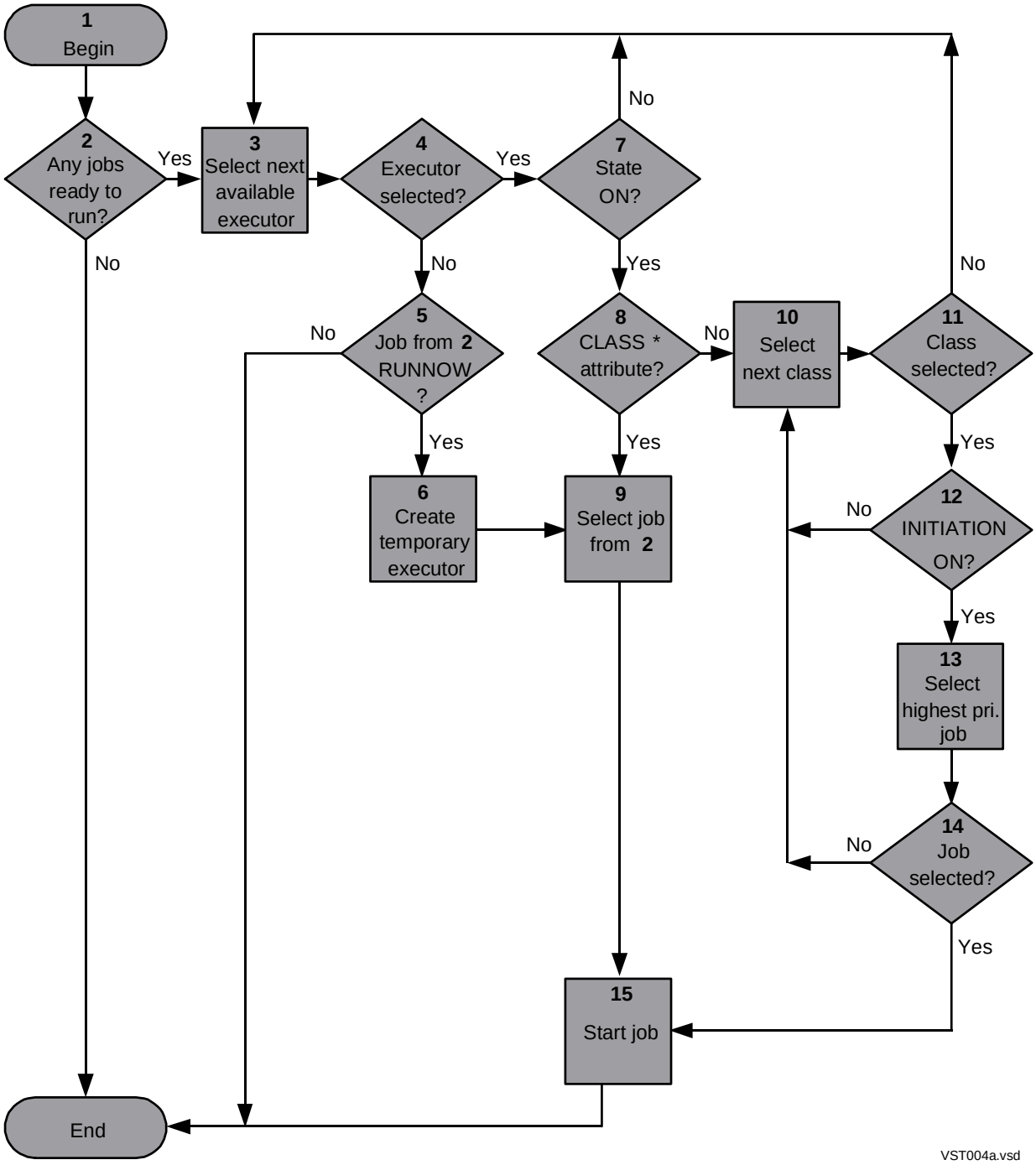


Table 1-1. Scheduling Algorithm Description

Item	Description
1. Begin.	These trigger events make the scheduler start the job selection and startup procedure: ● Submission of a job ● Satisfaction of a job's timing attributes (AFTER, AT, CALENDAR, EVERY, and WAIT) ● Alteration of a job's HOLD attribute from ON to OFF ● Completion of job startup or termination of a job
2. Any jobs ready to run?	The scheduler maintains an internal, prioritized list of jobs that are ready to run. It identifies the first (highest priority) job on the list and goes to Item 3. If there are jobs on the list, the scheduler exits the procedure. Ready to run jobs are in the RUNNOW, RUNNEXT, or READY states. Of these, RUNNOW jobs have the highest priority, followed by RUNNEXT then READY. The scheduler determines the priority of jobs with the same state from their SELPRI attributes. For jobs with the same state and SELPRI attribute, the scheduler determines priority by submission time on a first-in, first-out basis.
3. Select next available executor. 4. Executor selected?	The scheduler attempts selection from database file EXECUTOR of the next executor. If executor selection is successful, the scheduler goes to item 7. If unsuccessful (no more executors to select), the scheduler goes to item 5.
5. Job from item 2 RUNNOW? 6. Create temporary executor.	If the scheduler did not select an executor at item 3, it checks if the job identified at item 2 has a state of RUNNOW. If so, the scheduler creates a temporary executor for the job and goes to item 9. Otherwise, the scheduler exits the procedure.
7. State ON?	If the scheduler selects an executor at item 3, it checks if the executor's state is ON. If so, it goes to item 8. Otherwise, the scheduler returns to item 3.
8. Class * attribute?	The scheduler checks if the selected executor's CLASS attribute value is * (asterisk), meaning any class. If it is, the scheduler goes to item 9. Otherwise, the scheduler goes to item 10.
9. Select job from item 2.	The scheduler selects the job identified at item 2 and goes to item 15.
10. Select next class. 11. Class selected?	The scheduler attempts selection from database file JOBCLASS of the next class associated with the executor from item 3. If class selection is successful, the scheduler goes to item 12. If unsuccessful (no more classes to select), the scheduler returns to item 3.

Table 1-1. Scheduling Algorithm Description

Item	Description
12. INITIATION ON?	If the scheduler selects a class at item 10, it checks if the class's INITIATION attribute is ON. If so, it goes to item 13. Otherwise, the scheduler returns to item 10.
13. Select highest priority job. 14. Job selected?	The scheduler attempts selection of the highest priority job from the class selected at item 10 (not necessarily the job from item 2). If job selection is successful, the scheduler goes to item 15. If unsuccessful (no job selected), the scheduler returns to item 10. For information on how the scheduler prioritizes jobs, see item 2.
15. Start job.	The scheduler starts the job selected at item 9 or 14, then exits the procedure.

NetBatch High-PIN Capabilities

NetBatch product versions D20 and later use the high PIN feature of D-series systems. This feature allows more than 256 concurrent processes per CPU, thus removing a limitation of the C-series system. For more information on the feature, see the *Introduction to D-Series Systems*. Table 1-2 summarizes the high PIN capabilities of NetBatch product version D30. NetBatch product versions earlier than D20 do not have high PIN capabilities.

Table 1-2. NetBatch D20 High PIN Capabilities

	BATCHCAL	BATCHCOM	NBEXEC	NETBATCH
Can have a PIN creator?	Yes	Yes	Yes	Yes
Can communicate with high PIN requesters?	Not applicable	Yes	No	Yes
Can create high PIN processors?	Yes	Not applicable	No	Yes
Can run at a high PIN?	Yes	Yes	No	Yes
Can recognize high PIN process IDs?	Not applicable	Yes	No	Yes
Defaults to run at a high PIN?	Yes	Yes	Not applicable	Yes

For more information about the high PIN capabilities of NetBatch processes, see:

Process	Reference
BATCHCAL	Table 5-1, High PIN Capabilities in BATCHCAL, on page 5-4
BATCHCO	Table 6-1, High PIN Capabilities in BATCHCOM, on page 6-6
NBEXEC	Introducing NBEXEC on page B-1
NETBATCH	Table 3-2, High PIN Capabilities in NETBATCH, on page 3-14

2 Software Installation

This section describes how to install NetBatch:

Topic	Page
Installing NetBatch Software—Overview	2-1
Installation Procedure	2-3
Migrating a Scheduler Database	2-7

Note. You need a good working knowledge of NonStop software installation and system generation functions. For information on these topics, see the *DSM/SCM User's Guide* and the *System Generation Manual.*

Installing NetBatch Software—Overview

To install NetBatch software, the super ID (255,255):

1. Receives the software products into the DSM/SCM software archive
2. Updates the NetBatch subvolumes using DSM/SCM to build and apply a new configuration

Table 2-1. NetBatch Subvolumes

	Name	Variable Descriptions
Distribution Subvolume (DSV)	$*vol*.Y9190*Xnn*	*vol* is the volume specified when receiving product files. *Xnn* is the product version. For example, $D3000.Y9190D30.
Target Subvolumes (TSV)	$*vol*.SYSTEM $*vol*.ZNETBTCH $*vol*.ZSPIDEF $*vol*.ZSPISEGF $*vol*.ZTEMPL	*vol* is the volume specified when running DSM/SCM.

NetBatch Product Files

NetBatch product files can come on a site update tape (SUT) or software product revision (SPR) tape.

Table 2-2. NetBatch Product Files (page 1 of 2)

FileID	TSV	File Description
BATCHCAL	SYSTEM	Program file for NetBatch calendar program
BATCHCOM	SYSTEM	Program file for NetBatch command interpreter
BATCHIMU	SYSTEM	IMMU file containing NetBatch online help text

Table 2-2. NetBatch Product Files (page 2 of 2)

FileID	TSV	File Description
BATCHLIB	SYSTEM	Library file containing BATCHCAL, BATCHCOM,BATCHUTL, NBEXEC, and NETBATCH routines
BATCHUTL	SYSTEM	Program file for NetBatch national-language-support program
EXECUTE	ZNETBTCH	Source file containing NetBatch TACL macros and routines loaded by INSTALL's REPSUBSYS phase into directory: UTILS:ZBAT
NBEXDOC	ZNETBTCH	TFORM file containing information about NBEXEC
NBEXEC	SYSTEM	Program file for NetBatch executor program
NBSPIEX	ZNETBTCH	Source file containing sample Subsystem Programmatic Interface (SPI) and NB^JOB^SUBMIT programs for NetBatch subsystem
NETBATCH	SYSTEM	Program file for scheduler program
SBATTMPL	ZTEMPL	Source file for DSM format template used in formatting scheduler event messages
T9190*Xnn*	n.a. *	TGAL file containing NetBatch software release document
ZBATC	ZSPIDEF	Definition file, C
ZBATCOB	ZSPIDEF	Definition file, COBOL
ZBATDDL	ZSPIDEF	Definition file, DDL
ZBATSEGF	ZSPISEGF	Segment file containing NetBatch variables loaded by INSTALL's REPSUBSYS phase into TACL default segment file TACLSEGF
ZBATTACL	ZSPIDEF	Definition file, TACL
ZBATTAL	ZSPIDEF	Definition file, TAL
ZBATTMPL	ZTEMPL	Object file for DSM format template used in formatting scheduler event messages

* The NetBatch softdoc is stored in the DSM/SCM software archive, not in a NetBatch TSV.

Installation Prerequisites

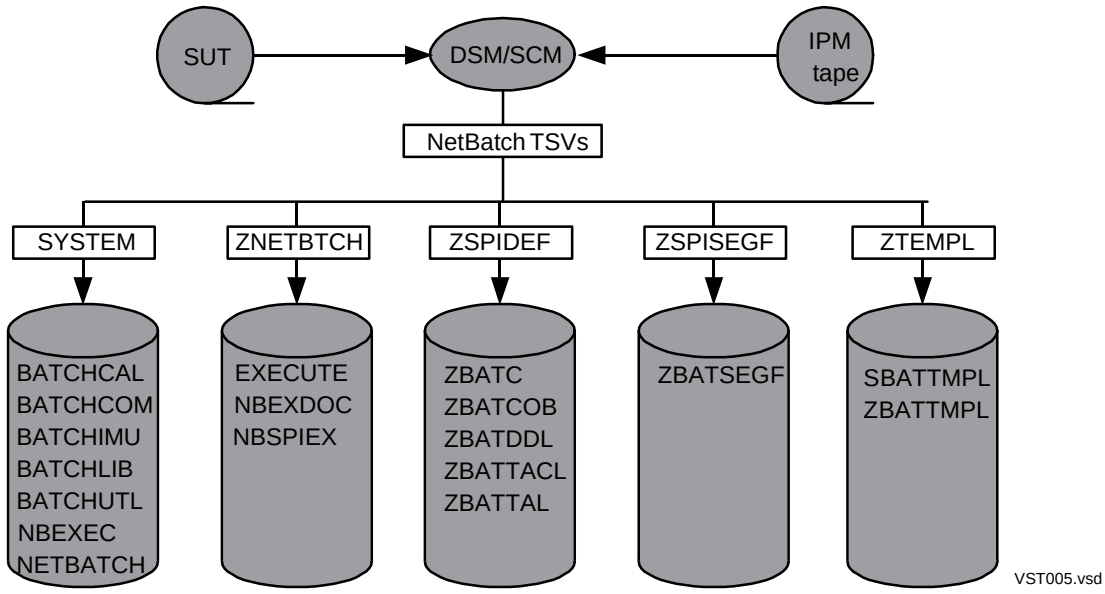
Table 2-3. NetBatch Installation Prerequisites (page 1 of 2)

	Prerequisite		
Hardware	NonStop system		
Firmware	Standard		
	Marketing product ID	Product Description	Version
Software	9072	NonStop OS package	D20 or later

Table 2-3. NetBatch Installation Prerequisites (page 2 of 2)

Prerequisite		
9150	DDL data definition language	D20 or later
9050	NonStop OS	G02.00 or later
0295	ENFORM PLUS	G02.00 or later

Figure 2-1. NetBatch Software Installation Overview



Note. ENFORM PLUS (T0295D46) is required only for the job-run history report generation feature.

Installation Procedure

The NetBatch software installation procedure varies, depending on whether NetBatch product files are on a SUT or SPR tape. For the latter, the procedure also varies depending on whether the files make up a full product release or a partial release only. For these reasons, detailed installation instructions appear in the softdoc supplied with the files and not in this section. However, this section gives an overview of the installation procedure with cross-references to softdoc instructions where necessary.

Step 1: Log On as the Super ID

Log on as the super ID (255,255). This lets you perform all installation tasks and avoid security violations that disrupt installation.

Step 2: Shut Down NETBATCH and Stop Its Processes

1. Shut down running NETBATCH schedulers, using the BATCHCOM command ABORT SCHEDULER or SHUTDOWN SCHEDULER.

   Note. Do not use the STOP command to stop a NETBATCH scheduler. STOP acts on the scheduler only and has no effect on processes of jobs the scheduler is controlling.

2. Stop running BATCHCAL, BATCHCOM, BATCHUTL, and NBEXEC processes, using the TACL STOP command.

To list processes using a given program file, use the TACL command STATUS *, PROG program-file-name. For example:

```
55> STATUS *, PROG $SYSTEM.SYSTEM.NETBATCH
Process … Program file Hometerm
$ZBAT B 0,25 … $SYSTEM.SYSTEM.NETBATCH $T2.#A
$SCHD B 0,74 … $SYSTEM.SYSTEM.NETBATCH $T7.#A
$SCHD 2,38 … $SYSTEM.SYSTEM.NETBATCH $T7.#A
$ZBAT 2,41 … $SYSTEM.SYSTEM.NETBATCH $T2.#A
```

Step 3: Receive NetBatch Product Files

Receive NetBatch product files from the SUT or SPR tape to the DSM/SCM software archive.

Step 4: Export and Print the NetBatch Softdoc

The NetBatch softdoc resides in the DSM/SCM software archive:

1. In DSM/SCM, select SUT or SPR in the software archive and list products.
2. Select the NetBatch product and list files.
3. Select the softdoc file and export it.

Step 5: Perform Pre-installation Tasks

Perform any pre-installation tasks listed in the softdoc's "Pre-Installation Instructions" that are not already complete. For example, a common pre-installation task involves backing up scheduler database files.

Step 6: Perform Installation-Related Tasks

Perform any installation-related tasks listed in the softdoc's "Installation Instructions" that are not already complete. Typically, these tasks include running one or more INSTALL phases such as REPSUBSYS.

Step 7: Perform Post-installation Tasks

Perform any post-installation tasks listed in the softdoc's "Post-Installation Instructions". Tasks can include creating and binding the NBFLAGS procedure, specifying the library file for NetBatch programs, securing files, and licensing the NETBATCH program file.

Creating and Binding the NBFLAGS Procedure

NBFLAGS is a TAL procedure that you create and bind into the BATCHLIB library file that lets you:

- Disable WAITON validation by the scheduler. For this validation, which otherwise occurs, the scheduler checks that dependent jobs do not wait on themselves. Disabling WAITON validation reduces the time it takes to submit jobs whose dependencies you already tested.
- Allow jobs owned by frozen users to run. These jobs would otherwise not run.

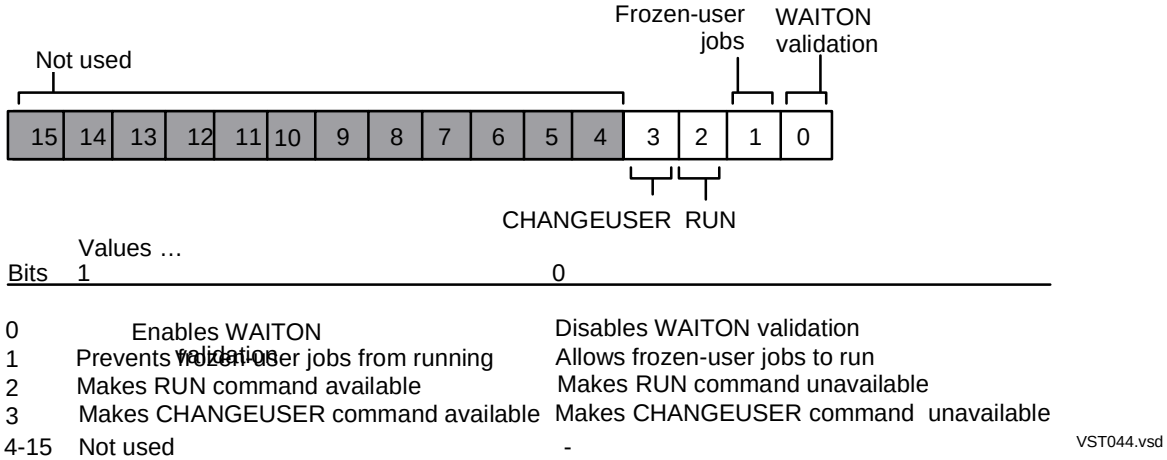
  For information on frozen users, see the FREEZE USER command in the *Safeguard Reference Manual*.
- Disable the BATCHCOM commands CHANGEUSER and RUN. Otherwise these commands are available to all users.

NBFLAGS returns a word of 16 bits:

Bits	Set to...	Description
0-3	0 or 1	Control WAITON validation, frozen-user jobs, and the commands CHANGEUSER and RUN
4-15	1	Not used

Omitting NBFLAGS from BATCHLIB is the same as including it with all bits set to 1.

Figure 2-2. NBFLAGS Bit Settings



To create NBFLAGS and bind it into BATCHLIB:

1. Make a backup copy of BATCHLIB. For example:

```
> FUP DUP $DATA7.TEST.BATCHLIB,
$NB.BACKUPS.BATCHLIB FILES DUPLICATED: 1
>
```

2. Create the NBFLAGS procedure. For example:

```
> TAL; SUPPRESS
> INT PROC NBFLAGS; BEGIN RETURN %B1111111111110111; END;
> EOF!
BINDER - OBJECT FILE BINDER ...
Object File
\MELBDEV.$DATA7.TEST.OBJECT.
>
```

   These bit settings enable WAITON validation, prevent frozen-user jobs from running, enable the RUN command, and disable the CHANGEUSER command.

3. Bind NBFLAGS into BATCHLIB. For example:

```
> BIND
BINDER - OBJECT FILE BINDER ...
@ADD * FROM $DATA7.TEST.BATCHLIB
@REPLACE * FROM $DATA7.TEST.OBJECT
@BUILD $DATA7.TEST.BATCHLIB !
@EXIT
```

Specifying the Library File

The programs BATCHCAL, BATCHCOM, BATCHUTL, NBEXEC, and NETBATCH search for their BATCHLIB library file in $SYSTEM.SYSTEM. If you move the file, you must specify the library file for the programs the first time you run them.

To specify the library file for a program you are running, include run option `LIB` *`file-name`* in the TACL RUN command. For example:

```
> BATCHCAL /LIB $VOL.SUBVOL.BATCHLIB/C}EXIT
> BATCHCOM /LIB $VOL.SUBVOL.BATCHLIB/; EXIT
> BATCHUTL /LIB $VOL.SUBVOL.BATCHLIB/
> NBEXEC /NAME, LIB $VOL.SUBVOL.BATCHLIB/
> NETBATCH /NAME $ZBAT, NOWAIT, LIB $VOL.SUBVOL.BATCHLIB/ $TRASH.ZBAT
> BATCHCOM $ZBAT; SHUTDOWN SCHEDULER
```

Securing Files

The softdoc might list NetBatch product files whose security you must change to recommended values before you use the files. The most common file security changes are to the files BATCHLIB and NETBATCH.

Licensing the NETBATCH Program File

The NETBATCH program file is the only file you must license before use (the program contains privileged code). To license the file, use the FUP LICENSE command. For example:

```
> FUP LICENSE $SYSTEM.SYSTEM.NETBATCH
```

Step 8: Restart Schedulers

Restart the schedulers shut down in Step 2.

This involves cold starting or warm starting the schedulers you shut down at the beginning of the installation procedure. In most cases, you would warm start the schedulers. For information on cold-start and warm-start procedures, see Section 3, Scheduler Planning, Configuration, and Management.

Migrating a Scheduler Database

Migrating a scheduler's database from one location to another is sometimes necessary (for example, when moving scheduler operations from a development node to a production node).

Step 1: Log on as the Super ID (255,255).

```
> LOGON SUPER.SUPER, password
```

Step 2: Shut Down the Scheduler

Shut down the scheduler that is using the database:

```
> BATCHCOM $SCHD; SHUTDOWN SCHEDULER
Scheduler shutting down
```

Step 3: Copy the Database Files to the Target Location

Copy the database files to the target location by using the FUP DUP command with the SAVEALL option:

```
> VOLUME $DATA7.SCHD
> FILES

\MELBDEV.$DATA7.SCHD

ATTACH ATTACH0 BATCHCTL CHKQUE CHKQUE0 EXECUTO0

EXECUTOR JOB JOBCLAS0 JOBCLASS LOGAAA NBATTX

NBATTX0

> FUP DUP * , \MELRISK.$DATA.SCHD.* , SAVEALL

WARNING –\MELRISK.$DATA.SCHD.JOB: ASSUMEID NOT PRESERVED AT

DESTINATION

FILES DUPLICATED: 13
```

SAVEALL preserves the owner ID, RWEP security attributes, and last-modified timestamp of all files except JOB.

Step 4: Secure the Migrated JOB File to Local Super ID

Secure the migrated JOB file “OOOO” to the local super ID by using the FUP SECURE command with the PROGID option:

```
> \MELRISK.TACL

TACL 1> LOGON SUPER.SUPER, password

> VOLUME $DATA.SCHD
> FILES


\MELRISK.$DATA.SCHD

ATTACH ATTACH0 BATCHCTL CHKQUE CHKQUE0 EXECUTO0

EXECUTOR JOB JOBCLAS0 JOBCLASS LOGAAA NBATTX

NBATTX0

> FUP

–SECURE JOB, "OOOO", PROGID
```

Step 5: Update Alternate Key File References

Update the alternate key file references of these migrated files by using the FUP ALTER command: ATTACH, CHKQUE, EXECUTOR, JOBCLASS, and NBATTX.

```
-ALTER ATTACH, ALTFILE (0, ATTACH0)
-ALTER CHKQUE, ALTFILE (0, CHKQUE0)
-ALTER EXECUTOR, ALTFILE (0, EXECUTO0)
-ALTER JOBCLASS, ALTFILE (0, JOBCLAS0)
-ALTER NBATTX, ALTFILE (0, NBATTX0)
-EXIT
```

Step 6: Warm Start Scheduler

Warm start the scheduler shut down in Step 2. For more information, see Warm Starting a Scheduler on page 3-18.

```
> NETBATCH /NAME $SCHD, NOWAIT/ $DATA.SCHD
> BATCHCOM $SCHD; START SCHEDULER
Scheduler started
```

Step 7: Verify and Alter Attributes

Where necessary, verify and alter the attributes of the scheduler and its classes, executors, attachment sets, and jobs. This ensures the attributes specify values appropriate to the scheduler's new location.(For example, the volume and subvolume specified for job input files might need changing.

Use BATCHCOM's INFO and ALTER commands to list and alter the attributes:

```
> BATCHCOM $SCHD; INFO SCHEDULER /OUT A/
> BATCHCOM $SCHD; INFO CLASS /OUT B/ *
> BATCHCOM $SCHD; INFO EXECUTOR /OUT C/ *
> BATCHCOM $SCHD; INFO ATTACHMENT-SET /OUT D/ ( * . * ) * , DETAIL
> BATCHCOM $SCHD; INFO JOB /OUT E/ *
```

3

Scheduler Planning, Configuration, and Management

This section describes how to work with the scheduler:

Topic	Page
Planning Schedulers	3-1
Starting Schedulers	3-9
Configuring Schedulers	3-23
Managing Schedulers	3-29

Planning Schedulers

Planning your organization's schedulers is a three-stage process that involves:

Task 1. Compile Job Information	3-1
Task 2. Determine Scheduler Numbers and Attributes	3-2
Task 3. Establish Class and Executor Configurations	3-6

Task 1. Compile Job Information

To start planning schedulers, gather information about jobs whose submission, execution, and management you want placed under NetBatch control. This information helps you determine how many schedulers you need, the schedulers' attributes, and their class and executor configurations.

This table lists the information you need for each job you identify. Record information about your jobs in a table similar to Figure 3-1 on page 3-2. For a detailed job planning procedure to help you get this information, see Planning Procedure on page 4-1.

Information required	Function
Job category	Describes the job or job type (for example, COBOL compiles)
Executor program	Specifies the job's executor program (for example, TACL or NBEXEC)
Executor-program priority	Specifies the execution priority of the job's executor-program process
Executor-program nodes	Specifies the nodes on which the job runs
CPU-bound or I/O-bound	Specifies whether job processing is CPU-bound or I/O-bound. (CPU-bound processing occurs predominantly in a CPU, with few references to I/O resources such as disk files. I/O-bound processing places a heavy demand on I/O resources, but few demands on CPU resources.)

Information required	Function
Tape drives	Specifies the number of tape drives required by the job
Output file	Specifies the name of the job's output file
Duration	Specifies how long the job takes to run
Frequency	Specifies how often the job runs

Figure 3-1 records information about jobs in a sample three-node system. It is the basis of the examples in this subsection.

Figure 3-1. Example of a Job Table for Scheduler Planning

Job category	Executor program	PRI	Nodes	CPU- or I/O-bound?	Tape drives	Output file	Duration	Frequency	Concurrent?
Backups ? daily	NBEXEC	149	\DEV	I/O	1	$S.#DAYBKUP	00:15	Daily (M–F)	No
Backups ? weekly	TACL	149	\DEV \PROD	I/O	2	$S.#WKBKUP	00:25	Weekly (Sat)	No
Backups ? monthly	TACL	149	\DEV \QA \PROD	I/O	3	$S.#MTHBKUP	00:40	Monthly (Sun)	No
COBOL compiles	COBOL85	119	\DEV	CPU	0	$S.#COBCOMP	00:30	On demand	Yes
Transaction log	TACL	150	\PROD	I/O	0	DAYLOG	00:01	Every 15 mins	Yes
Reports	Enform	110	\PROD	I/O	0	$S.#ADMIN	00:10	On demand	Yes
Interest calculation	MYPROG	130	\PROD	CPU	0	$TRM2.#A	03:00	Monthly	No

VST100.vsd

Task 2. Determine Scheduler Numbers and Attributes

Decide how many schedulers you need and determine their attributes. Your job table from Task 1 can help you make these decisions. Record your schedulers' details in a table similar to Figure 3-2 on page 3-5.

Determining How Many Schedulers You Need

The number of schedulers you need depends on:

- The number of nodes in your system
- The number of jobs that will run concurrently on each node
- Your organization's NetBatch user-training and job-testing needs

General Rule

As a rule, plan no more than one scheduler for each node where you want jobs placed under NetBatch control. A single scheduler on a node should be enough to satisfy users' requirements on that node.

The benefits of limiting schedulers to one per node are simplified job and system management are:

- You can manage CPU resources and job flow more efficiently and easily with one scheduler than with several.
- Having a single scheduler means there is only one scheduler database to maintain and one scheduler log file to examine.

Of course, NetBatch does not prevent you from having more than one scheduler per node. As you increase the number of schedulers, however, job and system management become more complex.

Exceptions to the Rule

The one scheduler per node rule has some exceptions:

- For user-training and job-testing purposes, plan at least one extra scheduler on each node where those activities might occur. The extra schedulers enable users to familiarize themselves with NetBatch functions and to test jobs without disrupting your production schedulers.
- For nodes where more than 500 concurrent jobs are likely, plan as many schedulers as you need to service the job load. For example, to run 600 jobs concurrently, you need at least two schedulers. For more information on concurrent jobs, see Task 1. Compile Job Information on page 3-1 and Task 3. Establish Class and Executor Configurations on page 3-6.

Determining Your Schedulers' Attributes

Every scheduler has a permanent set of attributes that:

- Specify the preferred CPUs for the scheduler's backup process
- Specify default values for some executor and job attributes
- Determine the number of tape drives available for use by jobs
- Enable or disable job submission, job initiation, and event-message generation
- Control use of the AT job attribute and the accumulation of run backlogs
- Determine whether the scheduler treats jobs submitted from licensed requesters on remote nodes as local jobs, not as remote jobs
- Specify the scheduler's concurrent-jobs, temporary-executors, and execution-priority limits

The attributes' values can be scheduler-assigned defaults, or values you specify during scheduler startup (by means of the ADD SCHEDULER command) or configuration (by means of the ALTER SCHEDULER command). For more information, see Section 7, Attributes.

Table 3-1. Scheduler Attributes (page 1 of 2)

Attribute	Function	Default Value
AT-ALLOWED	Determines whether users without execute access to the NETBATCH program file can submit jobs with the AT attribute	OFF
BACKUPCPU	Specifies the preferred CPUs for a scheduler's backup process	CPU of scheduler's primary process
CATCHUP	Determines whether jobs with the EVERY attribute accumulate run backlogs	ON
DEFAULT-CLASS	Specifies the class of a job submitted without the CLASS job attribute. Also specifies the class of an executor added without the CLASS executor attribute	DEFAULT
DEFAULT-EXECUTOR-PROGRAM	Specifies the executor program of a job submitted without the EXECUTOR-PROGRAM attribute	$SYSTEM.SYSTEM.TACL
DEFAULT-HIGHPIN	Specifies the HIGHPIN attribute of a job submitted without that attribute	OFF
DEFAULT-MAXPRINT-LINES	Specifies the maximum number of output file print lines for a job submitted without the MAXPRINTLINES attribute	NONE
DEFAULT-MAXPRINT-PAGES	Specifies the maximum number of output file print pages for a job submitted without the MAXPRINTPAGES attribute	NONE
DEFAULT-OUT	Specifies the output file of a job submitted without the OUT attribute	$S.#BATCH
DEFAULT-PRI	Specifies the execution priority of the executor-program process of a job submitted without the PRI attribute	120
DEFAULT-SELPRI	Specifies the selection priority of a job submitted without the SELPRI attribute	3
DEFAULT-STALL	Specifies the STALL attribute of a job submitted without that attribute	OFF
DEFAULT-STOP-ON-ABEND	Specifies the STOP-ON-ABEND attribute of a job submitted without that attribute	OFF
EMS	Enables or disables event-message generation while the scheduler is running	OFF

Table 3-1. Scheduler Attributes (page 2 of 2)

Attribute	Function	Default Value
INITIATION	Enables or disables job startup by the scheduler	ON
LOCALNAMES	Makes the scheduler treat jobs submitted from licensed requesters on the specified nodes as local jobs, not as remote jobs	RUN NETBATCH *remote-node* nodes
MAX-CONCURRENT-JOBS	Specifies the maximum number of concurrent jobs and temporary executors for the scheduler	500,500
MAX-PRI	Specifies an upper execution-priority limit for executor-program processes	199
SUBMIT-ALLOWED	Allows or disallows job submission by permitting or preventing use of the SUBMIT JOB command	ON
TAPEDRIVES	Specifies the number of tape drives available to jobs	2

Figure 3-2 shows details of schedulers planned for the three-node system from the job planning example earlier in this subsection. Most of the details derive from information in the job table (Figure 3-1 on page 3-2). For example, the TACL program is the default executor program for scheduler \PROD.$ZBAT because most jobs on \PROD are TACL jobs. Information not derived from the job table comes from the planner's system knowledge. For example, scheduler \DEV.$TEST will run in underutilized CPUs on \DEV.

Figure 3-2. Example of a Scheduler Planning Table

Scheduler						Scheduler attributes					
Node	Proc. name	Database location	CPUs		Event msgs req'd?	BACK-UP-CPU	DEFAULT EXECUTOR PROGRAM	DEFAULT-OUT	DEFAULT-PRI	TAPE-DRIVES	Other
			Prmry	Bckp							
\DEV	$DBAT	$DATA7.DBAT	0	1	No	0,1	COBOL85	$S.#COBCOMP	119	0	Defaults
\DEV	$TEST	$TRASH.TEST	2	3	No	2,3	TACL	$NULL	110	0	Defaults
\QA	$QBAT	$QAT1.QBAT	0	3	No	0,3	TACL	$S.#MTHBKUP	149	3	Defaults
\PROD	$ZBAT	$NB.ZBAT	1	2	Yes	1,2	TACL	$S.#ZBAT	149	3	Defaults

VST101.vsd

Task 3. Establish Class and Executor Configurations

Determine the class and executor configurations of each scheduler.

Establishing Class Configurations

Classes function primarily as CPU resource managers. Through their INITIATION attributes, you can control the flow of jobs to executors and therefore to those executors' CPUs. Thus, classes help you balance job workload across all CPUs in your system.

Classes are most effective as CPU resource managers when they cater to jobs with similar CPU requirements. Therefore, plan at least one class for use specifically by CPU-bound jobs, and plan other classes for I/O-bound jobs.

To further enhance the effectiveness of classes as CPU resource managers, have classes that cater for jobs with similar processing times. For example, consider planning separate classes for long and short jobs.

You also can use classes to group jobs that have a common function or purpose. For example, you could plan a class for payroll jobs, another for high-priority jobs, and one for jobs from a particular department. However, using classes in this way can result in a class whose jobs have considerably different CPU-resource requirements and processing times. Jobs from such a class make CPU load balancing difficult, so use classes to group jobs by function or purpose only after establishing what effect the jobs will have on CPU-management activities.

Class Numbers

The NetBatch product does not limit the number of classes you can set up for each scheduler. As the number of classes increases, however, so does the complexity of managing them.

You can determine a practical limit for class numbers by multiplying the number of executors in a scheduler by eight. (Eight is the maximum number of classes you can assign to an executor.) For example, a scheduler with four executors could have 32 classes (eight classes multiplied by four executors).

Class-Naming Conventions

Consider naming conventions when planning your schedulers' classes. Some commands let you specify multiple classes with wild-card characters, so use names you can mask easily with those characters.

Establishing Executor Configuration

The information you use to plan a scheduler's classes also helps you plan the executors for that scheduler. The information helps you plan the executors because executors perform a similar function to classes (controlling CPU workload).

Executor Numbers

The number of executors you plan depends on the number of CPUs on the scheduler's node and on your jobs' CPU and I/O resource requirements. HP recommends four executors per CPU:

- One executor for classes that group jobs by function or purpose.
- One executor for classes whose jobs are CPU-bound.
- Two executors for classes whose jobs are I/O-bound. You can have more executors for these classes, depending on your CPUs' power and the number of logical devices configured for them. For example, an eight-disk-per-CPU NonStop Cyclone system can handle more executors than a two-disk-per-CPU NonStop CLX system.

An exception to the four-executors-per-CPU rule applies when you want to run more concurrent jobs than there are executors. Because the number of started executors determines how many jobs can run concurrently (one job per executor), you might need extra executors. For example, to run 20 concurrent jobs in a scheduler with 16 executors (all started), you would need another four started executors.

CPU Assignment

Consider existing loads on CPUs before assigning executors to those CPUs. For example, a CPU might be running a critical online Pathway application. Increasing that CPU's workload by directing jobs to it through executors might affect the application's performance. In these circumstances, consider assigning only one executor to the CPU or perhaps none.

Class Assignment

As a general rule, assign classes whose jobs have similar characteristics to executors catering to jobs with those characteristics. For example, assign classes containing CPU-bound jobs to executors dedicated to processing CPU-bound jobs. Similarly, assign classes for I/O-bound jobs to executors processing I/O-bound jobs, and so on. Assigning a class to more than one executor helps avoid a job backlog in that class.

The order in which you assign classes to executors is also important. Jobs in the first class assigned to an executor take priority over jobs in the second class, jobs in the second class have priority over those in the third class, and so on. Varying the order in which you assign classes to executors ensures jobs in those classes have similar processing opportunities.

Executor-Naming Conventions

As with class names, consider naming conventions when planning your schedulers' executors. Some commands let you specify multiple executors with wild-card characters, so use names you can mask easily with those characters.

Example

Figure 3-3 shows an example of a table showing the classes and executors planned for a scheduler running on a four-CPU node. As well as showing the executors assigned to each CPU, the figure illustrates class distribution among those executors. For example, executors E1, E4, E8, and E12 have classes C1 through C4 assigned to them in different order. Jobs in those classes have similar processing opportunities because each class appears once at the top of a class list, once in second position, and so on.

Figure 3-3. Example of a Class and Executor Planning Table

CPUs	**0** (All jobs except CPU-bound jobs)			**1** (All jobs)				**2** (All jobs)				**3** (All jobs)			
Executors	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14	E15
Classes	C1	C5	C6	C2	C9	C7	C8	C3	C10	C12	C14	C4	C11	C13	C15
	C2	C6	C7	C3	C10	C8	C5	C4	C11	C13	C15	C1	C9	C12	C14
	C3	C7	C8	C4	C11	C5	C6	C1	C9			C2	C10		
	C4	C8	C5	C1		C6	C7	C2				C3			

Key to Executors

- E1?4, E6?8, E12 — For classes containing I/O-bound jobs
- E5, E9, E13 — For classes containing CPU-bound jobs
- E10?11, E14?15 — For classes containing jobs grouped by function or purpose

Key to Classes

- C1?4 — For I/O-bound jobs
- C5?8 — For CPU-bound jobs
- C9?11 — For department A's jobs
- C12 — For end-of-period jobs
- C13 — For department B's jobs
- C14 — For general ledger jobs
- C15

VST102.vsd

Starting Schedulers

Topic	Page
Overview	3-9
Running NETBATCH	3-10
Cold Starting a Scheduler	3-15
Warm Starting a Scheduler	3-18
Warm-Start Events	3-21

Overview

Scheduler startup is a task for NetBatch supervisors (users with execute access to the NETBATCH program file). It involves:

- Running the scheduler program NETBATCH by using the TACL RUN command
- Executing the BATCHCOM commands ADD SCHEDULER and START SCHEDULER (for a cold start) or the START SCHEDULER command only (for a warm start)

Cold Starts

To cold start a scheduler:

1. Run NETBATCH:

   ```
   35> NETBATCH /NAME $ABCD, NOWAIT, PRI 130, CPU 0/ $NB.ABCD !
   ```

2. Run the ADD SCHEDULER command:

   ```
   36> BATCHCOM $ABCD; ADD SCHEDULER, BACKUPCPU 0,1
   Scheduler added
   ```

3. Run the START SCHEDULER command:

   ```
   37> BATCHCOM $ABCD; START SCHEDULER
   Scheduler started
   ```

This command sequence purges the database specified in the RUN NETBATCH command and creates a new database. For complete cold-start instructions, see Cold Starting a Scheduler on page 3-15.

Warm Starts

To warm start a scheduler, run NETBATCH, then execute the START SCHEDULER command (ADD SCHEDULER command omitted). This command sequence opens the database specified in the RUN NETBATCH command:

```
24> NETBATCH /NAME $WXYZ, NOWAIT, PRI 149, CPU 2/ $BIG1.WXYZ
25> BATCHCOM $WXYZ; ALTER SCHEDULER, BACKUPCPU 2,3
```

```
Scheduler altered
26> BATCHCOM $WXYZ; START SCHEDULER
Scheduler started
```

For complete warm-start instructions, see Warm Starting a Scheduler on page 3-18.

Cold Starts Versus Warm Starts

This table describes when to cold start or warm start a scheduler:

If you want to …	Then perform a …
Start a scheduler whose database does not exist	Cold start
Start a scheduler that opens an existing scheduler database	Warm start
Start a scheduler that purges an existing scheduler database	Cold start

Running NETBATCH

To run the scheduler program NETBATCH, use the TACL RUN command:

```
[ RUN ] [ \node. ] [ volume. ] NETBATCH
/ NAME [ $schd ] , NOWAIT [ , run-option ]… /
[ schd-database-subvol ] [ ! ] [ \remote-node ]…
[ EMS ] [ DISPLAY-SPI ] [ REPORT-ON jobrun-database-subvol]
```

node

is the name of the node where the NETBATCH program file resides. If the node is remote, you must have remote access to it (that is, you must have remote passwords set up on both local and remote nodes). The default is the node where the process that creates the NETBATCH process is running.

volume

is one of:

$*volume-name.subvolume-name*

specifies the volume and subvolume containing the NETBATCH program file (usually $SYSTEM.SYSTEM).

subvolume-name

specifies the subvolume containing the NETBATCH program file.

The default volume varies, depending on whether the RUN command is explicit or implicit:

- For an explicit RUN command, enter RUN followed by the NETBATCH program file name. If you enter a partially qualified file name, the TACL program expands it by using the current default node, volume, and subvolume names.

- For an implicit RUN command, enter the NETBATCH program file name only. If you enter a partially qualified file name, the TACL program searches for the file in the subvolumes specified by the TACL #PMSEARCHLIST variable.

schd

specifies the name of the scheduler process. The name can contain from one to four alphanumeric characters (preceded by the dollar sign), of which the first must be alphabetic. Omitting *schd* makes the TACL program generate a process name of the default form (for example, $Z024).

NOWAIT

causes the TACL program to return its prompt after sending the startup message to the scheduler process, instead of pausing while the scheduler runs.

run-option

is one of these TACL RUN command options. For descriptions of these options, see the RUN[D] command description in the *TACL Reference Manual*.

CPU	HIGHPIN	INV	OUT	STATUS
DEBUG	IN	JOBID	OUTV	SWAP
DEFMODE	INLINE	LIB	PFS	TERM
EXTSWAP	INSPECT	MEM	PRI	WINDOW

schd-database-subvol

specifies, in one of these forms, the name of the subvolume where the scheduler creates, opens, and maintains its database files:

$*volume-name.subvolume-name*
$*volume-name*
subvolume-name

The default *schd-database-subvol* varies, depending on whether you omit the parameter or specify only one of *$volume-name* and *subvolume-name*.

- When you omit *schd-database-subvol*, the default is the current default subvolume. For example, the scheduler started by this command specifies (by omission) $DATA7.TRASH as its database subvolume:

```
$DATA7.TRASH 93> $NB.MAN14.NETBATCH /NAME, NOWAIT/
```

- When you specify one of *$volume-name* and *subvolume-name* for *schd-database-subvol*, NETBATCH uses the current defaults to supply the other. For example, the scheduler started by this command specifies $SYSTEM.TRASH as its database subvolume:

```
$A.TRASH 104> $NB.MAN14.NETBATCH /NAME, NOWAIT/ $SYSTEM
```

Note. BATCHCOM searches for a scheduler named $ZBAT when a RUN BATCHCOM command does not specify a scheduler. For this reason, consider using $ZBAT as a scheduler name.

!

indicates a cold start, thereby causing the scheduler to bypass validation of its database and to purge these files from the database:

ATTACH	CHKQUE0	JOB	JOBCLASS
ATTACH10	CHKQUESV	JOB0	JOBSAVED
BATCHCTL	EXECUTO0	JOB1	NBATTX
CHKQUE	EXECUTOR	JOBCLAS0	NBATTX0

! has no effect on files not listed previously, such as scheduler log files. Omitting ! indicates a warm start and causes the scheduler to validate its database.

remote-node

is the name of a node remote to the scheduler. The parameter, which sets the scheduler's LOCALNAMES attribute, makes the scheduler treat jobs submitted from licensed requesters on specified nodes as local jobs, not as remote jobs. (An example of such a requester is NetBatch-Plus.) Through the schedulre, the remotely submitted jobs gain the same access privileges on the scheduler's node as locally submitted jobs.

A scheduler without the LOCALNAMES attribute treats remotely submitted jobs as remote jobs subject to normal Nonstop system remote-access restrictions. *remote-node*, which can specify up to 30 remote nodes, can only be set when NETBATCH is run by the local super ID (255,255).

For more information, see LOCALNAMES Scheduler Attribute on page 7-77.

EMS

sets the EMS scheduler attribute to ON to enable event-message generation by the scheduler. For more information, see EMS Scheduler Attribute on page 7-54.

DISPLAY-SPI

makes the scheduler write to its log file SPI-format command and response buffers received from and sent to requesters such as BATCHCOM. The buffers appear in the log file in the same format as they appear in BATCHCOM when DISPLAY-SPI is set to ON. For details, see DISPLAY-SPI Command on page 6-86.

REPORT-ON

sets the REPORT scheduler attribute to ON to generate reports by the scheduler.

jobrun-database-subvol

specifies the location of the JOBRUN database, dictionary, and compiled query files.

Examples

- To run NETBATCH, creating scheduler $ZBAT's primary process in CPU 3, with the scheduler's database $DATA7.ZBAT:

```
NETBATCH /NAME $ZBAT, NOWAIT, CPU 3/ $DATA7.ZBAT
ATTACH CHKQUE0 JOB JOBCLASS
ATTACH10 CHKQUESV JOB0 JOBSAVED
BATCHCTL EXECUTO0 JOB1 NBATTX
CHKQUE EXECUTOR JOBCLAS0 NBATTX0
```

- To run NETBATCH with the primary process of scheduler $SCHD at a priority of 150, specifying a cold start and enabling EMS event-message generation:

```
NETBATCH /NAME $SCHD, NOWAIT, PRI 150/ $NB.SCHD ! EMS
```

- To run NETBATCH enabling scheduler $BANK to treat jobs submitted from three remote nodes as local jobs:

```
NETBATCH /NAME $BANK, NOWAIT/ $BIG2.BANK \DEV \QA \PROD
```

NETBATCH's High PIN Capabilities

Table 3-2 on page 3-14 lists the high PIN capabilities of NETBATCH version D20. NETBATCH versions earlier than version D20 do not have high PIN capabilities.

Table 3-2. High PIN Capabilities in NETBATCH

Feature	Implementation	Comments
Can have a high PIN creator?	Yes	None
Can communicate with high PIN requests?	Yes	None
Defaults to run at a high PIN?	Yes	A NETBATCH process runs at a high PIN by default when all these conditions exist: ● The operating system on the node where the process runs is a D-series system configured to handle high PIN processes. ● The NETBATCH process creator (for example, a D-series TACL process) can create high PIN processes. ● The NETBATCH process creator does not force the process to run at a low PIN. ● A high PIN is available in the target CPU. To prevent NETBATCH from running at a high PIN, specify run option HIGHPIN OFF in the TACL RUN command: `RUN NETBATCH /HIGHPIN OFF, ../...`
Can communicate with high PIN servers?	Yes	None
Can create high PIN processes?	Yes	None
Can run at a high PIN?	Yes	None
Can recognize high PIN process IDs?	Yes	None

Maximum Opens

The scheduler can have a maximum of 2000 openers, each of which can be any of:

- An executor-program process
- A child process of an executor-program process
- A BATCHCOM, NetBatch-Plus, or user-written requester process

Cold Starting a Scheduler

Step 1: Log On as a NetBatch Supervisor

You must be a NetBatch supervisor to run NETBATCH and to use the ADD SCHEDULER and START SCHEDULER commands. Users who are not NetBatch supervisors cannot perform cold-start tasks.

```
> LOGON 255,205, password

> PPD $ZBAT

(Process does not exist)

> FILEINFO $NB.ZBAT.*

No files match \MELBDEV.$NB.ZBAT. *

>
```

Step 2: Run the Scheduler Program

Run the scheduler program NETBATCH, specifying the ! parameter in the RUN command:

```
> NETBATCH /NAME $ZBAT, NOWAIT, PRI 120, CPU 1/ $NB.ZBAT !
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 1,49          $Z142
> FILEINFO $NB.ZBAT.*
$NB.ZBAT
             Code    EOF ... Owner RWEP PExt SExt
BATCHCTL O   847     552 ... 255,255 "OOOO" 2 2
LOGAAA O     847     528 ... 255,205 "NNNC" 50 100
>
```

Considerations

When you run the scheduler program for a cold start (RUN NETBATCH command includes ! parameter), the program:

- Creates the scheduler's primary process under super ID (255,255) ownership.
  - The execution priority of the primary process is one less than that of its TACL creator unless specified otherwise by the PRI run option or the $CMON process. For information about $CMON processes, see the *Guardian Programmer's Guide*.
  - The CPU of the process is the same as that of its TACL creator unless specified otherwise by the CPU run option or the $CMON process.
- Purges existing scheduler files from the database subvolume specified in the RUN NETBATCH command. (NETBATCH purges the files only when the command

includes the ! parameter.) For a list of the files purged, see the description of the ! parameter on page 3-12.

- Creates and opens the scheduler's configuration file BATCHCTL in the database subvolume specified in the RUN NETBATCH command. The file contains default values for the scheduler's attributes, as listed in Table 3-1 on page 3-4.
- Creates and opens the scheduler's log file. The program creates this file by default in the database subvolume specified in the RUN NETBATCH command. The default file ID is LOGxxx, where xxx are sequentially assigned characters in the range AAA-ZZZ (LOGAAA … LOGAAZ, LOGABA … LOGZZZ, LOGAAA …).
- To specify another log file instead of accepting the default, add map DEFINE =_ZBAT_LOG_FILE to the TACL environment before running NETBATCH. For example:

```
6> ADD DEFINE =_ZBAT_LOG_FILE, CLASS MAP, FILE $ZTN0.#PTY6
7> INFO DEFINE =_ZBAT_LOG_FILE
Define Name =_ZBAT_LOG_FILE
CLASS MAP
FILE \MELBDEV.$ZTN0.#PTY6
8> NETBATCH /NAME $ZBAT, NOWAIT/ $NB.ZBAT !
9> BATCHCOM $ZBAT; STATUS SCHEDULER
SCHEDULER STATUS
Process : \MELBDEV.$ZBAT Primary : 0,60
Database: \MELBDEV.$NB.ZBAT
Logfile : \MELBDEV.$ZTN0.#PTY6
.

.
```

Step 3: Create and Initialize the Scheduler's Database Files

Create and initialize the scheduler's database files by using the ADD SCHEDULER command:

```
95> BATCHCOM $ZBAT; ADD SCHEDULER, BACKUPCPU 1,2
Scheduler added
96> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 1,49 $Z142
97> FILEINFO $NB.ZBAT.*
$NB.ZBAT
           Code           EOF … Owner   RWEP   PExt SExt
ATTACH     847            0 …   255,255 "OOOO" 100  200
ATTACH0    847            0 …   255,255 "OOOO" 32   32
BATCHCTL O 847            552 … 255,255 "OOOO" 2    2
CHKQUE     847            0 …   255,255 "OOOO" 64   32
CHKQUE0     847            0 …  255,255 "OOOO" 64   32
EXECUTO0    847            0 …  255,255 "OOOO" 2    2
EXECUTOR    847            0 …  255,255 "OOOO" 6    6
JOB         847P           0 …  255,255 "OOOO" 60   60
JOBCLAS0    847            0 …  255,255 "OOOO" 2    2
JOBCLASS    847            0 …  255,255 "OOOO" 2    2
```

```
LOGAAA O     847              924… 255,205 "NNNC" 50   100
NBATTX       847              0 …  255,255 "OOOO" 16   16
NBATTX0      847              0 …  255,255 "OOOO" 16   16
98>
```

The ADD SCHEDULER command creates and initializes some files in the scheduler's database subvolume. For descriptions of the files, see ADD SCHEDULER Command on page 6-46.

Considerations

The command purges existing scheduler files from the database subvolume before creating the new files if the RUN NETBATCH command omitted the ! parameter. For a list of the files purged, see the description of the ! parameter in Running NETBATCH on page 3-10.

To nominate the CPU for the scheduler's backup process (created by the START SCHEDULER command), specify the BACKUPCPU attribute in the ADD SCHEDULER command.

Step 4: Start the Scheduler

The START SCHEDULER command prepares the scheduler for use. The command creates the scheduler's backup process (in the CPU specified by the scheduler's BACKUPCPU attribute) and opens the database files. To start the scheduler, use the START SCHEDULER command:

```
98> BATCHCOM $ZBAT; START SCHEDULER
Scheduler started
99> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 1,49 2,39 $Z142
100> FILEINFO $NB.ZBAT.*
$NB.ZBAT

Code EOF … Owner RWEP PExt SExt
ATTACH O 847 8192 … 255,255 "OOOO" 100 200
ATTACH0 O 847 0 … 255,255 "OOOO" 32 32
BATCHCTL O 847 552 … 255,255 "OOOO" 2 2
CHKQUE O 847 0 … 255,255 "OOOO" 64 32
CHKQUE0 O 847 0 … 255,255 "OOOO" 64 32
EXECUTO0 O 847 0 … 255,255 "OOOO" 2 2
EXECUTOR O 847 0 … 255,255 "OOOO" 6 6
JOB O 847P 0 … 255,255 "OOOO" 60 60
JOBCLAS0 O 847 0 … 255,255 "OOOO" 2 2
JOBCLASS O 847 0 … 255,255 "OOOO" 2 2
LOGAAA O 847 1584 … 255,205 "NNNC" 50 100
NBATTX O 847 0 … 255,255 "OOOO" 16 16
NBATTX0 O 847 0 … 255,255 "OOOO" 16 16
ATTACH CHKQUE0 EXECUTOR JOBCLAS0 NBATTX
ATTACH0 EXECUTO0 JOB JOBCLASS NBATTX0
CHKQUE
```

Warm Starting a Scheduler

This subsection describes the steps in the warm-start procedure. For information on processing that occurs during a scheduler warm start, see Warm-Start Events on page 3-21.

Step 1: Log On as a NetBatch Supervisor

You must be a NetBatch supervisor to run NETBATCH and to use the START SCHEDULER command. Users who are not NetBatch supervisors cannot perform warm-start tasks.

```
> LOGON 255,205, password
> PPD $ZBAT
(Process does not exist)
> FILEINFO $DATA7.ZBAT.*
$DATA7.ZBAT
Code EOF Last Modification Owner …
ATTACH 847 28672 18-Oct-94 11:05:59 255,255 …
ATTACH0 847 3072 18-Oct-94 11:05:58 255,255 …
BATCHCTL 847 552 18-Oct-94 16:26:36 255,255 …
CHKQUE 847 16384 18-Oct-94 16:25:46 255,255 …
CHKQUE0 847 0 18-Oct-94 16:25:50 255,255 …
EXECUTO0 847 12288 17-Oct-94 15:59:02 255,255 …
EXECUTOR 847 12288 18-Oct-94 16:20:38 255,255 …
JOB 847P 16384 18-Oct-94 16:25:42 255,255 …
JOBCLAS0 847 12288 14-Oct-94 11:02:57 255,255 …
JOBCLASS 847 12288 18-Oct-94 16:20:38 255,255 …
LOGAAA 847 52800 14-Oct-94 14:43:28 255,255 …
LOGAAB 847 660 14-Oct-94 14:51:08 255,255 …
NBATTX 847 0 18-Oct-94 13:57:38 255,255 …
NBATTX0 847 0 18-Oct-94 13:57:38 255,255 …
>
```

Step 2: Run the Scheduler Program

Run the NetBatch program NETBATCH:

- Do not specify the ! parameter in the RUN NETBATCH command when warm starting a scheduler.
- When the scheduler's previous log file was a disk file and the scheduler encounters a file-system error on that file, this warning appears:

  ```
  *WARNING* Scheduler log-file chaining lost: Error file-
  system-error-no.

  Continuing...
  ```

  The warning, which has no effect on the scheduler, indicates a break in the disk log file sequence. As a result, log file scanning by the STATUS-HISTORY command

terminates at the break with a “file not found” error instead of continuing through subsequent files.

```
> NETBATCH /NAME $ZBAT, NOWAIT, CPU 1, PRI 150/ $DATA7.ZBAT
*WARNING* Scheduler log-file chaining lost: Error 11.
Continuing …
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 1,52 $Z161
> FILEINFO $DATA7.ZBAT.*
$DATA7.ZBAT
Code EOF Last Modification Owner …
ATTACH 847 28672 18-Oct-94 11:05:59 255,255 …
ATTACH0 847 3072 18-Oct-94 11:05:58 255,255 …
BATCHCTL O 847 552 18-Oct-94 16:28:24 255,255 …
CHKQUE 847 16384 18-Oct-94 16:28:43 255,255 …
CHKQUE0 847 0 18-Oct-94 16:28:47 255,255 …
EXECUTO0 847 12288 17-Oct-94 15:59:02 255,255 …
EXECUTOR 847 12288 18-Oct-94 16:20:38 255,255 …
JOB 847P 16384 18-Oct-94 16:28:40 255,255 …
JOBCLAS0 847 12288 14-Oct-94 11:02:57 255,255 …
JOBCLASS 847 12288 18-Oct-94 16:20:38 255,255 …
LOGAAA 847 52800 14-Oct-94 14:43:28 255,255 …
LOGAAB 847 660 14-Oct-94 14:51:08 255,255 …
LOGAAD O 847 2772 18-Oct-94 16:28:48 255,205 …
NBATTX 847 0 18-Oct-94 13:57:38 255,255 …
NBATTX0 847 0 18-Oct-94 13:57:38 255,255 …
```

Considerations

When you run the scheduler program NETBATCH for a warm start (RUN NETBATCH command excludes ! parameter), the program:

- Creates the scheduler’s primary process under super ID (255,255) ownership.
  - The execution priority of the primary process is one less than its TACL creator, unless specified otherwise by the PRI run option or the $CMON process. For information about $CMON, see the *Guardian Programmer’s Guide*.
  - The CPU of the process is the same as that of its TACL creator unless specified otherwise by the CPU run option or the $CMON process.
- Opens the scheduler’s configuration file BATCHCTL in the database subvolume specified in the RUN NETBATCH command. The file contains the values of the scheduler’s attributes when the scheduler stopped.
- Creates and opens the scheduler’s log file. By default, the program creates this file in the database subvolume specified in the RUN NETBATCH command. The default file ID is LOGxxx, where xxx are sequentially assigned characters in the range AAA-ZZZ (LOGAAA … LOGAAZ, LOGABA … LOGZZZ, LOGAAA …).

- To specify another log file instead of accepting the default, add map DEFINE =_ZBAT_LOG_FILE to the TACL environment before running NETBATCH. For example:

```
> ADD DEFINE =_ZBAT_LOG_FILE, CLASS MAP, FILE $ZTN0.#PTY6
> INFO DEFINE =_ZBAT_LOG_FILE
Define Name =_ZBAT_LOG_FILE
CLASS MAP
FILE \MELBDEV.$ZTN0.#PTY6
> NETBATCH /NAME $ZBAT, NOWAIT/ $NB.ZBAT
> BATCHCOM $ZBAT; STATUS SCHEDULER
SCHEDULER STATUS
Process : \MELBDEV.$ZBAT Primary : 0,60
Database: \MELBDEV.$NB.ZBAT
Logfile : \MELBDEV.$ZTN0.#PTY6 .
.
```

- A scheduler-created disk log file adopts these attributes of the previous log if that log was a disk file: primary and secondary extent sizes, maximum extents, buffer size, expiration time, RWEP security, owner, and volume and subvolume.
- If the scheduler's log file prior to shutdown was not a disk file, the scheduler automatically reopens that file when warm started. For example, a scheduler that logs to $0 before being shut down continues logging to $0 when restarted.

Step 3: Start the Scheduler

To start the scheduler, use the START SCHEDULER command. The ALTER SCHEDULER command in this example specifies the CPUs for the scheduler's backup process.

```
> BATCHCOM $ZBAT; ALTER SCHEDULER, BACKUPCPU 1,0
Scheduler altered
> BATCHCOM $ZBAT; START SCHEDULER
Scheduler started
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 1,52 0,50 $Z161
> FILEINFO $DATA7.ZBAT.*
$DATA7.ZBAT
Code EOF Last Modification Owner …
ATTACH O 847 28672 18-Oct-94 11:05:59 255,255 …
ATTACH0 O 847 3072 18-Oct-94 11:05:58 255,255 …
BATCHCTL O 847 552 18-Oct-94 16:30:00 255,255 …
CHKQUE O 847 16384 18-Oct-94 16:28:43 255,255 …
CHKQUE0 O 847 0 18-Oct-94 16:28:47 255,255 …
EXECUTO0 O 847 12288 17-Oct-94 15:59:02 255,255 …
EXECUTOR O 847 12288 18-Oct-94 16:30:16 255,255 …
JOB O 847P 16384 18-Oct-94 16:28:40 255,255 …
JOBCLAS0 O 847 12288 14-Oct-94 11:02:57 255,255 …
JOBCLASS O 847 12288 18-Oct-94 16:30:17 255,255 …
LOGAAA 847 52800 14-Oct-94 14:43:28 255,255 …
LOGAAB 847 660 14-Oct-94 14:51:08 255,255 …
LOGAAD O 847 3828 18-Oct-94 16:30:18 255,205 …
```

```
NBATTX O 847 0 18-Oct-94 13:57:38 255,255 …
NBATTX0 O 847 0 18-Oct-94 13:57:38 255,255 …
```

The START SCHEDULER command prepares the scheduler for use. The command creates the scheduler's backup process (in the CPU specified by the scheduler's BACKUPCPU attribute) and opens the database files.

Warm-Start Events

When a scheduler warm starts, these significant events occur:

1. When warm-start processing begins (RUN NETBATCH command excludes the ! parameter):
   a. The scheduler renames its JOB file to JOBSAVED.
   b. The scheduler uses JOBSAVED to build a new JOB file.

      At the end of warm-start processing, the scheduler deletes JOBSAVED. If the scheduler stops before processing ends, JOBSAVED remains in the database for use during the next warm start, after which the scheduler deletes the file. No user intervention is necessary.

      If the JOB file is not secured "OOOO" to the local super ID by using the FUP SECURE command with the PROGID option, warm-start processing fails:

      ```
      25> NETBATCH /NAME $ZBAT, NOWAIT/ $DATA7.ZBAT
      Warning - $ZBAT replied error 201 to the STARTUP message
      - continuing
      2: Process terminated with fatal errors or diagnostics
      NetBatch: ERROR JOB file not CODE 847, SUPER.SUPER,
      SECURE "OOOO", PROGID
      ```

      To correct the problem:

      1. Use the FUP GIVE command to assign JOB file ownership to the local super ID.
      2. Use the FUP SECURE, PROGID command to secure the file "OOOO" with the PROGID option.
2. The scheduler process replies to its startup message, returning the TACL prompt, when JOB file processing is complete. (JOB file processing can take up to five minutes, depending on the number of jobs in the scheduler.) The scheduler returns file-system error 12 (file in use) to an opener such as BATCHCOM until JOB file processing ends.
3. After JOB file processing, the scheduler stops jobs that were executing when the scheduler stopped. (A job can still be executing if the scheduler stopped by some means other than an ABORT SCHEDULER or SHUTDOWN SCHEDULER command.) The state of jobs stopped in this manner is SPECIAL-2 (2:NB failed).

4. After stopping jobs, the scheduler reviews its executors and deletes those whose state was DELETE when the scheduler stopped. The scheduler then updates each executor's state:

State when scheduler stopped	State after warm start
ACTIVE	ON or DOWN (depending on CPU availablity)
DELETE	NA (executor deleted)
DOWN	ON or DOWN (depending on CPU availablity)
OFF	OFF
ON	ON or DOWN (depending on CPU availablity)
STOP	OFF

5. Following its executor review, the scheduler deletes attachment sets that are not in use by jobs and whose attributes include TEMPORARY ON.

On starting up, if the scheduler's INITIATION attribute is set to ON, it immediately begins to select and start jobs. To prevent jobs from starting, set the attribute to OFF before entering the START SCHEDULER command; for example:

```
> NETBATCH /NAME $ZBAT, NOWAIT/ $DATA7.ZBAT
> BATCHCOM $ZBAT; ALTER SCHEDULER, INITIATION OFF; START
SCHEDULER
Scheduler altered
Scheduler started
```

Configuring Schedulers

Scheduler configuration is a task for a NetBatch supervisor to perform after a scheduler cold start. Configuration involves:

- Specifying the scheduler's attributes
- Adding the scheduler's classes
- Adding and starting the scheduler's executors

Scheduler Configuration Commands

This table lists the BATCHCOM commands to configure a scheduler. For information on the syntax, operation, and results of the commands, see Section 6, Commands. For information on attributes you can specify with the commands, see Section 7, Attributes.

Command	Function
OPEN SCHEDULER	Specifies the target scheduler
ALTER SCHEDULER	Specifies the scheduler's attributes as listed in Table 3-1 on page 3-4
ADD CLASS	Adds the scheduler's classes and specifies their INITIATION attributes
ADD EXECUTOR	Adds the scheduler's executors and specifies their CLASS and CPU attributes
START EXECUTOR	Makes the newly added executors available for use by jobs

Scheduler Configuration Procedure

Step 1: Log On as a NetBatch Supervisor

```
> LOGON 255,205, password
>
```

Step 2: Start an Interactive BATCHCOM Session

Start an interactive BATCHCOM session and open the scheduler you want to configure:

```
> BATCHCOM $ZBAT
BATCHCOM - T9190D30 …
NETBATCH SERVER - T9190D30 …
1}
```

Step 3: Specify the Scheduler's Attributes

Specify the scheduler's attributes by using the ALTER SCHEDULER command:

```
1} ALTER SCHEDULER, DEFAULT-CLASS C1, DEFAULT-OUT
$S.#ZBAT, DEFAULT-PRI 149, TAPEDRIVES 3
```

```
Scheduler altered
2}
```

Step 4: Add the Scheduler's Attributes and Specify Their INITIATION Attributes

Add the scheduler's classes and specify their INITIATION attributes by using the ADD CLASS command. (Classes adopt the attribute INITIATION ON unless specified otherwise.)

```
2} ADD CLASS C1; ADD CLASS C2; ADD CLASS C3;
ADD CLASS C4, INITIATION OFF
Class C1 added
Class C2 added
Class C3 added
Class C4 added
3}
```

Step 5: Add the Scheduler's Executors and Specify Their CLASS and CPU Attributes

Add the scheduler's executors and specify their CLASS and CPU attributes by using the ADD EXECUTOR command:

```
3} ADD EXECUTOR E1, CLASS (C1, C2, C3, C4), CPU 0
Executor E1 added
4} ADD EXECUTOR E4, CLASS (C2, C3, C4, C1), CPU 1
Executor E4 added
5} ADD EXECUTOR E8, CLASS (C3, C4, C1, C2), CPU 2
Executor E8 added
6} ADD EXECUTOR E12, CLASS (C4, C1, C2, C3), CPU 3
Executor E12 added
7}
```

Step 6: Start the Scheduler's Executors

Start the scheduler's executors by using the START EXECUTOR command:

```
7} START EXECUTOR *
Executor E1 started
Executor E12 started
Executor E4 started
Executor E8 started
8}
```

Step 7: End the BATCHCOM Session

End the BATCHCOM session by using the EXIT command:

```
8} EXIT
>
```

Displaying Scheduler Configuration

You can display a scheduler's configuration, its attributes, and the attributes of its classes and executors by using these commands. For information about the syntax, operation, and results of the commands, see Section 6, Commands and Managing Schedulers on page 3-29.

Command	Function
STATUS SCHEDULER	Displays scheduler details that include: ● The scheduler's process name, primary and backup CPUs and PINs, database location, log file name, and current time ● A count of the scheduler's executors, classes, and jobs by state ● The number of active and suspended job processes, configured and in-use tape drives, and attachment sets
INFO SCHEDULER	Lists scheduler attributes
INFO CLASS	Lists class attributes
STATUS EXECUTOR	Displays executors' names, CPUs, states, and for an executor in use by a job, the job's number and its class name
INFO EXECUTOR	Lists executor attributes

Examples

- This example shows the result of a STATUS SCHEDULER command:

```
1} STATUS SCHEDULER
SCHEDULER STATUS
Process : \MELBDEV.$ZBAT Primary : 1,25 Backup : 2,36
Database: \MELBDEV.$NB.ZBAT
Logfile : \MELBDEV.$NB.ZBAT.LOGAAB
Time : 19OCT94 12:40:24
EXECUTORS JOBS CLASSES PROCESSES
---------- -------------- -------- --------------
OFF 0 READY 0 OFF 1 ACTIVE 0
ON 4 EXECUTING 0 ON 3 SUSPENDED 0
ACTIVE 0 SPECIAL 0
STOP 0 TIME 0
DOWN 0 EVENT 0 TAPEDRIVES ATTACHMENT SETS
DELETE 0 SUSPENDED 0 ------------ --------------
RUNNEXT 0 CONFIGURED 3 DEFINED 0
RUNNOW 0 IN USE 0
TAPE 0
```

- This example shows the result of an INFO SCHEDULER command:

```
2} INFO SCHEDULER
SCHEDULER ATTRIBUTES
backupcpu: 1,2
at-allowed: Off
submit-allowed: On
initiation: On
```

```
default-executor-program: \MELBDEV.$SYSTEM.SYSTEM.TACL
max-pri: 199
default-pri: 149
max-concurrent-jobs: 500,500
default-selpri: 3
default-maxprintlines: None
default-maxprintpages: None
tapedrives: 3
default-out: \MELBDEV.$S.#ZBAT
default-class: C1
default-stall: Off
default-stop-on-abend: Off
default-highpin: Off
catchup: On
ems: Off
```

- This example shows the result of an INFO CLASS command:

```
3} INFO CLASS C1
CLASS ATTRIBUTE for C1
initiation: On
```

- This example shows the result of a STATUS EXECUTOR command:

```
4} STATUS EXECUTOR *
EXECUTOR STATUS
```

```
EXECUTOR                 CPU STATE  JOB    CLASS
------------------------ --- ------ ----- ------------------
E1                       0    ON
E12                      3    ON
E4                       1    ON
E8                       2    ON
```

- This example shows the result of a INFO EXECUTOR command:

```
5} INFO EXECUTOR E1
EXECUTOR ATTRIBUTES for E1
cpu: 0
classes: C1
C2
C3
C4
```

Automating Scheduler Configuration

To automate scheduler configuration, have BATCHCOM process an EDIT input file containing the configuration commands. To create the configuration file, do one of:

- Use the EDIT program.
- Execute INFO SCHEDULER, INFO CLASS *, and INFO EXECUTOR * commands that include the OBEY-FORM qualifier and send their output to a disk file.

Using EDIT to Create a Scheduler Configuration File

Step 1: Create and Open a File for the Scheduler's Configuration Commands

Create and open a file for the scheduler's configuration commands by using the EDIT program:

```
12> EDIT SCHDCONF !
CURRENT FILE IS $DATA7.SCHD.SCHDCONF
```

Step 2: Add the Configuration Commands to the File

Add the configuration commands to the file, then close the file and stop the EDIT program:

```
*ADD
1 ALTER SCHEDULER, AT-ALLOWED ON, DEFAULT-EXECUTOR-
PROGRAM FUP
2 ADD CLASS DEFAULT
3 ADD EXECUTOR EXEC-0, CLASS DEFAULT, CPU 0
4 START EXECUTOR EXEC-0
5 //
*EXIT
>
```

Using INFO Commands to Create a Scheduler Configuration File

The prerequisite to using these commands is a running scheduler whose attributes you want to capture in BATCHCOM-command format.

Step 1: Execute INFO Commands and Send Their Output to a Disk File

Execute INFO SCHEDULER, INFO CLASS *, and INFO EXECUTOR * commands that include the OBEY-FORM qualifier and send their output to a disk file.

Note. To add job details to the file, use the INFO JOB* command with the OBEY-FORM qualifier:

```
> BATCHCOM $ZBAT; INFO SCHEDULER /OUT ZBATCONF/, OBEY-FORM;
INFO CLASS /OUT ZBATCONF/ *, OBEY-FORM; INFO EXECUTOR /OUT
ZBATCONF/ *, OBEY-FORM
>
```

Step 2: Display the Commands in the File

Display the commands in the file by using the EDIT program, then add, update, and delete commands as necessary:

```
> EDIT ZBATCONF; LIST ALL
CURRENT FILE IS $NB.TEST.ZBATCONF
```

```
1 ASSUME SCHEDULER
2 RESET
3 ALTER, BACKUPCPU 1,2
4 ALTER, AT-ALLOWED OFF
5 ALTER, SUBMIT-ALLOWED ON
6 ALTER, INITIATION ON
7 ALTER, DEFAULT-EXECUTOR-PROGRAM
\MELBDEV.$SYSTEM.SYSTEM.TACL
8 ALTER, MAX-PRI 199
9 ALTER, DEFAULT-PRI 149
10 ALTER, MAX-CONCURRENT-JOBS 500,500
11 ALTER, DEFAULT-SELPRI 3
12 ALTER, DEFAULT-MAXPRINTLINES NONE
13 ALTER, DEFAULT-MAXPRINTPAGES NONE
14 ALTER, TAPEDRIVES 3
15 ALTER, DEFAULT-OUT \MELBDEV.$S.#ZBAT
16 ALTER, DEFAULT-CLASS C1
17 ALTER, DEFAULT-STALL OFF
18 ALTER, DEFAULT-STOP-ON-ABEND OFF
19 ALTER, DEFAULT-HIGHPIN OFF
20 ALTER, CATCHUP ON
21 ALTER, EMS OFF
22 ADD CLASS C1, INITIATION ON
23 ADD CLASS C2, INITIATION ON
24 ADD CLASS C3, INITIATION ON
25 ADD CLASS C4, INITIATION OFF
26 ADD EXECUTOR E1, CPU 0, CLASS (C1, C2, C3, C4)
27 ADD EXECUTOR E12, CPU 3, CLASS (C4, C1, C2, C3)
28 ADD EXECUTOR E4, CPU 1, CLASS (C2, C3, C4, C1)
29 ADD EXECUTOR E8, CPU 2, CLASS (C3, C4, C1, C2)
*ADD LAST
30 START EXECUTOR *
31 //
*EXIT
```

Configuring a Scheduler From a Configuration File

After creating a scheduler configuration file, you can use it to configure a scheduler by having BATCHCOM process it. To do this, run BATCHCOM with the IN run option specifying the configuration file and the scheduler process specified as a program parameter. For example:

```
> BATCHCOM /IN ZBATCONF/ $SCHD
BATCHCOM - T9190D30 - (31OCT94^01JUN94)
NETBATCH SERVER - T9190D30 …
ASSUME SCHEDULER
RESET
ALTER, BACKUPCPU 1,2
Scheduler altered .
.
```

Managing Schedulers

This subsection provides instructions for various scheduler management tasks:

Topic	Page
Determining Whether a Scheduler Is Running	3-29
Displaying Scheduler, Executor, and Class Status	3-29
Altering Executor and Class Attributes	3-33
Stopping and Restarting Executors	3-34
Deleting Executors and Classes	3-35
Dealing With Scheduler Log Files	3-38
Enabling and Disabling EMS Event-Message Generation	3-44
Stopping a Scheduler	3-45

Determining Whether a Scheduler Is Running

To find out if a specified scheduler is running or to list all running schedulers, use the TACL STATUS command. The command variants that let you perform these tasks are:

- `STATUS [\`*`node`*`.]$`*`process-name`*

  For example:

  ```
  > STATUS \MELBDEV.$ZBAT
  ```

- `STATUS [\`*`node`*`.]*, PROG [`*`program-file-name`*`]`
  `STATUS [\`*`node`*`.]*, PROG [`*`program-file-name-template`*`]`

  For example:

  ```
  > STATUS *, PROG $SYSTEM.SYSTEM.NETBATCH
  ```

For more information about the STATUS command, see the *TACL Reference Manual*.

Displaying Scheduler, Executor, and Class Status

To display scheduler, executor, and class status, use these BATCHCOM commands:

- STATUS SCHEDULER
- STATUS EXECUTOR
- INFO CLASS

Displaying Scheduler Status

The STATUS SCHEDULER command displays scheduler details that include:

- The scheduler's process name, primary and backup CPUs and PINs, database location, log file name, and current time

- A count of the scheduler's executors, classes, and jobs by state
- The number of active and suspended job processes, configured and in-use tape drives, and attachment sets

For example:

```
1} STATUS SCHEDULER
SCHEDULER STATUS
Process : \MELBDEV.$NBAT Primary : 0,263 Backup : 1,264
Database: \MELBDEV.$NB.NBAT
Logfile : \MELBDEV.$NB.NBAT.LOGAAO
Time : 19OCT94 13:11:11
EXECUTORS JOBS CLASSES PROCESSES
---------- -------------- -------- --------------
OFF 1 READY 0 OFF 0 ACTIVE 0
ON 7 EXECUTING 0 ON 11 SUSPENDED 0
ACTIVE 0 SPECIAL 12
STOP 0 TIME 6
DOWN 0 EVENT 1 TAPEDRIVES ATTACHMENT SETS
DELETE 0 SUSPENDED 0 ------------ --------------
RUNNEXT 0 CONFIGURED 3 DEFINED 15
RUNNOW 0 IN USE 0
TAPE 0
```

Displaying Executor Status

The STATUS EXECUTOR command displays executors' names, CPUs, states, and for an executor in use by a job, the job's number and its class name. For example:

```
31} STATUS EXECUTOR *
EXECUTOR STATUS
_____________________________________________________________

EXECUTOR                 CPU  STATE  JOB   CLASS
------------------------ --- ------ ----- -------------------
CLX-0                     0   STOP   99    MANUFACTURING
CLX-0-SPARE1              0   OFF
CLX-0-SPARE2              0   DELETE 102   ACCOUNTS
CLX-1                     1   ACTIVE 100   SALES
CLX-1-SPARE               1   ON
CLX-2                     2   ACTIVE 101   DISTRIBUTION
CLX-2-SPARE               2   ON
CLX-3                      3   DOWN
```

Executor States

This table lists and describes executor states. For more information on the states, see STATUS EXECUTOR Command on page 6-163.

State	Description
ACTIVE	The executor is in use by a job.
DELETE	The executor is in use by a job. When the job finishes, the scheduler deletes the executor.
DOWN	The executor is not available for use because its CPU is down.
OFF	The executor is not available for use.
ON	The executor is available for use.
STOP	The executor is in use by a job, but was the subject of a STOP EXECUTOR command. When the job finishes (that is, its executor program stops), the scheduler changes the executor's state to OFF.

Displaying Class Information

The INFO CLASS command displays the values of class INITIATION attributes:

```
16} INFO CLASS *
CLASS ATTRIBUTE for C1
initiation: On
CLASS ATTRIBUTE for C2
initiation: On
CLASS ATTRIBUTE for C3
initiation: Off
```

Altering Scheduler Attributes

To change a scheduler's attributes, NetBatch supervisors can use the ALTER SCHEDULER command. For example:

```
6} INFO SCHEDULER, AT-ALLOWED, DEFAULT-OUT
SCHEDULER ATTRIBUTES
at-allowed: On
default-out: \MELBDEV.$S.#BATCH
7} ALTER SCHEDULER, AT-ALLOWED OFF, DEFAULT-OUT $NULL
Scheduler altered
8} INFO SCHEDULER, AT-ALLOWED, DEFAULT-OUT
SCHEDULER ATTRIBUTES
at-allowed: Off
default-out: \MELBDEV.$NULL
```

Except BACKUPCPU, attributes changed by the ALTER SCHEDULER command take effect on command execution. The command changes that attribute's value in the scheduler's database but does not force an actual backup CPU change. Such a change occurs only when the scheduler creates a new backup process (for example,

when one of the scheduler's processes stops or during scheduler startup). For example:

```
> PPD $SCHD
Name Primary Backup Ancestor
$SCHD 1,49 0,59 $Z185
> BATCHCOM $SCHD; ALTER SCHEDULER, BACKUPCPU 2,3
Scheduler altered
> PPD $SCHD
Name Primary Backup Ancestor
$SCHD 1,49 0,59 $Z185
> STOP 1,49
> PPD $SCHD
Name Primary Backup Ancestor
$SCHD 0,59 2,39 $Z185
```

Switching a Scheduler's CPUs

NetBatch supervisors can switch a scheduler's primary process to the CPU of its backup process and the backup process to the CPU of its primary process by using the SWITCHCPU SCHEDULER command. For example:

```
> PPD $NBAT
Name Primary Backup Ancestor
$NBAT 0,263 1,264 $C3
> BATCHCOM $NBAT; SWITCHCPU SCHEDULER
Scheduler CPUs switched
> PPD $NBAT
Name Primary Backup Ancestor
$NBAT 1,264 0,263 $C3
```

The SWITCHCPU SCHEDULER command lets you improve scheduler performance when, for example, the scheduler's primary process is competing for system resources in a busy CPU. By switching CPUs, you can run the scheduler's primary process in the less heavily used CPU of its backup.

The SWITCHCPU SCHEDULER command switches scheduler processes only between the current primary and backup CPUs. It does not switch the primary process to a different backup CPU as specified by the scheduler's BACKUPCPU attribute. (BACKUPCPU applies only when the scheduler creates a new backup process—for example, when one of the scheduler's processes stops or during scheduler startup.) For example:

```
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 0,268 3,269 $C8
> BATCHCOM $ZBAT; ALTER SCHEDULER, BACKUPCPU 1
Scheduler altered
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 0,268 3,269 $C8
> BATCHCOM $ZBAT; SWITCHCPU SCHEDULER
Scheduler CPUs switched
> PPD $ZBAT
```

```
Name Primary Backup Ancestor
$ZBAT 3,269 0,268 $C8
> BATCHCOM $ZBAT; SHUTDOWN SCHEDULER
Scheduler shutting down
> NETBATCH /NAME $ZBAT, NOWAIT/ $DATA7.ZBAT
> BATCHCOM $ZBAT; START SCHEDULER
Scheduler started
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 0,267 1,268 $C8
```

Altering Executor and Class Attributes

NetBatch supervisors can alter the attributes of a scheduler's executors and classes by using these BATCHCOM commands:

- ALTER EXECUTOR
- ALTER CLASS

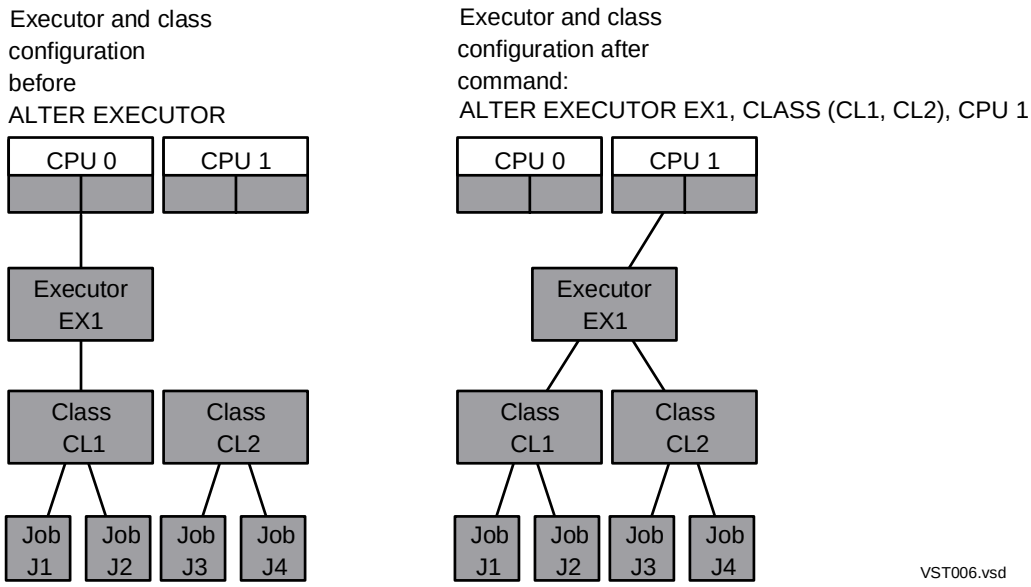
Altering Executor Attributes

The ALTER EXECUTOR command changes an executor's CLASS and CPU attributes. For example:

```
4} INFO EXECUTOR EX1
EXECUTOR ATTRIBUTES for EX1
cpu: 0
classes: CL1
5} ALTER EXECUTOR EX1, CLASS (CL1, CL2), CPU 1
Executor EX1 altered
6} INFO EXECUTOR EX1
EXECUTOR ATTRIBUTES for EX1
cpu: 1
classes: CL1
CL2
```

The CLASS attribute specifies the executor's classes. The CPU attribute specifies its CPU. Together, they route jobs from the classes through the executor to the CPU. Figure 3-4 illustrates the preceding example.

Figure 3-4. Example of ALTER EXECUTOR Command



Altering Class Attributes

The ALTER CLASS command switches a class's INITIATION attribute from ON to OFF or from OFF to ON. For example:

```
2} INFO CLASS DEFAULT
CLASS ATTRIBUTE for DEFAULT
initiation: On
3} ALTER CLASS DEFAULT, INITIATION OFF
Class DEFAULT altered
4} INFO CLASS DEFAULT
CLASS ATTRIBUTE for DEFAULT
initiation: Off
```

The INITIATION attribute determines whether jobs from the class are available for execution (ON—jobs available; OFF—jobs unavailable). Therefore, you can use the attribute to control the flow of jobs from the class to its executors. For example, setting INITIATION OFF for a class prevents jobs assigned to the class from executing.

Stopping and Restarting Executors

NetBatch supervisors can stop a scheduler's executors and restart them by using these BATCHCOM commands:

- STOP EXECUTOR
- START EXECUTOR

Stopping Executors

The STOP EXECUTOR command stops an executor whose state is ACTIVE or ON. For example:

```
23} STATUS EXECUTOR *
EXECUTOR STATUS

_____________________________________________________________________
EXECUTOR                  CPU STATE  JOB   CLASS
------------------------ --- ------ ----- ---------
EX1                       1  ACTIVE  1    CL1
EX2                       2  ON
24} STOP EXECUTOR *
Executor EX1 stopped
Executor EX2 stopped
25} STATUS EXECUTOR *
EXECUTOR STATUS
EXECUTOR                  CPU STATE  JOB   CLASS
------------------------ --- ------ ----- ---------
EX1                       1  STOP    1    CL1
EX2                        2  OFF
```

Stopping an executor prevents jobs from using it to access CPU resources.

Restarting Executors

The START EXECUTOR command starts a stopped executor (state OFF or STOP). For example:

```
34} STATUS EXECUTOR *
EXECUTOR STATUS

_____________________________________________________________________
EXECUTOR                  CPU STATE  JOB    CLASS
------------------------ --- ------ ----- ---------
EX1                       1   STOP   1     CL1
EX2                       2   OFF
35} START EXECUTOR *
Executor EX1 started
Executor EX2 started
36} STATUS EXECUTOR *
EXECUTOR STATUS
EXECUTOR                  CPU  STATE JOB    CLASS
------------------------ --- ------ ----- ---------
EX1                       1   ACTIVE 1     CL1
EX2 2 ON
```

Starting a stopped executor lets jobs from the executor’s classes run in its CPU.

Deleting Executors and Classes

NetBatch supervisors can stop a scheduler’s executors and restart them by using these BATCHCOM commands:

- DELETE EXECUTOR
- DELETE CLASS

Deleting Executors

The DELETE EXECUTOR command deletes an executor from a scheduler. For example:

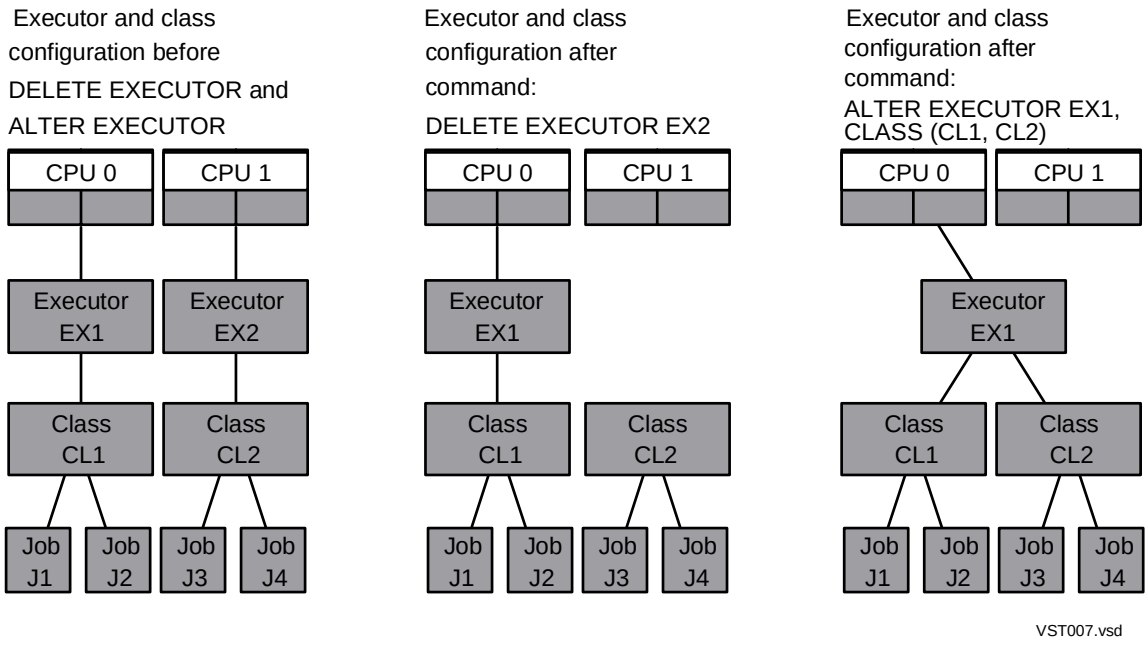
```
4} DELETE EXECUTOR EXEC1
Executor EXEC1 deleted
```

After deleting an executor, reassign classes that were unique to it to other executors. Otherwise, the scheduler never scans the classes for jobs. For example:

```
19} INFO EXECUTOR *, CLASS
EXECUTOR ATTRIBUTES for EX1
classes: CL1
EXECUTOR ATTRIBUTES for EX2
classes: CL2
20} DELETE EXECUTOR EX2
Executor EX2 deleted
21} ALTER EXECUTOR EX1, CLASS (CL1, CL2)
Executor EX1 altered
```

Figure 3-5 on page 3-36 illustrates this example.

Figure 3-5. Example of DELETE EXECUTOR Command



Deleting Classes

The DELETE CLASS command deletes a class from a scheduler. For example:

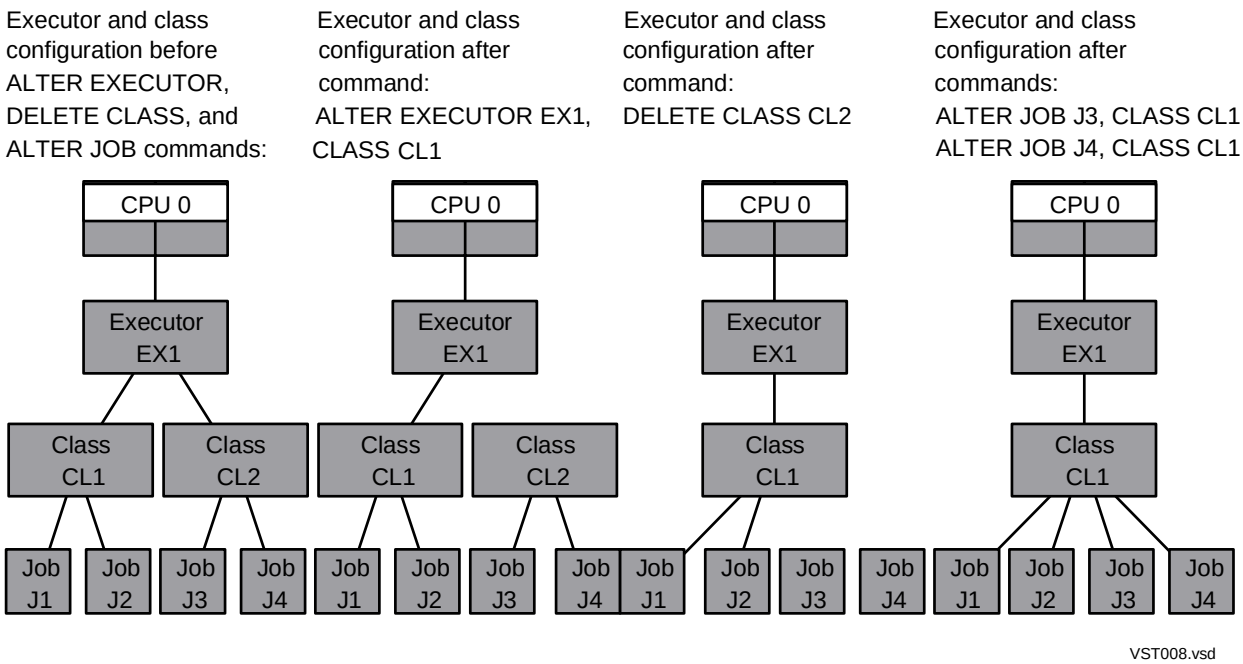
```
12} DELETE CLASS ENFORM-JOBS
Class ENFORM-JOBS deleted
```

The scheduler disallows deletion of a class assigned to an executor. To delete such a class, dissociate it from the executor by using the ALTER EXECUTOR command before the DELETE CLASS command. For example:

```
33} DELETE CLASS CL2
2104-E CLASS CL2 is in use by one or more executors. To
delete the CLASS, remove it from the executors first.
34} INFO EXECUTOR *, CLASS
EXECUTOR ATTRIBUTES for EX1
classes: CL1
CL2
35} ALTER EXECUTOR EX1, CLASS CL1
Executor EX1 altered
36} DELETE CLASS CL2
Class CL2 deleted
```

Deleting a class does not delete the class's jobs. However, the jobs are never available for execution unless you reassign them to another class by using the ALTER JOB command. Alternatively, add a new class with the same name as the deleted class. Figure 3-6 on page 3-37 illustrates class deletion and reassignment of a deleted class's jobs.

Figure 3-6. Example of DELETE CLASS Command



Dealing With Scheduler Log Files

The scheduler records in a log file details of events that occur while it is running. You can use the log, for example, for troubleshooting purposes. The log can be a device, a process, or any type of unstructured disk file except an EDIT file.

Figure 3-7 on page 3-38 shows a commented example of a scheduler log file. The example lists events such as scheduler startup, class and executor configuration, job submission and execution, and scheduler shutdown. For explanations of the example's event keywords and a breakdown of the logging formats used in the event descriptions, see Table 3-3 on page 3-41 and Table 3-4 on page 3-43.

Figure 3-7. Example of a Scheduler Log File

```
NETBATCH SCHEDULER/MONITOR - T9190D20 ...                                          Log file header

03MAR93  14:49:02  CREATE  Scheduler Primary \MELBDEV.$SCHD CPU 1                   Primary scheduler process started (CPU 1)
03MAR93  14:49:02  COLD    Purge any existing scheduler database files             Cold start indicated
03MAR93  14:49:13  OPEN    U_255,255 P_BATCHCOM \MELBDEV.$:0:83:17941432           BATCHCOM opens scheduler
03MAR93  14:49:38  ADD     SCHEDULER \MELBDEV.$:0:83:17941432 U_255,255            ADD SCHEDULER command
                           H_\MELBDEV.$T4.#A                                       executed
03MAR93  14:49:44  START   SCHEDULER \MELBDEV.$:0:83:17941432 U_255,255            START SCHEDULER command
                           H_\MELBDEV.$T4.#A                                       executed
03MAR93  14:49:46  CREATE  Scheduler Backup \MELBDEV.$SCHD CPU 2 U_255,255        Backup scheduler process
                           H_\MELBDEV.$T4.#A                                       started
03MAR93  14:49:49  START   \MELBDEV.$SCHD:20110521                                Confirmation that backup process
                                                                                   started
03MAR93  14:49:59  ADD     CLASS DEFAULT U_255,255 H_\MELBDEV.$T4.#A               ADD CLASS command
                                                                                   executed
03MAR93  14:50:40  ADD     EXECUTOR E0 CPU 0 U_255,255 H_\MELBDEV.$T4.#A           ADD EXECUTOR command
                                                                                   executed
03MAR93  14:50:51  START   EXECUTOR E0 U_255,255 H_\MELBDEV.$T4.#A                START EXECUTOR command
                                                                                   executed
03MAR93  14:50:52  UPDATE  EXECUTOR E0 S_ON U_255,255 H_\MELBDEV.$T4.#A           Executor state changed to
                                                                                   ON
03MAR93  14:57:15  ADD     JOB ABC C_DEFAULT:3 J_1 U_255,255 H_\MELBDEV.$T4.#A    SUBMIT JOB command
                                                                                   executed
03MAR93  14:57:15  LIST    JOB ABC READY J_1 U_255,255 H_\MELBDEV.$T4.#A          Job state changed to
                                                                                   READY
03MAR93  14:57:19  BEGIN   JOB (SUPER.SUPER)ABC E_E0 L_1362 J_1 U_255,255        Executor program startup
                                                                                   begins
03MAR93  14:57:19  UPDATE  EXECUTOR E0                                          Executor selected; state changed to
                           S_ACTIVE                                                ACTIVE
03MAR93  14:57:20  UPDATE  JOB ABC J_1                                          Job state changed to
                                                                                   EXECUTING
03MAR93  14:57:20  LIST    JOB ABC EXECUTING J_1
03MAR93  14:57:20  START   EXECUTOR-PROGRAM U_255,255 J_1 P_TACL                 Confirmation that executor program
                           \MELBDEV.$Z587:20112569                                 started
03MAR93  14:57:39  STOP    CC_0 EXECUTOR-PROGRAM J_1 \MELBDEV.$Z587:20112569      Confirmation that executor program
                                                                                   stopped
03MAR93  14:57:39  UPDATE  EXECUTOR E0                                          Executor released and state changed
                           S_ON                                                    to
03MAR93  14:57:39  FINISH  JOB ABC T_0:0:5:100 J_1 P_TACL                        ON
                                                                                   Confirmation that job
                                                                                   finished
03MAR93  14:57:39  DELETE  JOB ABC J_1                                          Job deleted from scheduler
                                                                                   database
03MAR93  15:01:12  SHUTDN  SHUTDOWN SCHEDULER \MELBDEV.$:0:83:17941432           SHUTDOWN SCHEDULER command
                           U_255,255 H_\MELBDEV.$T4.#A                             executed
03MAR93  15:01:12  DOSTOP  Myself U_255,255 H_\MELBDEV.$T4.#A                    Backup scheduler process
                                                                                   stopped

Event timestamp    Event keyword    Event description                                          VST103.vsd
```

Creating and Opening a Scheduler Log File

The scheduler opens its log file when you run the NETBATCH program during scheduler startup. For more information, see Warm Starting a Scheduler on page 3-18.

Sizing a Scheduler Log File

A scheduler-created disk log file adopts these attributes of the previous log if that log was a disk file: primary and secondary extent sizes, maximum extents, buffer size, expiration time, RWEP security, owner, and volume and subvolume.

The default extent size of a scheduler-created disk log file is 50 primary extents and 100 secondary extents. The default maximum extents the file can have is 100. (A file of this size can record up to 154,375 scheduler events.) To change the size of a disk log file, use the FUP ALTER command. For information on this command, see the *File Utility Program (FUP) Reference Manual*.

When a disk log file is full, the scheduler automatically closes it and creates and opens a default log file in the same subvolume.

Switching a Scheduler Log File

NetBatch supervisors can close a running scheduler's log file and open another by using the SWITCHLOG SCHEDULER command. You can specify the file you want created and opened, or let the scheduler create and open a default file:

```
1} STATUS SCHEDULER .
Logfile : \MELBDEV.$NB.ZBAT.LOGAAB .
2} SWITCHLOG SCHEDULER
Scheduler logfile switched
3} STATUS SCHEDULER .
Logfile : \MELBDEV.$NB.ZBAT.LOGAAC .
```

This SWITCHLOG SCHEDULER command opens a specified log file:

```
4} STATUS SCHEDULER .
Logfile : \MELBDEV.$NB.ZBAT.LOGAAC .
5} SWITCHLOG SCHEDULER $ZTN0.#PTY6
Scheduler logfile switched
6} STATUS SCHEDULER .
Logfile : \MELBDEV.$ZTN0.#PTY6 .
```

You cannot specify a DEFINE in the SWITCHLOG SCHEDULER command.

Listing the Contents of a Scheduler Log File

You can list the contents of an open or closed scheduler log file by using the FUP COPY command. For example, this command lists the contents of an open log file and wraps lines that are too long to the next line:

```
> FUP COPY LOGAFO,,SHARE,FOLD
```

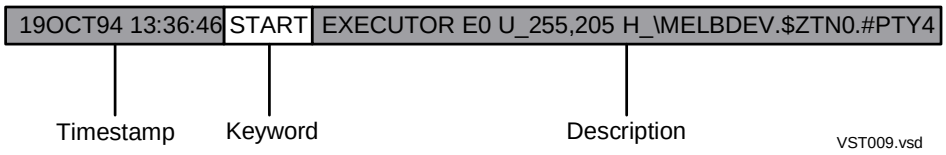
You also can use the EDIT program to list a closed log file's contents after putting the contents into an EDIT file. For example, this command creates file LOGTEMP, puts the contents of LOGADQ into that file, then lists LOGTEMP:

```
> EDIT; GET LOGADQ PUT LOGTEMP; LIST ALL
```

For information about the FUP COPY command, see the *File Utility Program (FUP) Reference Manual*. For information about the EDIT program, see the *EDIT User's Guide and Reference Manual*.

Interpreting the Contents of a Scheduler Log File

The log file records scheduler events in chronological order. Each event has a timestamp, a keyword identifying the event type, and an event description. For example:



For explanations of event keywords and a breakdown of the logging formats used in event descriptions, see Table 3-3 and Table 3-4 on page 3-43.

The NETBATCH log file is an unstructured file with a record length of 132 bytes. If the information to be logged is greater than 132 bytes, it is split into two consecutive records. The fields used in the records are:

<Timestamp>: contains the date and time of the event.

<Main Event>: contains the event that has occurred.

<Jobno>: contains the job number associated with the event.

<Description>: contains information on the attributes linked with the event.

The format in which the information will be split is shown below.

Case 1: A job related event will be logged as

Record 1: <Timestamp> <Main Event> <Jobno>

Record 2: <Timestamp> <Description>

For example, the BEGIN JOB event will be logged as

```
Record 1: 06APR06 12:44:41 BEGIN JOB (SUPER.SUPER)
J23456789012345678901234:1 E_E23456789012345678901234 L_12 J_1
Record 2: 06APR06 12:44:41 J_1 P_DELAY \NODE1.$Z0H5:38494157
U_255,255
```

Case 2: A job independent event will be logged as

Record 1: <Timestamp> <Main Event>

Record 2: <Timestamp> <Description>

For example, the SWITCHLOG event will be logged as

```
Record 1: 06APR06 12:41:42 LOG    SWITCHLOG
\NODE1.$DSMSCM.S2345678.LOGAAA TO \NODE1.$DSMSCM.S2345678.LOGAAB
Record 2: 06APR06 12:41:42 U_255,255 H_\NODE1.$ZTNT.#PT1RBGU
```

Event Keywords

Table 3-3. Scheduler Log File Keywords (page 1 of 3)

Keyword	Description	Example
A'VATE	Suspended job reactivated by ACTIVATE JOB command.	A'VATE JOB XYZ J_1 U_255,255H_\MELBDEV.$T4.#A
ABEND	ABEND system message received from process specified in event.	ABEND CC_5 EXECUTOR-PROGRAM J_2\MELBDEV.$X224:23831737
ABENDT	ABEND system message text.	ABENDT TACL fatal error: Couldn't open TACL IN J_2 \MELB DEV.$X224:23831737
ABORT	Job aborted because of event or request (for example, execution of STOP JOB command).	ABORT JOB XYZ by NetBatch J_1 P_TACL\MELBDEV.$X206:23826617 U_255,255H_\MELBDEV.$T4.#A
ADD	Object added in response to user request.	ADD JOB XYZ C_DEFAULT:3 J_1U_255,255H_\MELBDEV.$T4.#A
ALIVE	One process of a NonStop process pair stopped.	ALIVE Process \MELBDEV.$X806:26047161 stopped J_51 \MELBDEV.$X829:26053049
ALTER	Object altered in response to user request.	ALTER EXECUTOR E0 U_255,255H_\MELBDEV.$T4.#A
BACKUP	Backup CPU specified for scheduler not available, so backup created in another CPU.	BACKUP CPU 3 not available U_255,255H_\MELBDEV.$T4.#A
BEGIN	Job initiation (executor-program process started).	BEGIN JOB (SUPER.SUPER)XYZ E_E1L_1135 J_1 U_255,255
CHKPNT	Scheduler's backup process took over.	CHKPNT takeover from DOIT^SCHEDULER+%003036I U_255,255H_\MELBDEV.$T4.#A
CLOCK	System clock changed.	CLOCK CHANGED 3458221102 microseconds, TIME ADJUSTMENT
CLOSE	CLOSE system message received from process specified in event.	CLOSE \MELBDEV.$:0:31:20665784U_255,255

Table 3-3. Scheduler Log File Keywords (page 2 of 3)

Keyword	Description	Example
COLD	Scheduler cold started.	COLD Purge any existing scheduler database files
CREATE	Primary or backup scheduler process created.	CREATE Scheduler Backup \MELBDEV.$SCHD CPU 2 U_255,255 H_\MELBDEV.$T4.#A
DELETE	Object deleted in response to user request.	DELETE JOB XYZ J_1 U_255,255 H_\MELBDEV.$T4.#A
DOSTOP	Process stopped by scheduler.	DOSTOP \MELBDEV.$X206:23826617(STOP return error 638) J_1U_255,255 H_\MELBDEV.$T4.#A
DOWN	CPU failed or executor made unavailable because of CPU failure.	DOWN CPU \MELBDEV.3
ERROR	Nonfatal error condition detected by scheduler.	ERROR EMS log fail SPI error -5 U_255,255H_\MELBDEV.$T4.#A
FINISH	Job completion (executor-program process terminated).	FINISH JOB XYZ T_0:0:0:0 J_1 P_TACL U_194,53 H_\MELBDEV.$T4.#A
LIMIT	Job exceeded its execution time limit.	LIMIT Exceeded J_1
LIST	Job state updated.	LIST JOB XYZ SPECIAL-1 J_1 U_255,255 H_\MELBDEV.$T4.#A
LOG	SWITCHLOG SCHEDULER command executed.	LOG SWITCHLOG =_ZBAT_LOG_FILE to $DATA7.SCHD.LOGAAS U_255,255 H_\MELBDEV.$T4.#A
MSG	Home terminal or OUT message written to scheduler by process specified in event.	MSG * ERROR * Process name error: File in use J_11 P_TACL \MELBDEV.$DMG:24872633
OPEN	OPEN system message received from process specified in event.	OPEN U_255,255 P_BATCHCOM \MELBDEV.$:0:31:20665784
PURGE	Job input file purged by scheduler.	PURGE IN File: $DATA7.NB.PURGEME J_2
RESTRT	Job restarted.	RESTRT Restart job, due to CPU failure J_7
RLSE	Dependent job released by its master.	JOB B Ok J_8 U_255,255 H_\MELBDEV.$SCHD
SHUTDN	SHUTDOWN SCHEDULER command executed.	SHUTDN SHUTDOWN SCHEDULER \MELBDEV.$:0:188:20678840 U_255,255 H_\MELBDEV.$T4.#A
START	START system message received from process specified in event.	START EXECUTOR-PROGRAM U_255,255 J_1 P_TACL \MELBDEV.$X206:23826617

Table 3-3. Scheduler Log File Keywords (page 3 of 3)

Keyword	Description	Example
STOP	STOP system message received from process specified in event.	STOP CC_6 by \MELBDEV.$SCHD:23826361 U_255,255 J_1 \MELBDEV.$X206:23826617
STOPT	STOP system message text.	STOPT Message text for J_2 \MELBDEV.$X284:25137081
SUSPND	SUSPEND JOB command executed.	SUSPND JOB XYZ J_1 U_255,255 H_\MELBDEV.$T4.#A
SWITCH	SWITCHCPU SCHEDULER command executed.	SWITCH CPU Scheduler process, primary 0 , backup 1 U_255,255 H_\MELBDEV.$T4.#A
UP	CPU reloaded or object now available because of CPU reload.	UP CPU \MELBDEV.3
UPDATE	Scheduler database updated.	UPDATE EXECUTOR E1 S_ACTIVE
WARM	Scheduler warm started.	WARM Load Scheduler database

Logging Formats Used in Event Descriptions

Table 3-4. Scheduler Log File Logging Formats (page 1 of 2)

Logging Format	Description	Example
C_ *class-name: SELPRI*	Supplies the name of a job's class and the value of the job's SELPRI attribute.	C_DEFAULT:5
CC_ *completion-code*	Supplies the completion code issued by a process deletion message.	CC_6 by \MELBDEV.$ZBAT:160442 17
E_ *executor-name*	Supplies the name of an executor.	E_TEST-EXECUTOR
H_ *home-term*	Identifies the home terminal of a request.	H_\MELBDEV.$LANA.#ZWN0140
J_ *job-number*	Identifies a job by number.	J_605
L_ *spooler-job-number*	Supplies the spooler-job number of a job's output file.	L_1358
P_ *program-file-ID*	Supplies the file ID of a job's executor program.	P_TACL
S_ *executor-state*	Indicates the state of an executor.	S_ACTIVE

Table 3-4. Scheduler Log File Logging Formats (page 2 of 2)

Logging Format	Description	Example
T_ *cpu-time*	Supplies the CPU time taken by a job in hours, minutes, seconds, and microseconds.	T_0:0:12:163
U_ *group-ID, user-ID*	Identifies the user making the request.	U_205,70
\ *node.$: CPU: PIN: sequence-number*	Identifies an unnamed process.	\MELBDEV.$:0:165:14312376

Enabling and Disabling EMS Event-Message Generation

The scheduler has an EMS interface through which it can send information about these events to an EMS collector. For more information about the interface and about scheduler event messages, see the *NetBatch Management Programming Manual*.

Event	Event message
Job started	102: ZBAT-EVT-JOB-START
Job failed to start	301: ZBAT-EVT-JOB-START-ERROR
Job exceeded execution time limit	204: ZBAT-EVT-JOB-OVER-LIMIT
Job stopped normally	202: ZBAT-EVT-JOB-NORMAL-STOP
Job stopped abnormally	203: ZBAT-EVT-JOB-ABNORMAL-STOP
Executor's CPU down	200: ZBAT-EVT-EXECUTOR-DOWN
Executor's CPU up	201: ZBAT-EVT-EXECUTOR-UP
Scheduler started	100: ZBAT-EVT-SCHEDULER-START
Scheduler stopped	101: ZBAT-EVT-SCHEDULER-STOP
Scheduler abended	500: ZBAT-EVT-SCHEDULER-ABENDED
Logon Failure	501 ZBAT-EVT-LOGON-FAILURE
LogFile Creation Error	502 ZBAT-EVT-LOGFILE-CREATE-ERROR

You can enable NetBatch event-message generation when starting a scheduler or while the scheduler is running:

- To enable event-message generation when starting a scheduler, include the EMS parameter in the RUN NETBATCH command. For example:

```
> NETBATCH /NAME $ZBAT, NOWAIT/ $NB.ZBAT EMS
> BATCHCOM $ZBAT; INFO SCHEDULER, EMS
SCHEDULER ATTRIBUTES
ems: On
```

- To enable event-message generation while the scheduler is running, use the ALTER SCHEDULER command to set the scheduler's EMS attribute to ON or ERROR.

To disable event-message generation, use the ALTER SCHEDULER command to set the EMS attribute to OFF. For more information, see EMS Scheduler Attribute on page 7-54.

Stopping a Scheduler

NetBatch supervisors can stop (shut down) a scheduler by using either of these BATCHCOM commands:

Command	Function
ABORT SCHEDULER	Stops all executing and suspended processes associated with jobs, then stops the scheduler.
SHUTDOWN SCHEDULER	Stops suspended processes associated with jobs, then stops the scheduler after allowing all executing processes associated with jobs to finish.

This table describes when to use ABORT SCHEDULER and SHUTDOWN SCHEDULER:

If you want to …	Then use …
Immediately stop a scheduler and all processes it controls	ABORT SCHEDULER
Stop a scheduler and all processes it controls, but allow processes that are executing to finish before shutdown takes effect	SHUTDOWN SCHEDULER

Example

To use the SHUTDOWN SCHEDULER command to stop a scheduler:

```
> PPD $SCHD
Name Primary Backup Ancestor
$SCHD 0,466 3,270 $C4
> BATCHCOM $SCHD; SHUTDOWN SCHEDULER
Scheduler shutting down
> PPD $SCHD
(Process does not exist)
```

Using TACL Commands to Stop a Scheduler

The TACL STATUS and STOP commands let you stop processes in a TACL environment. Although these commands also stop a scheduler, HP recommends against using them. The commands act on the scheduler only and have no effect on the processes it controls. After the scheduler stops, the processes remain in an uncontrolled state.

Recovering From an Unplanned Scheduler Stoppage

An unplanned scheduler stoppage can occur, for example, when someone inadvertently stops the scheduler with a TACL STOP command. To recover from such a stoppage:

1. Establish the cause of the shutdown. Use the scheduler's last log file to help.
2. Correct the cause of the shutdown as necessary.
3. Warm start the scheduler. See Warm Starting a Scheduler on page 3-18.

4

Job Planning, Submission, and Management

This section describes how to work with NetBatch jobs:

Topic	Page
Planning Jobs	4-1
Submitting Jobs	4-13
Managing Jobs	4-42

Planning Jobs

This subsection outlines a job planning procedure that lets your organization take advantage of NetBatch job management facilities. The procedure is based on the assumption that programs comprising the jobs:

- Generate Event Management Service (EMS) event messages when significant errors and warnings occur during processing. The programs should send these messages to the same EMS collector as the NetBatch scheduler.
- Use completion codes to indicate process termination status. This lets you use the NetBatch job-recovery attributes IFFAILS, RESTART, STALL, and STOP-ON-ABEND to manage job recovery and restart functions.
- Follow a standard recovery strategy and enable automated or partially automated recovery. Automated or partially automated recovery is likely to involve use of NonStop Transaction Management Facility (TMF) software.

The procedure covers such topics as identifying programs and run environments, establishing recovery strategies, and determining job attributes. The subsection also discusses input-file creation, describes the use of completion codes in job management, and explains how the scheduler tracks and controls jobs' processes.

Planning Procedure

Step 1: Identify and Categorize Existing and Potential Jobs

Create a table that categorizes existing and potential jobs by type or business function (for example, backups or payroll processing). For each job category:

- Record job ownership details and list the programs used
- Indicate whether the jobs are CPU-bound or I/O-bound
- Itemize their resource requirements (disks, tape drives, printers, and so on)

Table 4-1. Sample Job Category Table

Job Category	Ownership	Programs	CPU or I/O-Bound	Resources
Backups	Operations	BACKUP, FUP	I/O	Disk, tape drive
Program compilations	Development	C, COBOL, TAL	CPU	Disk, spooler
Payroll processing	Paymaster, Accounts	Payroll programs	I/O	Disk, spooler, printer, tape drive
End-of-period reporting	Sales, Accounts, Administration	Enform	I/O	Spooler, printer
Transaction log processing	EFT Services	Transaction log programs	I/O	Disk, spooler, printer, tape drive

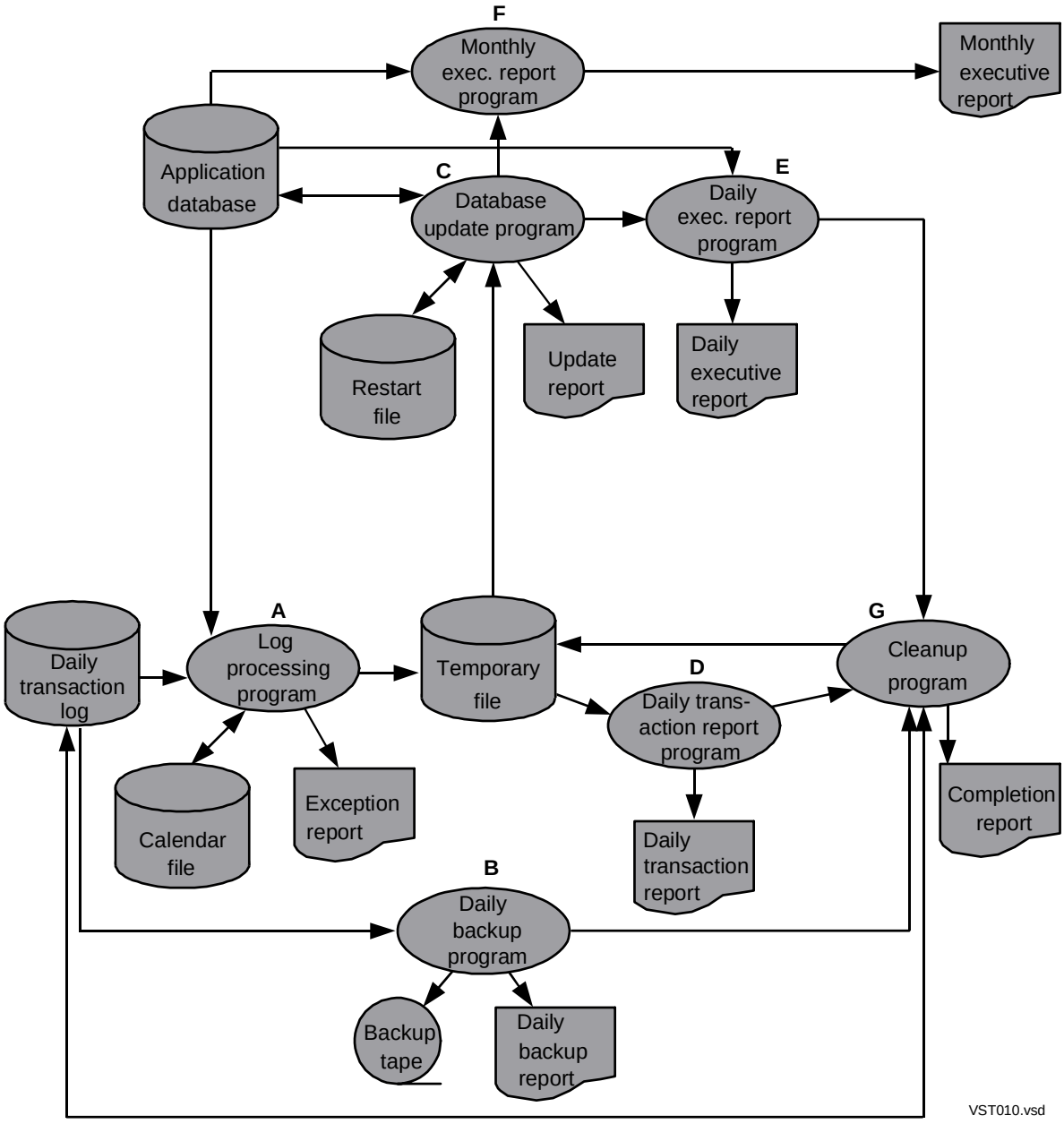
Step 2: Identify Programs and Run Environments

For each job category identified in Step 1, draw a diagram showing the interrelationships of programs in the category and the programs' run environments. For the latter, include:

- Details of inputs (for example, files or tapes)
- Details of outputs (for example, reports and operator messages)
- File creation and file open attributes
- Information about tape processing, log messages, completion codes, and program duration
- Details of TMF transactions

Figure 4-1 on page 4-3 shows the programs and run environments identified for a sample job category (transaction log processing) from the previous figure.

Figure 4-1. Sample Programs and Run Environments Diagram



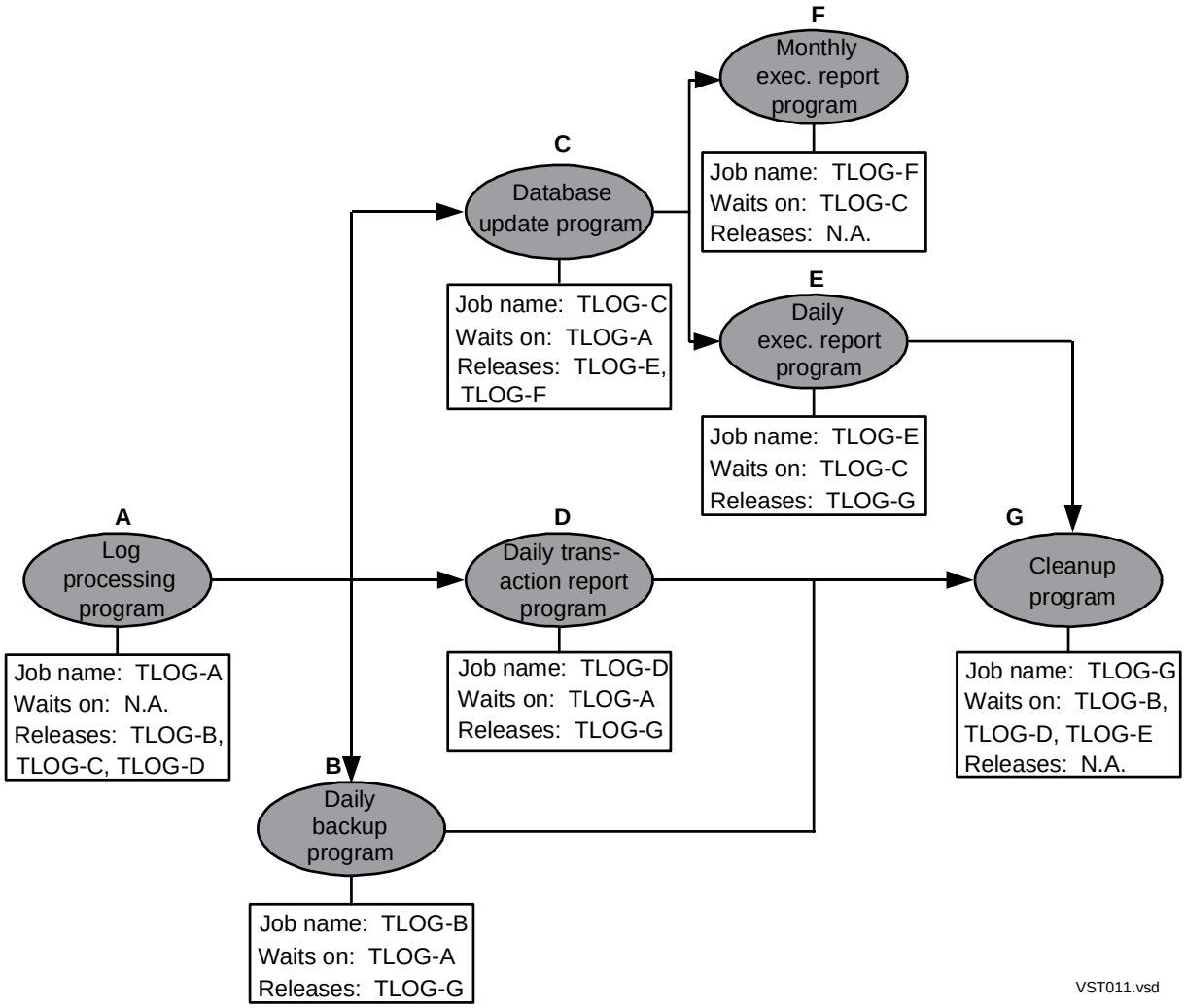
Step 3: Identify Jobs and Dependencies

For each program identified in Step 2:

1. Assign a job name
2. Draw a dependency diagram showing the dependencies between the jobs. In some cases, a job might include several programs.

Figure 4-2 shows the job names assigned to the programs from the previous figure and the dependencies between those programs.

Figure 4-2. Sample Job Dependencies Diagram



Step 4: Establish Recovery Strategies

Establish a job-recovery strategy for use when a job fails. Consider such things as:

- Each job's importance
- The degree of automation required in the recovery process
- Each job's ability to restart itself
- How long each job takes to run

The recovery strategy is implemented when writing the jobs' programs, creating their input files, and assigning their IFFAILS, RESTART, STALL, and STOP-ON-ABEND

attributes. (For a decision chart to help you determine the values of these attributes, see Controlling a Job's Behavior on Process Failure on page 4-28.)

For an example of a job-recovery strategy, consider the transaction log processing example from the previous steps. The recovery strategy for this application is to reduce operator intervention by partially automating job recovery and restart functions. Implementation of the strategy is by means of TMF software (for transaction recovery) and the IFFAILS, RESTART, STALL, and STOP-ON-ABEND attributes. Table 4-2 records information applicable to this strategy for jobs that make up the application.

Table 4-2. Sample Job-Recovery Strategies

Job	TMF Protected?	Restartable?	Restart Procedure?	IF-FAILS	RE-START	Stall	Stop_ ON-ABEND
TLOG-A	Yes-one TMF transaction	Yes-from beginning job	Manual restart from SPECIAL-9 (STALL) state. Automatic restart if job stops with completion code 7 or because of CPU failure.	ON	ON	ON	ON
TLOG-B	No	As for TLOG-A	As for TLOG-A	ON	ON	ON	ON
TLOG-C	Yes-one TMF transaction for each 500 records read from temporary file.	Yes-by means of program restart logic.	As for TLOG-A	ON	ON	ON	ON
TLOG-D	No	As for TLOG-A	As for TLOG-A	ON	ON	ON	ON
TLOG-E	No	As for TLOG-A	As for TLOG-A	ON	ON	ON	ON
TLOG-F	No	As for TLOG-A	As for TLOG-A	ON	ON	ON	ON
TLOG-G	No	As for TLOG-A	As for TLOG-A	ON	ON	ON	ON

Step 5: Create Job Planning Table

Create a table that lists the jobs identified at Step 3 and specifies their attributes. For information about job attributes, see Submitting a Job With the SUBMIT JOB Command on page 4-13.

The table can help you when planning your organization's schedulers (see Section 3, Scheduler Planning, Configuration, and Management) and submitting the jobs.

In Figure 4-3, Job TLOG-C submits job TLOG-F on a monthly basis by using TACL code to detect the end of the month.

Figure 4-3. Sample Jobs Planning Table

	Attributes						
Job	**IN**	**EVERY**	**AFTER**	**WAITON**	**TAPEDRIVES**	**IFFAILS**	**RESTART**
TLOG-A	$DATA7.OPS.TLOGA	1 DAY	18:00	N.A.	0	ON	ON
TLOG-B	$DATA7.OPS.TLOGB	1 DAY	18:00	TLOG-A	1	ON	ON
TLOG-C	$DATA7.OPS.TLOGC	1 DAY	18:00	TLOG-A	0	ON	ON
TLOG-D	$NB.RPRTS.TLOGD	1 DAY	18:00	TLOG-A	0	ON	ON
TLOG-E	$NB.RPRTS.TLOGE	1 DAY	18:00	TLOG-C	0	ON	ON
TLOG-F	$NB.RPRTS.TLOGF	Submitted monthly by TLOG-C		TLOG-C	0	OFF	ON
TLOG-G	$BIG1.OPS.TLOGG	1 DAY	18:00	TLOG-B, TLOG-D, TLOG-E	0	ON	ON

VST117.vsd

In addition to the attributes listed in Figure 4-3, the jobs adopt these default attributes from their scheduler:

Attribute	Value
CLASS	DEFAULT
EXECUTOR-PROGRAM	$SYSTEM.SYSTEM.TACL
HIGHPIN	ON
MAXPRINTLINES	NONE
MAXPRINTPAGES	NONE
OUT	$S.#DEFAULT
PRI	120
SELPRI	3
STALL	ON
STOP-ON-ABEND	ON

Creating a Job Input File

A job's input file contains the statements executed by the job's executor program. (For example, if a TACL process is the executor program, the file contains TACL commands; if the COBOL compiler is the executor program, the file contains the program source code.) The syntax of the statements in the file must follow the syntax rules of the executor program.

Figure 4-4 shows sample TACL input files for two of the jobs (TLOG-A and TLOG-B) planned earlier.

Figure 4-4. Sample Input File

TACL input file for job TLOG ?A:

```
#SET #INFORMAT TACL
RUN A
[#IF (:_COMPLETION:MESSAGECODE    = -5 AND
     :_COMPLETION:COMPLETIONCODE = 7)
COMMENT Stop myself if A stops with completion code 7
   |THEN|
     #STOP /COMPLETIONCODE 7/ [#PROCESSINFO /PROCESSID/]
COMMENT Release A's dependent jobs if A runs successfully
   |ELSE|
     ZBAT:RELEASE $ZBAT *
]
```

TACL input file for job TLOG ?B:

```
BACKUP $TAPE1, $DATA1.DAYLOGS.TODAYS, LISTALL, OPEN, &
AUDITED, VERIFYREEL
ZBAT:RELEASE $ZBAT TLOG-G
```

VST118.vsd

Using Completion Codes to Test Process Termination Status

Every process that executes returns a completion code to its ancestor in its ABEND, STOP, or PROCESS_STOP_ system message. The code indicates whether the process terminated normally or otherwise.

Completion codes are integers in the range -32768 to 32767. These considerations apply to their use:

Completion Codes	Comments
-32768 to -1	Reserved for HP use. Only a privileged caller can have a negative completion code returned to its ancestor.
0 to 999	Reserved for shared use by HP and its customers.
1000 to 32767	Reserved for customer use. Not used by HP.

Table 4-3 lists and describes the standard completion codes used by NonStop system language compilers and other HP products. For more information about the codes, see the *Guardian Procedure Calls Reference Manual*.

Table 4-3. Completion Codes

Code	Description
-3	The process terminated itself but passed bad parameters to the ABEND, STOP, or PROCESS_DELETE_ system procedure.
-2	The process terminated itself, but the operating system could not pass completion code and termination information because of a resource problem.
-1	Trap detected.
0	Normal, voluntary termination with no errors.
1	Normal, voluntary termination with warning diagnostics.
2	Abnormal, voluntary termination with fatal errors or diagnostics.
3	Abnormal, voluntary, but premature termination with fatal errors or diagnostics.
4	The process did not start.
5	The process called the ABEND or PROCESS_STOP_ system procedure on itself.
6	An external, authorized process called the ABEND, STOP, or PROCESS_STOP_ system procedure to delete the process.
7	The process sent a restart request to the NetBatch scheduler.
8	Same as for completion code 1, but you must examine the process's output file to determine whether the results are acceptable.

Testing Completion Codes

The TACL #CASE and #IF built-in functions enable you to determine why a process of a TACL job stopped. By including the functions in the job's input file, you can test completion codes and either return text or branch to other code. For more information on the functions, see the *TACL Reference Manual*. For information on how the TACL program handles completion codes, see the *TACL Programming Guide*.

Figure 4-5 on page 4-9 shows sample #CASE and #IF constructs that test completion codes.

Figure 4-5. Sample #CASE and #IF Constructs for Completion Code Testing

```
[#CASE [:_COMPLETION:COMPLETIONCODE]
   |0| #OUTPUT  text
   |1| #OUTPUT  text
   |2| #OUTPUT  text
    ⋮

   |OTHERWISE| #OUTPUT   text
]

[#IF [#COMPUTE :_COMPLETION:COMPLETIONCODE > 0]
  |THEN|  non-zero-branch
   |ELSE|  zero-branch
]                                                    VST119.vsd
```

Figure 4-6 shows the input file of a job that compiles a program. The file includes #CASE statements that test the completion code of the compilation. If the compilation is successful, the job runs the resulting object.

Figure 4-6. Sample Job Input File Containing #CASE Statements

```
#SET #INFORMAT TACL
#SET #OUTFORMAT TACL
#PUSH SOURCE_FILE
#PUSH OBJECT_FILE
#SET SOURCE_FILE $NB.FILES.SRCFILE
#SET OBJECT_FILE $NB.FILES.OBJFILE
COBOL85 /IN [SOURCE_FILE], OUT $S.#LIST / [OBJECT_FILE]   [#CASE
[:_COMPLETION:COMPLETIONCODE]
  |0| #OUTPUT No errors or warnings compiling [OBJECT_FILE]
      RUN [OBJECT_FILE]
        #OUTPUT [OBJECT_FILE] completion:
        OUTVAR :_COMPLETION
   |1| #OUTPUT Warnings compiling [OBJECT_FILE], completion:
      OUTVAR :_COMPLETION
      RUN [OBJECT_FILE]
        #OUTPUT [OBJECT_FILE] completion:
        OUTVAR :_COMPLETION
   |2| #OUTPUT Fatal compilation errors
   |3| #OUTPUT Compilation failed
   |4| #OUTPUT Compilation never started
   |5| #OUTPUT Compilation abended
   |6| #OUTPUT External stop
   |7| #OUTPUT Compilation restarted
   |8| #OUTPUT Check your listing
   |OTHERWISE| #OUTPUT Unexpected completion:
            OUTVAR :_COMPLETION
]                                                    VST120.vsd
```

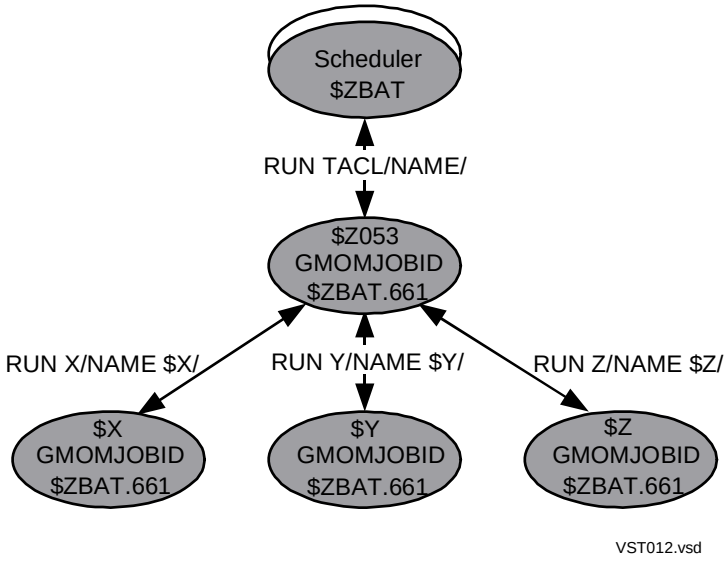
Identifying Processes of a Job

When a scheduler starts a job with the attribute JOBID-ZERO OFF:

1. It tags the job's executor-program process with a unique identifier called a GMOMJOBID. This identifier contains the scheduler's process name (that is, the name of the job's ancestor process, or GMOM) and the job's number (JOBID); for example, $ZBAT.278.
2. The executor-program process propagates the GMOMJOBID to its child processes. Thus, the GMOMJOBID identifies processes related to the job. Moreover, it lets the scheduler receive creation and completion information about the processes and to suspend, reactivate, and stop them.

Figure 4-7 shows the scheduler starting job 661 whose executor-program process ($Z053) starts three other processes. $Z053 transmits its GMOMJOBID ($ZBAT.661) to those processes.

Figure 4-7. Example of GMOMJOBID Propagation



Displaying GMOMJOBIDs

To display the GMOMJOBID of a process, use the TACL STATUS command. This example shows the command listing the GMOMJOBIDs of processes from Figure 4-7 (the command's results appear in truncated form):

```
> STATUS *, GMOMJOBID $ZBAT.661, DETAIL
Pid: 1,47 ($X) Primary
GMOMJOBID: $ZBAT.661
Pid: 1,48 ($Z) Primary
GMOMJOBID: $ZBAT.661
Pid: 1,55 ($Z053) Primary
GMOMJOBID: $ZBAT.661
```

```
Pid: 1,78 ($Y) Primary
GMOMJOBID: $ZBAT.661
```

For more information about the STATUS command, see the *TACL Reference Manual*.

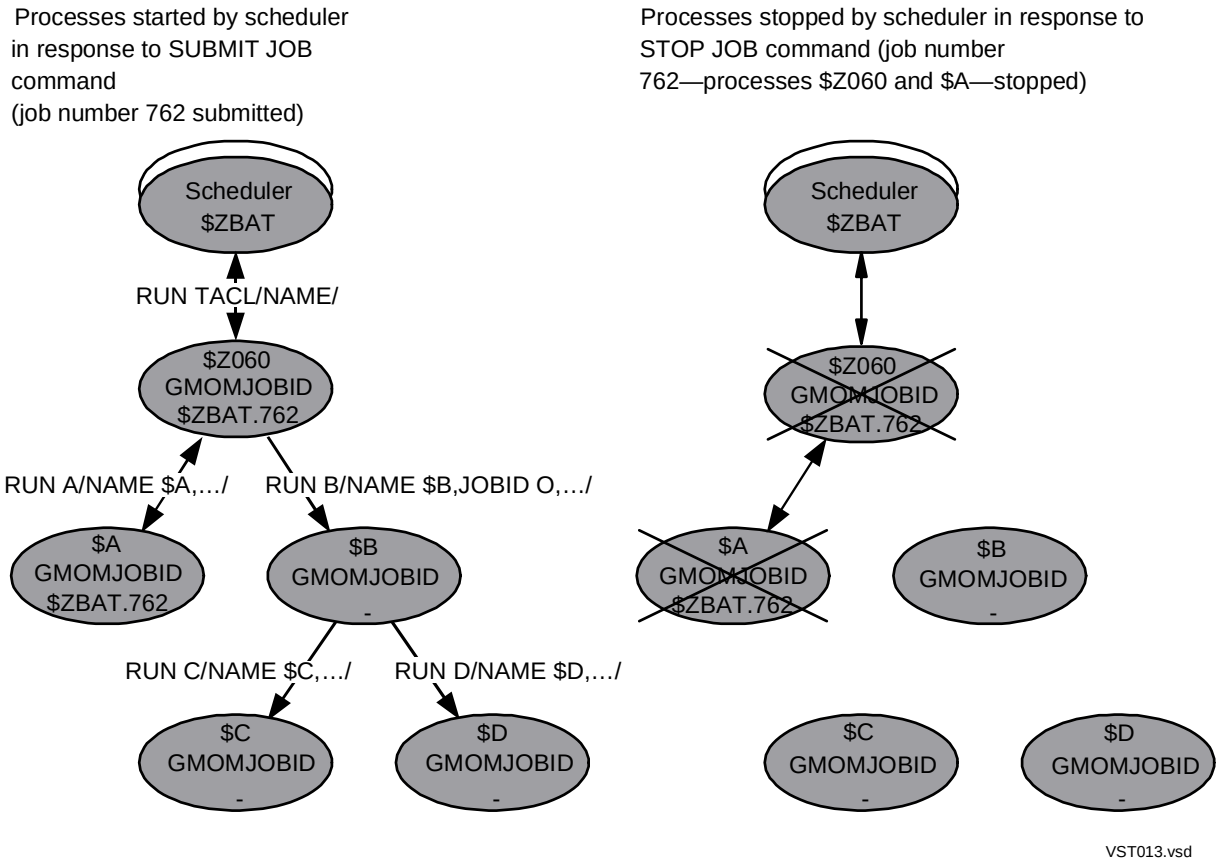
Dissociating a Process From a Job

In some cases, you might want to prevent a process from receiving its creator's GMOMJOBID so the process executes outside scheduler control. For example, you might want a NOWAIT Pathway process dissociated from the job that starts it.

To prevent a process from receiving a GMOMJOBID, specify run option JOBID 0 in the process's TACL RUN command. This option overrides the creator's GMOMJOBID and dissociates the process from the job. As a result, the scheduler does not receive creation and completion information from the process. Also, no control of the process is possible through the scheduler. For example, the STOP JOB command stops and deletes all processes having the job's GMOMJOBID, but not the dissociated process (see Figure 4-8).

For more information about run option JOBID, see the *TACL Reference Manual*.

Figure 4-8. Example of Dissociated Processes



Dissociating a Job From the Scheduler

To prevent the scheduler from assigning a GMOMJOBID to a job's executor-program process, submit the job with the attribute JOBID-ZERO ON. This attribute dissociates the job completely from the scheduler.

Specifying TERM, IN, OUT, and NOWAIT for a Dissociated Process

These recommendations relate to specifying run options TERM, IN, OUT, and NOWAIT in a TACL RUN command that specifies JOBID 0:

- A dissociated process cannot open a scheduler for home terminal, input, or output purposes unless the open is by means of the SPI process name qualifier #ZSPI. (For example, TERM $ZBAT.#ZSPI.) This limitation means the TERM, IN, and OUT run options of a dissociated process cannot specify a scheduler if the process does not send SPI open requests.

  For information on using the #ZSPI qualifier, see the *SPI Programming Manual*.

- A dissociated process should have the NOWAIT run option specified. Running the process with this option lets the job finish before the process and therefore release the job's executor.

For more information on run options TERM, IN, OUT, and NOWAIT, see the *TACL Reference Manual*.

ZBAT:JOBINFO

ZBAT:JOBINFO is a TACL macro supplied with NetBatch software (in product file ZBATSEGF). The macro expands to a space-separated job name and number. By invoking the macro from a job's input file, you can return the job's name and number to its output file. For example:

```
> FUP COPY INFILE
#PUSH JOBNAME, JOBNUMBER
#SETMANY JOBNAME JOBNUMBER, [ZBAT:JOBINFO]
#OUTPUT Job [JOBNAME] (job number [JOBNUMBER]) now running
> BATCHCOM $ZBAT; SUBMIT JOB, EXECUTOR-PROGRAM TACL, IN
INFILE,
OUT [#MYTERM]
Job ZBAT-0003 Jobnumber 3 submitted
> PAUSE
TACL …
.
.
Job ZBAT-0003 (job number 3) now running
```

Submitting Jobs

To submit jobs to a NetBatch scheduler, use:

- The BATCHCOM command SUBMIT JOB
- The NetBatch-Plus application

You also can submit jobs programmatically from user-written programs.

This subsection discusses job submission using the SUBMIT JOB command only:

Function	Page
Submitting a Job With the SUBMIT JOB Command	4-13
Specifying a Job's Class and Selection Priority	4-19
Specifying a Job's Executor Program and the Program's Run Options and Parameters	4-20
Specifying a Job's Input File, Defaults, and PURGE-IN-FILE Attribute	4-21
Specifying a Job's Output File, Maximum Print Lines or Maximum Print Pages, and Log File	4-22
Scheduling a Job	4-23
Specifying a Job's Hold Characteristics	4-26
Controlling a Job's Behavior on Process Failure	4-28
Specifying a Job's Tape Drives Requirements	4-30
Specifying a Job's Dependencies	4-31
Specifying a Job's ASSIGNs, DEFINEs, and PARAMs	4-34
Specifying a Job's Description	4-40

For NetBatch-Plus information, see the *NetBatch-Plus Reference Manual*. For information about programmatic job submission, see the *NetBatch Management Programming Manual*.

Submitting a Job With the SUBMIT JOB Command

The SUBMIT JOB command submits a job to the scheduler. On accepting the job, the scheduler assigns it a unique number in the range 1 through 32767 for systems running G-series RVUs. For systems running H-series RVUs, the range is 1 through 9999. If the job was submitted without a name, the scheduler generates a name of the form ZBAT-*nnnn*, where *nnnn* is the job's number; for example, ZBAT-0016.

The scheduler's SUBMIT-ALLOWED attribute controls use of the SUBMIT JOB command. You can use the command when the attribute is set to ON but not when it is set to OFF. To check the value of the attribute, use the INFO SCHEDULER command.

Examples

- The SUBMIT JOB command in this example submits a job named DAILY-BACKUP. The job's attributes nominate its run time, class, and executor program, its input and output files, and its tape drives requirements. The example also shows the command response, which contains the scheduler-assigned job number.

```
2} SUBMIT JOB DAILY-BACKUP, EVERY 1 DAYS, AFTER 18:00, CLASS
OPERATIONS, EXECUTOR-PROGRAM BACKUP, IN $NB.BACKUPS.BKUPIN,
OUT
$S.#BKUPOUT, TAPEDRIVES 1
Job DAILY-BACKUP Jobnumber 4 submitted
```

- This example shows a scheduler with the SUBMIT-ALLOWED OFF attribute rejecting a submitted job:

```
63} INFO SCHEDULER, SUBMIT-ALLOWED
SCHEDULER ATTRIBUTES
submit-allowed: Off
64} SUBMIT JOB X
2128-E SUBMIT-ALLOWED is OFF; the scheduler is not accepting
job submissions
65} INFO JOB X
2099-E JOB does not exist
```

Specifying a Job Name

When you specify a job name, it must be in the correct form. Otherwise job submission fails. A valid job name contains 1 through 24 letters and numbers. It can contain hyphens but must begin with a letter and end with a letter or number. It cannot contain spaces.

Specifying Job Attributes

You use job attributes to determine when a job runs, its processing environment and dependencies, and its behavior on failure. In particular, job attributes let you perform all the tasks listed on page 4-13.

Table 4-4 on page 4-15 lists the job attributes, describes their functions, and gives their default values. For more information about the attributes, see Section 7, Attributes. For a discussion about attribute defaulting, see SUBMIT JOB Command on page 6-183.

Table 4-4. Job Attributes (page 1 of 4)

Attribute	Function	Default Value
AFTER	Specifies the date and time after which a job becomes eligible for execution. The attribute also lets you change a nonexecuting job's AT attribute to AFTER.	No AFTER attribute.
AT	Specifies the date and time at which the scheduler is to create a temporary executor for and execute a job.	No AT attribute.
ATTACHMENT-SET	Assigns up to three attachment sets to a job or dissociates attachment sets from a job.	No ATTACHMENT-SET attribute.
CALENDAR	Specifies the name of the BATCHCAL file containing a job's run calendar. To remove the attribute from the job, specify CALENDAR with no accompanying calendar file name.	No CALENDAR attribute.
CLASS	Specifies a job's class.	DEFAULT-CLASS scheduler attribute.
DESCRIPTION	Contains text that describes a job.	No DESCRIPTION attribute.
EVERY	Specifies that job execution occur at regular, specified intervals.	No EVERY attribute.
EXECUTOR PROGRAM	Specifies the program file of the program the scheduler starts as the initial process of a job.	DEFAULT-EXECUTOR-PROGRAM scheduler attribute.
EXTSWAP	Specifies the name of the swap file for the default extended data segment of a job's executor-program process.	No EXTSWAP attribute. Extended swap file created on volume specified by SWAP attribute of DEFINE =_DEFAULTS. If DEFINE does not have SWAP attribute, volume is selected by operating system.
HIGHPIN	Determines whether a job's executor-program process runs at a low PIN or at a high PIN. Only available in NetBatch product version D20 or later.	DEFAULT-HIGHPIN scheduler attribute.
HOLD	Determines whether a job is available for execution.	No HOLD attribute.
HOLDAFTER	Determines whether the scheduler puts a job on hold when the job finishes executing.	No HOLDAFTER attribute.

Table 4-4. Job Attributes (page 2 of 4)

Attribute	Function	Default Value
IFFAILS	Determines whether the scheduler reschedules a recurrent job (a job that has the CALENDAR or EVERY attribute) that fails during execution. Works in combination with the job's RESTART, STALL, and STOP-ON-ABEND attributes.	No IFFAILS attribute.
IN	Specifies the name of a job's input file.	No IN attribute. Job uses scheduler as its input file.
JOB-LOG	Directs log-file events for a job to a specified file.	No JOB-LOG attribute.
JOBID-ZERO	Determines whether the scheduler assigns a GMOMJOBID to a job's executor-program process.	OFF.
LIB	Specifies the name of the user library file for a job's executor program.	No LIB attribute. Executor program uses library file specified when program last ran. The program runs without a library file if that is how it ran previously. The current library file applies if program is running.
LIMIT	Specifies an execution time limit for a job.	No LIMIT attribute.
MAXPRINTLINES	Specifies the maximum number of print lines for a job's spooler output file.	DEFAULT-MAXPRINTLINES scheduler attribute.
MAXPRINTPAGES	Specifies the maximum number of print pages for a job's spooler output file.	DEFAULT-MAXPRINTPAGES scheduler attribute.
MEM	Specifies the minimum number of 2048-byte memory pages allotted to a job's executor-program process for user data.	No MEM attribute. Job has memory pages allotted according to memory-pages value specified in executor program. This default also applies when MEM attribute specifies a number less than that value.
NAME	Specifies a name for a job's executor-program process.	No NAME attribute. Executor-program process assigned system-generated name.
OUT	Specifies the output file to which the scheduler writes data produced by an executing job.	DEFAULT-OUT scheduler attribute.

Table 4-4. Job Attributes (page 3 of 4)

Attribute	Function	Default Value
PFS	Specifies the size in bytes of a job's executor-program process file segment (PFS).	No PFS attribute. Executor program has PFS size specified in program.
PRI	Specifies the execution priority of a job's executor-program process.	DEFAULT-PRI scheduler attribute.
PURGE-IN-FILE	Determines whether the scheduler is to purge a job's input file when it deletes the job.	No PURGE-IN-FILE attribute.
RESTART	Determines whether the scheduler restarts a job that stops with completion code 7 or terminates because of CPU failure. Works in combination with the job's IFFAILS, STALL, and STOP-ON-ABEND attributes.	No RESTART attribute.
RUND	Specifies whether a job's executor program enters the Guardian debug facility Debug or the Inspect interactive symbolic debugger when the program runs.	No RUND attribute.
SAVEABEND	Specifies whether a job's executor-program process is to create a save file if the process traps or abends.	No SAVEABEND attribute.
SELPRI	Specifies the selection priority of a job in its class.	DEFAULT-SELPRI scheduler attribute.
STALL	Determines whether the scheduler puts in the SPECIAL-9 state a failed job it would otherwise reschedule or delete. Works in combination with the job's IFFAILS, RESTART, and STOP-ON-ABEND attributes.	DEFAULT-STALL scheduler attribute.
STARTUP	Specifies one or more program parameters the scheduler sends a job's executor-program process in the startup message.	No STARTUP attribute.
STOP-ON-ABEND	Determines whether the scheduler stops a job and all its processes if any process of the job terminates because of CPU failure; abends with any completion code; or stops with completion code -3, -2, -1, 2, 3, 4, 5, or 6.	DEFAULT-STOP-ON-ABEND scheduler attribute.

Table 4-4. Job Attributes (page 4 of 4)

Attribute	Function	Default Value
SWAP	Specifies the name of the swap file for the user data stack segment of a job's executor-program process.	No SWAP attribute. Swap file created on volume specified by SWAP attribute of DEFINE =_DEFAULTS. If DEFINE does not have SWAP attribute, volume selected by operating system.
TAPEDRIVES	Specifies the number of tape drives required by a job.	No TAPEDRIVES attribute.
TERM	Specifies the home terminal of a job's executor-program process.	No TERM attribute. Job uses scheduler as its home terminal.
VOLUME	Specifies the default node, volume, and subvolume for unqualified file references in a job's input file. You also can use the attribute to specify the default security for disk files created by the job.	Node, volume, subvolume, and security defaults set by the last SET JOB VOLUME command. If there was no such command, the default values are the defaults current when the session began.
WAIT	Delays execution of a job for a specified period from the current time on the node where the job's scheduler runs.	No WAIT attribute.
WAITON	Specifies the names of up to 50 jobs on which execution of another job depends. To remove the attribute from the job, specify WAITON with no options.	No WAITON attribute.

Displaying Job Attributes

The INFO JOB command displays all attributes of a job or jobs, or specified attributes only. For example, this command displays all attributes of job COBOL-COMPILE:

```
19} INFO JOB COBOL-COMPILE
JOB ATTRIBUTES for COBOL-COMPILE
jobnumber: 6
volume: \DEV.$BIG1.TEMP, "NCNC"
in: \DEV.$DATA6.DEV.MYPROG
out: \DEV.$S.#COBOL
executor-program: \DEV.$SYSTEM.SYSTEM.COBOL85
pri: 119
selpri: 1
maxprintlines: None
maxprintpages: 20
class: COMPILES
stall: Off
stop-on-abend: On
after: 20OCT94 23:30:00
```

```
highpin: On
submit: 20OCT94 15:51:26
alter: 20OCT94 15:51:27
user: 255,205
next-runtime: 20OCT94 23:30:00
```

This command displays only specified attributes of job COBOL-COMPILE:

```
20} INFO JOB COBOL-COMPILE, IN, OUT, EXECUTOR-PROGRAM, PRI,
SELPRI, CLASS
JOB ATTRIBUTES for COBOL-COMPILE
jobnumber: 6
in: \DEV.$DATA6.DEV.MYPROG
out: \DEV.$S.#COBOL
executor-program: \DEV.$SYSTEM.SYSTEM.COBOL85
pri: 119
selpri: 1
class: COMPILES
```

Specifying a Job's Class and Selection Priority

The CLASS and SELPRI attributes assign a job to a class and specify the job's selection priority in that class. The class links the job to an executor and hence to the executor's CPU. This link enables the scheduler to start, in the specified CPU, the job's executor-program process.

Example

This example shows submission of three jobs assigned to class DEFAULT. The class's INITIATION attribute before job submission is set to OFF, which prevents the scheduler from selecting the jobs. The INITIATION attribute is set to ON after job submission, which enables the scheduler to select and run the jobs. The jobs' SELPRI attributes determine their execution order.

```
79} ALTER CLASS DEFAULT, INITIATION OFF
Class DEFAULT altered
80} SUBMIT JOB P, CLASS DEFAULT, SELPRI 1
Job P job number 482 submitted
81} SUBMIT JOB Q, LIKE P, SELPRI 4
Job Q job number 483 submitted
82} SUBMIT JOB R, LIKE P, SELPRI 7
Job R job number 484 submitted
83} STATUS JOB *, SELPRI
JOB  JOBNAME          USERID  SEL  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
482   P               255,255  1   READY       DEFAULT
483   Q               255,255  4   READY       DEFAULT
484   R               255,255  7   READY       DEFAULT
84} ALTER CLASS DEFAULT, INITIATION ON
Class DEFAULT altered
85} STATUS JOB *, SELPRI
JOB  JOBNAME          USERID  SEL  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
482   P               255,255  1   READY       DEFAULT
```

```
483   Q               255,255  4   READY       DEFAULT
484   R               255,255  7   EXECUTING   DEFAULT
86} STATUS JOB *, SELPRI
JOB  JOBNAME           USERID  SEL  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
482   P               255,255 1    READY       DEFAULT
483   Q               255,255 4    EXECUTING   DEFAULT
87} STATUS JOB *, SELPRI
JOB  JOBNAME          USERID  SEL  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
482   P               255,255 1    EXECUTING   DEFAULT
88} STATUS JOB *, SELPRI
2117-I No JOB selected
```

Specifying a Job's Executor Program and the Program's Run Options and Parameters

These attributes specify a job's executor program and the program's run options and parameters. The executor program executes the statements in the job's input file. The run options and parameters specify items such as the program's execution priority and startup message.

Example

This example shows the EXECUTOR-PROGRAM, STARTUP, PRI, and HIGHPIN attributes of two jobs. The jobs are the same except for their HIGHPIN attributes. Job D has the attribute HIGHPIN ON, which makes the job's executor-program process run at a high PIN (276). Job E has the attribute HIGHPIN OFF, which makes the job's executor-program process run at a low PIN (42).

```
> BATCHCOM $SCHD; SUBMIT JOB D, IN INFILE,
EXECUTOR-PROGRAM $SYSTEM.SYS00.TACLHI,
STARTUP ";SEGVOL $NB", PRI 180, HIGHPIN ON
Job D job number 7 submitted
> BATCHCOM $SCHD; SUBMIT JOB E, LIKE D, HIGHPIN OFF
Job E job number 8 submitted
> BATCHCOM $SCHD; INFO JOB *, EXECUTOR-PROGRAM, STARTUP,
PRI, HIGHPIN
JOB ATTRIBUTES for D
jobnumber: 7
executor-program: $SYSTEM.SYS00.TACLHI
startup: ;SEGVOL $NB
pri: 180
highpin: On
JOB ATTRIBUTES for E
jobnumber: 8
executor-program: $SYSTEM.SYS00.TACLHI
startup: ;SEGVOL $NB
pri: 180
highpin: Off
> STATUS *, GMOMJOBID $SCHD.7
Process Pri … Program file Hometerm
$Z0007 0,276 180 … $SYSTEM.SYS00.TACLHI $SCHD
```

```
> STATUS *, GMOMJOBID $SCHD.8
Process Pri … Program file Hometerm
$Z227 0,42 180 … $SYSTEM.SYS00.TACLHI $SCHD
Attributes
Executor program EXECUTOR-PROGRAM
Run options EXTSWAP, HIGHPIN, JOBID-ZERO, LIB, MEM, NAME,
PFS,
PRI, RUND, SAVEABEND, SWAP, and TERM
Run parameters STARTUP
Job Planning, Submission, and Management
```

Specifying a Job's Input File, Defaults, and PURGE-IN-FILE Attribute

The IN and VOLUME attributes specify a job's input file, and volume and security defaults. (The input file contains the statements executed by the job's executor program.) The PURGE-IN-FILE attribute determines whether the scheduler purges the input file on job deletion.

Example

This example shows the effect on a job of the IN, VOLUME, and PURGE-IN-FILE attributes. The IN attribute specifies the job's input file, which contains a FUP CREATE command executed by the job's executor program. The VOLUME attribute specifies the volume, subvolume, and security defaults for the file created by the command. The PURGE-IN-FILE attribute makes the scheduler purge the input file after deleting the job.

```
> FILEINFO $DATA7.TEMP.*
$DATA7.TEMP
Code …
INFILE 101 …
> FUP COPY $DATA7.TEMP.INFILE
CREATE X
> BATCHCOM $ZBAT; SUBMIT JOB K, EXECUTOR-PROGRAM FUP,
IN $DATA7.TEMP.INFILE, VOLUME $DATA3.NEWFILES, "AAAA",
PURGE-IN-FILE ON
Job K job number 492 submitted
> BATCHCOM $ZBAT; INFO JOB 492, IN, VOLUME, PURGE-IN-FILE
JOB ATTRIBUTES for K
jobnumber: 492
volume: $DATA3.NEWFILES, "AAAA"
in: $DATA7.TEMP.INFILE
purge-in-file: On
> FILEINFO $DATA3.NEWFILES.X
$DATA3.NEWFILES
Code … RWEP …
X 0 … "AAAA" …
> FILEINFO $DATA7.TEMP.*
No files match \MELBQAT.$DATA7.TEMP.*
```

Specifying a Job's Output File, Maximum Print Lines or Maximum Print Pages, and Log File

The OUT attribute specifies a job's output file, which collects data the job produces. If the file is a spooler collector process, the MAXPRINTLINES and MAXPRINTPAGES attributes specify the spooler job's maximum print lines or maximum print pages. The JOB-LOG job attribute directs log file events for a job to a specified file. For more information about job output, see Dealing With Job Output on page 4-47.

Example

This example shows submission and execution of two jobs whose OUT attributes specify spooler locations. The jobs' MAXPRINTLINES and MAXPRINTPAGES attributes specify the maximum lines and maximum pages for their spooler output.

```
> BATCHCOM $ZBAT; SUBMIT JOB Y, OUT $S.#MAXLINE,
MAXPRINTLINES 600, MAXPRINTPAGES NONE
Job Y job number 695 submitted
> BATCHCOM $ZBAT; SUBMIT JOB Z, OUT $S.#MAXPAGE,
MAXPRINTLINES NONE, MAXPRINTPAGES 10
Job Z job number 696 submitted
> BATCHCOM $ZBAT; INFO JOB *, OUT, MAXPRINTLINES,
MAXPRINTPAGES
OB ATTRIBUTES for Y
jobnumber: 695
out: $S.#MAXLINE
maxprintlines: 600
maxprintpages: None
JOB ATTRIBUTES for Z
jobnumber: 696
out: $S.#MAXPAGE
maxprintlines: None
maxprintpages: 10
> BATCHCOM $ZBAT; STATUS JOB *
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
695  Y                205,70  13   EXECUTING   DEFAULT
696  Z                205,70  15   EXECUTING   DEFAULT
34> SPOOLCOM
)JOB 13, STATUS DETAIL
JOB: 13
.
LOCATION: #MAXLINE
.
MAXIMUM LINES: 600
MAXIMUM PAGES:
)JOB 15, STATUS DETAIL
JOB: 15
.
LOCATION: #MAXPAGE
.
MAXIMUM LINES:
MAXIMUM PAGES: 10
```

Scheduling a Job

The attributes AFTER, AT, CALENDAR, EVERY, and WAIT are timing attributes that specify when a job runs and its run frequency. (AFTER, AT, and WAIT are mutually exclusive, as are CALENDAR and EVERY.) The LIMIT attribute specifies an execution time limit for the job.

A job with the CALENDAR or EVERY attribute is a recurrent job because the scheduler automatically reschedules the job after it runs. A job whose attributes do not include CALENDAR or EVERY is a nonrecurrent job.

AFTER Attribute

The AFTER attribute specifies a time after which a job becomes eligible for execution.

Example

The AFTER attribute in this example schedules a job to run after 11:45 a.m. on May 01, 2002. The scheduler puts the job in the TIME state pending execution.

```
57> TIME
April 2, 2002 14:18:35
58> BATCHCOM $ZBAT; SUBMIT JOB F, AFTER MAY 01 2002 11:45
Job F job number 702 submitted
59> BATCHCOM $ZBAT; INFO JOB F, AFTER
JOB ATTRIBUTES for F
jobnumber: 702
after: 01MAY2002 11:45:00
next-runtime: 01MAY2002 11:45:00
60> BATCHCOM $ZBAT; STATUS JOB F
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
702   F                205,70   01MAY2002         DEFAULT
```

AT Attribute

The AT attribute specifies a time at which the scheduler runs a job. The attribute makes the scheduler create a temporary executor for the job. The scheduler deletes the temporary executor when the job finishes.

The scheduler's AT-ALLOWED attribute determines whether users who are not NetBatch supervisors (that is, users without execute access to the NETBATCH program file) can submit jobs with the AT attribute. To check the value of AT-ALLOWED, use the INFO SCHEDULER command.

Examples

- This example shows a scheduler with the AT-ALLOWED OFF attribute rejecting a job submitted with the AT attribute:

  ```
  3} INFO SCHEDULER, AT-ALLOWED
  SCHEDULER ATTRIBUTES
  ```

```
at-allowed: Off
4} SUBMIT JOB M, AT 19:30
2056-E AT-ALLOWED is currently OFF; submit AFTER time
5} INFO JOB M
2099-E JOB does not exist
```

- This example shows submission of a job with the AT attribute. The scheduler creates a temporary executor for the job and deletes the executor when the job finishes:

```
17} STATUS EXECUTOR *
2117-I No EXECUTOR selected
18} SUBMIT JOB, EXECUTOR-PROGRAM DELAY, STARTUP "1 MINS", AT
Job ZBAT-0009 Jobnumber 9 submitted
19} INFO JOB 9, AT
JOB ATTRIBUTES for ZBAT-0009
jobnumber: 9
at:
next-runtime: 20OCT2002 17:06:27
20} STATUS JOB 9
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
9    ZBAT-0009        205,70  148  EXECUTING   DEFAULT
21} STATUS EXECUTOR *
EXECUTOR CPU STATE JOB CLASS
------------------------ --- ------ ---- -------
__TEMP_EXEC_9 0 DELETE 9 DEFAULT
22} STATUS JOB 9; STATUS EXECUTOR *
2099-E JOB does not exist
2117-I No EXECUTOR selected
```

CALENDAR Attribute

The CALENDAR attribute assigns a BATCHCAL-generated run calendar to a job. The calendar is a file containing a series of run times at which the scheduler automatically runs the job. For information about BATCHCAL and run calendars, see Section 5, Run Calendar Generation and Display.

Example

This example shows generation of a simple run calendar and the submission of a job using that calendar. The scheduler schedules the job to run at the next time in the calendar relative to the current time.

```
15> BATCHCAL
1} NOVDATES == File ID of run calendar
2} 09:00 == Default job-start time
3} 2001 NOV * == Run daily through 11/2001 at 9 a.m.
4} EXIT
16> TIME
October 20, 2001 17:20:13
17> BATCHCAL /IN NOVDATES/ NEXT-DATE
2001-11-01 09:00:00
18> BATCHCOM $ZBAT; SUBMIT JOB, CALENDAR NOVDATES
```

```
Job ZBAT-0013 Jobnumber 13 submitted
19> BATCHCOM $ZBAT; INFO JOB 13, CALENDAR
JOB ATTRIBUTES for ZBAT-0013
jobnumber: 13
calendar: \MELBDEV.$NB.MAN14.NOVDATES
next-runtime: 01NOV2001 09:00:00
```

EVERY Attribute

The EVERY attribute specifies a regular interval at which the scheduler automatically runs a job.

Example

This example shows the EVERY attribute specifying an eight-hour execution frequency for a job. After running the job once when it is submitted, the scheduler automatically reschedules it to run again eight hours later.

```
> TIME
April 2, 1993 15:56:38
> BATCHCOM $ZBAT; SUBMIT JOB H, EVERY 8:00 HOURS
Job H job number 714 submitted
> BATCHCOM $ZBAT; INFO JOB H, EVERY
JOB ATTRIBUTES for H
jobnumber: 714
every: 8:00 HOURS
next-runtime: 02APR93 15:57:02
> BATCHCOM $ZBAT; STATUS JOB H
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
714  H                255,205      EXECUTING   DEFAULT
> BATCHCOM $ZBAT; STATUS JOB H
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
714  H                255,205 23:57:02         DEFAULT
> BATCHCOM $ZBAT; INFO JOB H, EVERY
JOB ATTRIBUTES for H
jobnumber: 714
every: 8:00 HOURS
next-runtime: 02APR93 23:57:02
```

WAIT Attribute

The WAIT attribute prevents the scheduler from executing a job for a specified period after submission.

Example

This example shows the WAIT attribute delaying job execution for 30 minutes from the time of submission:

```
> TIME
April 2, 1993 16:06:50
```

```
> BATCHCOM $ZBAT; SUBMIT JOB I, WAIT 0:30
Job I job number 715 submitted
> BATCHCOM $ZBAT; INFO JOB I, WAIT; STATUS JOB I
JOB ATTRIBUTES for I
jobnumber: 715
after: 02APR93 16:37:20
next-runtime: 02APR93 16:37:20
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
715  I                  255,205 16:37:20          DEFAULT
```

LIMIT Attribute

The LIMIT attribute specifies an execution time limit for a job. When the limit is reached, the job continues to execute, but its state changes from EXECUTING to OVER LIMIT.

Example

This example shows a job with the LIMIT attribute exceeding the specified execution time:

```
9} SUBMIT JOB, EXECUTOR-PROGRAM DELAY, STARTUP "3 MINS",
LIMIT
0:01
Job ZBAT-0016 Jobnumber 16 submitted
10} STATUS JOB 16
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
16   ZBAT-0016        205,70  149  EXECUTING   DEFAULT
11} STATUS JOB 16
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
16   ZBAT-0016         205,70   149  OVER LIMIT   DEFAULT
```

Specifying a Job's Hold Characteristics

The HOLD and HOLDAFTER attributes determine whether the scheduler puts a job on hold before or after the job runs. A job on hold has a state of SPECIAL-1. You can take it off hold by using the ALTER JOB command.

Example

This example shows the effect of the HOLD and HOLDAFTER attributes. The HOLD ON attribute prevents job C from executing when submitted. The ALTER JOB command sets the HOLD attribute to OFF, thus enabling the job to run. The HOLDAFTER ON attribute makes the scheduler put the job back on hold after job execution (that is, the HOLD attribute is reset to ON).

```
10} SUBMIT JOB C, HOLD ON, HOLDAFTER ON
Job C job number 469 submitted
11} STATUS JOB C
```

```
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
469 C 255,255 1:Hold DEFAULT
12} INFO JOB C, HOLD, HOLDAFTER
JOB ATTRIBUTES for C
jobnumber: 469
hold: On
holdafter: On
13} ALTER JOB C, HOLD OFF
Job C job number 469 altered
14} STATUS JOB C
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
469  C                255,255 212  EXECUTING   DEFAULT
15} INFO JOB C, HOLD, HOLDAFTER
JOB ATTRIBUTES for C
jobnumber: 469
hold: Off
holdafter: On
16} STATUS JOB C
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
469  C                255,255 212  1:Hold      DEFAULT
17} INFO JOB C, HOLD, HOLDAFTER
JOB ATTRIBUTES for C
jobnumber: 469
hold: On
holdafter: On
```

Controlling a Job's Behavior on Process Failure

The IFFAILS, RESTART, STALL, and STOP-ON-ABEND attributes determine how the scheduler treats a job when a process of the job fails. Table 4-5 shows the effect on a job's state of all combinations of the attributes.

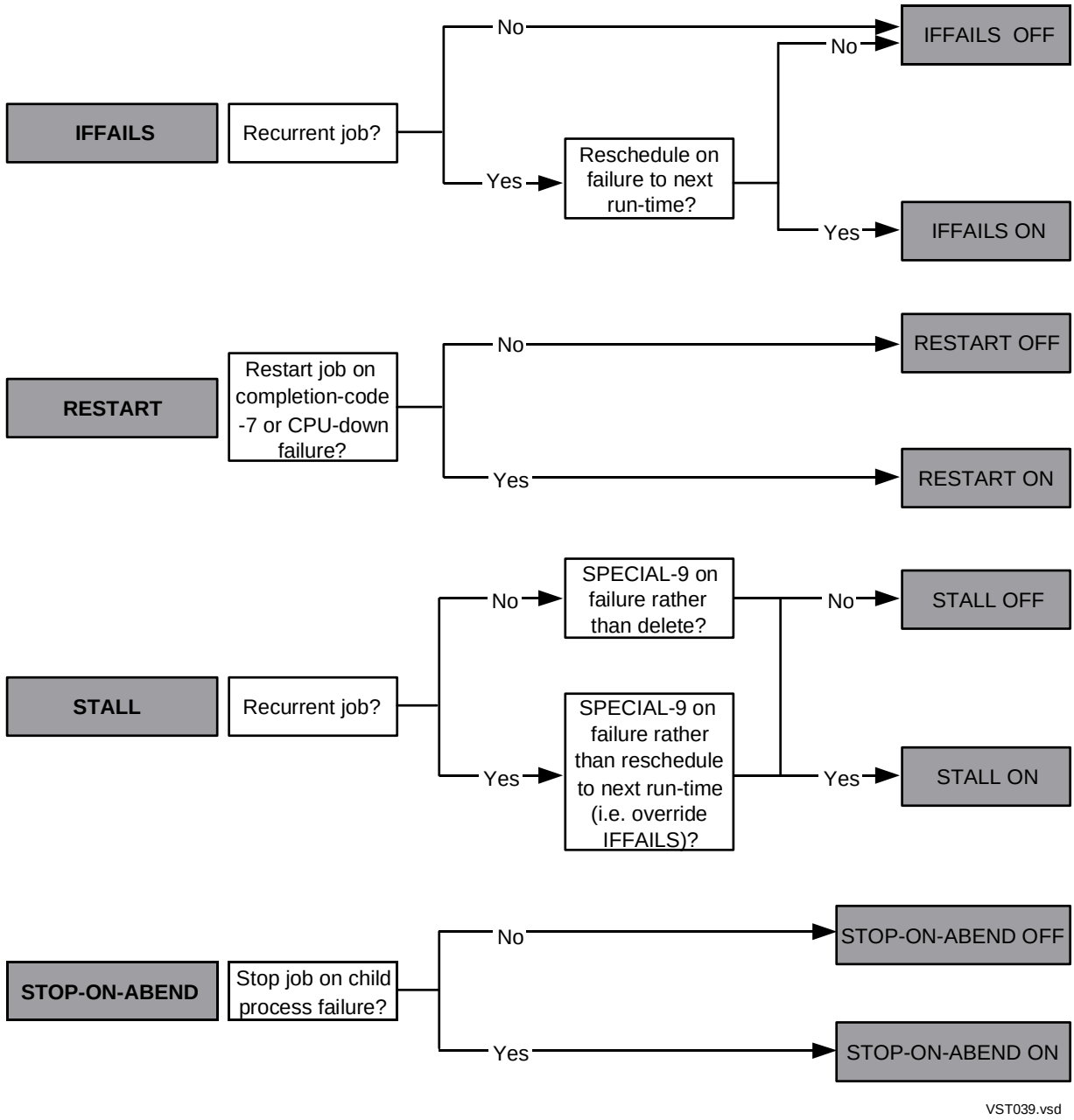
Table 4-5. Job States on Process Failure

	Attributes				Executor Program Process		Child of Executor Program Process	
Job Type	IFFAILS	RESTART	STALL	STOP-ON-ABEND	Stopped With CC 7 or CPU Failed	Abended With Any CC or Stopped With CC -3, -2, -1, 2, 3, 4, 5, or 6	CPU Failed	Abended With Any CC or Stopped With CC -3, -2, -1, 2, 3, 4, 5, or 6
Recurrent	OFF	ON	ON	ON	READY	SPECIAL-5	READY	SPECIAL-5
	OFF	ON	ON	OFF	READY	SPECIAL-5	EXECUTING	EXECUTING
	OFF	ON	OFF	ON	READY	SPECIAL-5	READY	SPECIAL-5
	OFF	ON	OFF	OFF	READY	SPECIAL-5	EXECUTING	EXECUTING
	OFF	OFF	ON	ON	SPECIAL-6	SPECIAL-6	SPECIAL-6	SPECIAL-6
	OFF	OFF	ON	OFF	SPECIAL-6	SPECIAL-6	EXECUTING	EXECUTING
	OFF	OFF	OFF	ON	SPECIAL-6	SPECIAL-6	SPECIAL-6	SPECIAL-6
	OFF	OFF	OFF	OFF	SPECIAL-6	SPECIAL-6	EXECUTING	EXECUTING
	ON	OFF	OFF	OFF	TIME	TIME	EXECUTING	EXECUTING
	ON	OFF	OFF	ON	TIME	TIME	TIME	TIME
	ON	OFF	ON	OFF	SPECIAL-9	SPECIAL-9	EXECUTING	EXECUTING
	ON	OFF	ON	ON	SPECIAL-9	SPECIAL-9	SPECIAL-9	SPECIAL-9
	ON	ON	OFF	OFF	READY	TIME	EXECUTING	EXECUTING
	ON	ON	OFF	ON	READY	TIME	READY	TIME
	ON	ON	ON	OFF	READY	SPECIAL-9	EXECUTING	EXECUTING
	ON	ON	ON	ON	READY	SPECIAL-9	READY	SPECIAL-9
Nonrecurrent	N.A.	OFF	ON	ON	SPECIAL-9	SPECIAL-9	SPECIAL-9	SPECIAL-9
		OFF	ON	OFF	SPECIAL-9	SPECIAL-9	EXECUTING	EXECUTING
		OFF	OFF	ON	Job deleted	Job deleted	Job deleted	Job deleted
		OFF	OFF	OFF	Job deleted	Job deleted	EXECUTING	EXECUTING
		ON	OFF	OFF	READY	Job deleted	EXECUTING	EXECUTING
		ON	OFF	ON	READY	Job deleted	READY	Job deleted
		ON	ON	OFF	READY	SPECIAL-9	EXECUTING	EXECUTING
		ON	ON	ON	READY	SPECIAL-9	READY	SPECIAL-9

Determining a Job's IFFAILS, RESTART, STALL, and STOP-ON-ABEND Attributes

Figure 4-9 shows a decision chart that can help you determine the values of your jobs' IFFAILS, RESTART, STALL, and STOP-ON-ABEND attributes. For more information on the attributes, see Section 7, Attributes.

Figure 4-9. Decision Chart for Setting IFFAILS, RESTART, STALL, and STOP-ON-ABEND Attributes



Example

This example shows the effect of the IFFAILS, RESTART, STALL, and STOP-ON-ABEND attributes on two recurrent jobs whose CPU fails. The jobs have the same RESTART, STALL, and STOP-ON-ABEND attributes, but different IFFAILS attributes. Job A has the IFFAILS OFF attribute, so the scheduler puts the job in the SPECIAL-6 state after failure. Job B has the IFFAILS ON attribute, which causes the scheduler to put the job in the TIME state after failure.

```
23> TIME
April 1, 1993 11:49:22
24> BATCHCOM $ZBAT; SUBMIT JOB A, IN DELAY10, EVERY 7 DAYS,
IFFAILS OFF, RESTART OFF, STALL OFF, STOP-ON-ABEND ON
Job A job number 466 submitted
25> BATCHCOM $ZBAT; SUBMIT JOB B, LIKE A, IFFAILS ON
Job B job number 467 submitted
26> BATCHCOM $ZBAT; STATUS JOB *
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
466  A                255,255 206  EXECUTING   DEFAULT
467  B                255,255 208  EXECUTING   DEFAULT
27> STATUS *, GMOMJOBID $ZBAT.466
Process … Program file Hometerm
$Z204 3,12 … $SYSTEM.SYS00.TACL $ZBAT
3,13 … $SYSTEM.SYS00.DELAY $ZBAT
28> STATUS *, GMOMJOBID $ZBAT.467
Process … Program file Hometerm
$Z205 3,11 … $SYSTEM.SYS00.TACL $ZBAT
3,14 … $SYSTEM.SYS00.DELAY $ZBAT
29> DIVER /CPU 3/
PROCESSOR FAILURE: 3
30> BATCHCOM $ZBAT; STATUS JOB *
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
466  A                255,255 206 6:Fail RsOff DEFAULT
467  B                  255,255 208 08APR93       DEFAULT
```

Specifying a Job's Tape Drives Requirements

The TAPEDRIVES job attribute specifies how many tape drives a job needs when it runs. The scheduler assesses tape drive availability before job startup by comparing the number of drives the job needs with the drives available. ("Drives available" is the value of the TAPEDRIVES scheduler attribute unless the drive is in use.) The job is eligible to run when enough drives are available.

Example

This example shows submission of two jobs, each of which requires two tape drives. Job N runs when submitted and uses two of the three drives specified for the

scheduler. Job O waits to run until job N finishes, because not enough drives are available until then.

```
10} STATUS SCHEDULER
.
.
… TAPEDRIVES …
… ------------- …
… CONFIGURED 3 …
… IN USE 0 …
.
.
11} SUBMIT JOB N, TAPEDRIVES 2
Job N job number 720 submitted
12} STATUS SCHEDULER
.
.
… TAPEDRIVES …
… ------------- …
… CONFIGURED 3 …
… IN USE 2 …
.
.
13} SUBMIT JOB O, TAPEDRIVES 2
Job O job number 721 submitted
14} STATUS JOB *
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
720  N                255,255      EXECUTING DEFAULT
721  O                255,255 TAPE           DEFAULT
15} STATUS JOB *
OB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
721 O 255,255 EXECUTING DEFAULT
```

Specifying a Job's Dependencies

The WAITON attribute enables you to make execution of a job dependent on the job's release by up to 50 other jobs. A job with the WAITON attribute is a dependent job. The jobs on which it depends are master jobs. The relationships between the dependent job and its masters are dependencies.

Example

The WAITON attribute in this example specifies that execution of a job depends on the job's release by two masters. The scheduler puts the dependent job in the EVENT state pending its release.

```
12} SUBMIT JOB DEPENDENT-JOB, WAITON (MASTER-JOB-1, MASTER-
JOB-2)
Job DEPENDENT-JOB Jobnumber 20 submitted
13} INFO JOB 20, WAITON
JOB ATTRIBUTES for DEPENDENT-JOB
```

```
jobnumber: 20
waiton: MASTER-JOB-1, Not Released
MASTER-JOB-2, Not Released
14} STATUS JOB 20
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
20   DEPENDENT-JOB      255,205 EVENT             DEFAULT
```

Releasing a Dependent Job

A dependent job does not run until released by all its masters. For the master jobs to release the dependent job, each must do one of:

- Invoke the TACL macro ZBAT:RELEASE—if the master job's executor program is a TACL process.
- Execute an explicit or implicit $RELEASE command—if the master job's executor program is an NBEXEC process. NBEXEC automatically executes an implicit $RELEASE * command if the job runs without errors.
- Execute the NetBatch programmatic command RELEASE JOB.
- Terminate normally (if the dependent job has the attribute WAITON master-job STOP or WAITON master-job STOP-ABEND) or abnormally (WAITON master-job STOP-ABEND).

To release dependent jobs from their masters without running the masters, use the RELEASE-WAITON Command on page 6-115.

ZBAT:RELEASE

The ZBAT:RELEASE macro comes with NetBatch software (in product file ZBATSEGF). When invoked from a master job's input file, the macro makes the master job's scheduler release the nominated dependent jobs. The syntax of the statement that invokes the macro is:

```
ZBAT:RELEASE [ [ \node. ] $schd ] job-name-range [ error ]
```

node

is a node name. The default is the node where the master job's scheduler is running.

schd

is the process name of the scheduler of the dependent jobs specified by *job-name-range*. The default is the name of the master job's scheduler process.

job-name-range

is a dependent job name or a range of dependent job names specified with the asterisk (*) and question mark (?) wild-card characters.

error

causes the scheduler to log an error or warning number in the master job's executor-program output file when a release of a dependent job is unsuccessful. The NetBatch message corresponding to the number indicates the reason the release was unsuccessful. (For example, error 2099—"job does not exist"—would appear in the output file if *job-name-range* specified a nonexistent job.) The scheduler does not set error if the release succeeds.

To test error and retry an unsuccessful release or quit, use the TACL #IF built-in function; Figure 4-10 shows sample input-file statements that do this. For more information on the function, see the *TACL Reference Manual*.

Figure 4-10. Sample #IF Statements for ZBAT:RELEASE Error

```
#PUSH ERROR
ZBAT:RELEASE \MELBDEV.$ZBAT DEPENDENT-JOB ERROR
[ #IF [ ERROR ]
  |THEN|
    [ ZBAT:RELEASE \MELBDEV.$ZBAT DEPENDENT-JOB ERROR ]
    [ #IF [ ERROR ]
      |THEN|
        [ #OUTPUT ERROR RELEASING DEPENDENT-JOB ]
    ]
]
```

VST112.vsd

$RELEASE

The $RELEASE command is an NBEXEC command that performs a similar function to ZBAT:RELEASE. The syntax of the command is:

```
$RELEASE [ [ \ node. ] $ schd , ] job-name-range
```

node

is a node name. The default is the node where the master job's scheduler is running.

schd

is the process name of the scheduler of the dependent jobs specified by *job-name-range*. The default is the name of the master job's scheduler process.

job-name-range

is a dependent job name or a range of dependent job names specified with the asterisk (*) and question mark (?) wild-card characters.

Examples

- This example shows submission of a TACL master job that invokes the ZBAT:RELEASE macro to release a dependent job. The scheduler flags the dependent job as released by the master job after it executes the master.

```
1} INFO JOB DEPENDENT-JOB, WAITON
JOB ATTRIBUTES for DEPENDENT-JOB
jobnumber: 20
waiton: MASTER-JOB-1, Not Released
MASTER-JOB-2, Not Released
2} RUN EDIT; GET TACLIN !; ADD
TEXT EDITOR - T9601D20 - (01JUN93)
CURRENT FILE IS $DATA7.USER.TACLIN
1 ZBAT:RELEASE \MELBDEV.$ZBAT DEPENDENT-JOB
2 //
*EXIT
3} SUBMIT JOB MASTER-JOB-1, EXECUTOR-PROGRAM TACL, IN TACLIN
Job MASTER-JOB-1 Jobnumber 21 submitted
4} INFO DEPENDENT-JOB, WAITON
JOB ATTRIBUTES for DEPENDENT-JOB
jobnumber: 20
waiton: MASTER-JOB-1, Released
MASTER-JOB-2, Not Released
```

- This example shows submission of an NBEXEC master job that executes the $RELEASE command to release a dependent job. The scheduler flags the dependent job as released by the master job after it executes the master.

```
10} INFO DEPENDENT-JOB, WAITON
JOB ATTRIBUTES for DEPENDENT-JOB
jobnumber: 20
waiton: MASTER-JOB-1, Released
MASTER-JOB-2, Not Released
11} RUN EDIT; GET NBEXECIN !; ADD
TEXT EDITOR - T9601D20 - (01JUN93)
CURRENT FILE IS $DATA7.USER.NBEXECIN
1 $RELEASE \MELBDEV.$ZBAT, DEPENDENT-JOB
2 //
*EXIT
12} SUBMIT JOB MASTER-JOB-2, EXECUTOR-PROGRAM NBEXEC, IN
NBEXECIN
Job MASTER-JOB-2 Jobnumber 22 submitted
13} INFO DEPENDENT-JOB, WAITON
JOB ATTRIBUTES for DEPENDENT-JOB
jobnumber: 20
waiton: MASTER-JOB-1, Released
MASTER-JOB-2, Released
```

Specifying a Job's ASSIGNs, DEFINEs, and PARAMs

The ATTACHMENT-SET attribute links an attachment set to a job. When the job runs, the scheduler propagates the set's ASSIGNs, DEFINEs, and PARAMs to the job's executor-program process.

Example

This example shows submission of a TACL job whose ATTACHMENT-SET attribute specifies the job's ASSIGNs, DEFINEs, and PARAMs. The job writes its ASSIGNs, DEFINEs, and PARAMs to an output file to illustrate their propagation by the scheduler.

```
> FUP COPY INFILE
ASSIGN
INFO DEFINE **
PARAM
> BATCHCOM $SCHD; INFO ATTACHMENT-SET (SUPER.FPP)STANDARD,
ASSIGN, DEFINE, PARAM
ATTACHMENT-SET ATTRIBUTES for (SUPER.FPP)STANDARD
attachments: DEFINE =OUT, CLASS SPOOL, LOC \MELBQAT.$S
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
\MELBDEV.$NB.NOSUBVOL
ASSIGN TRANSLOG, $DATA7.OPS.TRANSLOG
PARAM DAY TUESDAY
> BATCHCOM $SCHD; SUBMIT JOB S, EXECUTOR-PROGRAM TACL, IN
INFILE, OUT OUTFILE, ATTACHMENT-SET (SUPER.FPP)STANDARD
Job S job number 9 submitted
> FUP COPY OUTFILE
ASSIGN
TRANSLOG
Physical file: $DATA7.OPS.TRANSLOG
INFO DEFINE **
Define Name =OUT
CLASS SPOOL
LOC \MELBQAT.$S
Define Name =_DEFAULTS
CLASS DEFAULTS
VOLUME \MELBDEV.$NB.NOSUBVOL
PARAM
DAY .TUESDAY.
```

Adding Attachment Sets

The ADD ATTACHMENT-set command adds an attachment set to a scheduler. For example:

```
3} CHANGEUSER FPP.MANAGER password
3} ADD ATTACHMENT-SET SALES, (ASSIGN ORDERS,
$BIG2.MRKTNG.ORDERS), (PARAM PERIOD Q31993),
(PARAM REGION 8), (PARAM COMMISSION 10%)
Attachment-set (FPP.MANAGER)SALES added
```

Attachment-Set Attributes

An attachment set's attributes specify its ASSIGNs, DEFINEs, PARAMs, and security, and determine whether the set is permanent or temporary. Table 4-6 on page 4-36 lists the attributes, describes their functions, and gives their default values. For more information about the attributes, see Section 7, Attributes.

Table 4-6. Attachment-Set Attributes

Attribute	Function	Default Value
ASSIGN	Specifies the name and attributes of an ASSIGN.	No default.
DEFINE	Specifies the name and attributes of a DEFINE.	DEFINE =_DEFAULTS specifying the node and volume set by the last SET ATTACHMENT-SET (DEFINE =_DEFAULTS, VOLUME …), SYSTEM, or VOLUME command. If there was no such command, the DEFINE specifies the current node and volume from the TACL environment.
PARAM	Specifies the name and value of a PARAM.	No default.
SECURITY	Controls user access to an attachment set.	UUUU.
TEMPORARY	Determines whether the scheduler automatically deletes an attachment set not assigned to a job.	• OFF for attachment sets added with a user-specified name. • ON for attachment sets added with a scheduler-generated ID. For more information, see Using #CURRENT on page 4-38.

Making Attachment-Set Inquiries

The INFO ATTACHMENT-SET and STATUS ATTACHMENT-SET commands display information about attachment sets.

- INFO ATTACHMENT-SET displays all attributes of an attachment set or sets, or specified attributes only. For example:

```
17} INFO ATTACHMENT-SET (FPP.MANAGER)SALES, DETAIL
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)SALES
security: "UUUU"
temporary: Off
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$DATA6.SALES
ASSIGN ORDERS, $BIG2.MRKTNG.ORDERS
PARAM PERIOD Q31993
PARAM REGION 8
PARAM COMMISSION 10%
```

- STATUS ATTACHMENT-SET lists attachment sets and the names and owners of jobs using those sets. For example:

```
22} STATUS ATTACHMENT-SET (*.*)*
ATTACHMENT SET IN USE BY JOBS
-------------------------------- ----------------------
(NB.USER)USER-1 (NB.USER)STATISTICS
(NB.USER)USER-2 None
```

```
(FPP.MANAGER)ADDRESS (FPP.MANAGER)MAILOUT
(FPP.MANAGER)SALES (SUPER.SUPER)Q3-BONUSES
(SUPER.FPP)5 None
(SUPER.SUPER)6 None
```

Altering Attachment-Set Attributes

The ALTER ATTACHMENT-SET command changes attachment-set attributes. In particular, the command lets you:

- Add ASSIGNs, DEFINEs, and PARAMs to an attachment set. For example:

```
34} INFO ATTACHMENT-SET (FPP.MANAGER)SALES, DEFINE
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)SALES
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$DATA6.SALES
35} ALTER ATTACHMENT-SET (FPP.MANAGER)SALES, (DEFINE =OUT,
CLASS SPOOL, LOC \MELBDEV.$S.#FPP, HOLDAFTER ON,
COPIES 2)
Attachment-set (FPP.MANAGER)SALES altered
36} INFO ATTACHMENT-SET (FPP.MANAGER)SALES, DEFINE
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)SALES
attachments: DEFINE =OUT, CLASS SPOOL, LOC
\MELBDEV.$S.#FPP, COPIES 2,
HOLDAFTER ON
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$DATA6.SALES
```

- Change ASSIGNs and PARAMs. For example:

```
50} INFO ATTACHMENT-SET (FPP.MANAGER)SALES, ASSIGN
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)SALES
attachments: ASSIGN ORDERS, $BIG2.MRKTNG.ORDERS
51} ALTER ATTACHMENT-SET (FPP.MANAGER)SALES, (ASSIGN
ORDERS, $DATA7.PRODUCT.ORDERS)
Attachment-set (FPP.MANAGER)SALES altered
52} INFO ATTACHMENT-SET (FPP.MANAGER)SALES, ASSIGN
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)SALES
attachments: ASSIGN ORDERS, $DATA7.PRODUCT.ORDERS
```

- Alter or clear attributes of DEFINEs. For example:

```
54} INFO ATTACHMENT-SET (FPP.MANAGER)SALES, DEFINE =OUT
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)SALES
attachments: DEFINE =OUT, CLASS SPOOL, LOC
\MELBDEV.$S.#FPP, COPIES 2,
HOLDAFTER ON
55} ALTER ATTACHMENT-SET (FPP.MANAGER)SALES, (DEFINE =OUT,
LOC \MELBORN.$S, COPIES, HOLDAFTER)
Attachment-set (FPP.MANAGER)SALES altered
56} INFO ATTACHMENT-SET (FPP.MANAGER)SALES, DEFINE =OUT
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)SALES
attachments: DEFINE =OUT, CLASS SPOOL, LOC \MELBORN.$S
```

- Alter SECURITY and TEMPORARY attributes; for example:

```
57} INFO ATTACHMENT-SET (FPP.MANAGER)SALES, SECURITY,
TEMPORARY
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)SALES
security: "UUUU"
temporary: Off
58} ALTER ATTACHMENT-SET (FPP.MANAGER)SALES, SECURITY
"GOGO", TEMPORARY ON
Attachment-set (FPP.MANAGER)SALES altered
59} INFO ATTACHMENT-SET (FPP.MANAGER)SALES, SECURITY,
TEMPORARY
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)SALES
security: "GOGO"
temporary: On
```

Deleting Attachment Sets and Attachment-Set ASSIGNs, DEFINEs, and PARAMs

The DELETE ATTACHMENT-SET command lets you:

- Delete specified ASSIGNs, DEFINEs, and PARAMs from an attachment set. For example:

```
2} INFO ATTACHMENT-SET (FPP.MANAGER)SALES, PARAM
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)SALES
attachments: PARAM PERIOD Q31993
PARAM REGION 8
PARAM COMMISSION 10%
3} DELETE ATTACHMENT-SET (FPP.MANAGER)SALES, PARAM REGION
Attachment-set (FPP.MANAGER)SALES altered
4} INFO ATTACHMENT-SET (FPP.MANAGER)SALES, PARAM
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)SALES
attachments: PARAM PERIOD Q31993
PARAM COMMISSION 10%
```

- Delete an attachment set from a scheduler. For example:

```
5} DELETE ATTACHMENT-SET (FPP.MANAGER)SALES
Attachment-set (FPP.MANAGER)SALES deleted
```

Using #CURRENT

#CURRENT is a BATCHCOM variable available for use when dealing with attachment sets. The variable's functions are to:

- Point to the attachment set specified by the last ADD or ALTER *attachment-set* command. This feature enables you to specify the set in later DELETE, INFO, and STATUS *attachment-set* commands and the ATTACHMENT-SET job attribute by specifying #CURRENT instead of the set's name. For example:

```
58} ADD ATTACHMENT-SET (SUPER.FPP)ADMIN
Attachment-set (SUPER.FPP)ADMIN added
59} INFO ATTACHMENT-SET (SUPER.FPP)ADMIN, DETAIL
```

```
ATTACHMENT-SET ATTRIBUTES for (SUPER.FPP)ADMIN
security: "UUUU"
temporary: Off
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$NB.SUPER
60} INFO ATTACHMENT-SET #CURRENT, DETAIL
ATTACHMENT-SET ATTRIBUTES for (SUPER.FPP)ADMIN
security: "UUUU"
temporary: Off
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$NB.SUPER
61} SUBMIT JOB X, ATTACHMENT-SET #CURRENT
Job X job number 5 submitted
62} INFO JOB X, ATTACHMENT-SET
JOB ATTRIBUTES for X
jobnumber: 5
attachment-set: (SUPER.FPP)ADMIN
```

#CURRENT has a null value at the start of a BATCHCOM session. Also, BATCHCOM resets the value to null on execution of a RESET or SET attachment-set command or an OPEN scheduler command. (The INFO ATTACHMENT-SET command indicates a null value for #CURRENT by an "undefined substitution" message.) For example:

```
32> BATCHCOM $ZBAT
1} INFO ATTACHMENT-SET #CURRENT, DETAIL
-^-0345-
E Undefined substitution
2} ALTER ATTACHMENT-SET (SUPER.SUPER)DATE, …
Attachment-set (SUPER.SUPER)DATE altered
3} INFO ATTACHMENT-SET #CURRENT, DETAIL
ATTACHMENT-SET ATTRIBUTES for (SUPER.SUPER)DATE
.
.
4} OPEN $SCHD
NETBATCH SERVER - T9190D20 … Time: 09APR93 13:25:58
5} INFO ATTACHMENT-SET #CURRENT, DETAIL
-^-0345-
E Undefined substitution
```

- Assign a scheduler-generated ID to an attachment set added with #CURRENT specified instead of a set name. The ID contains the name of the current user and a numeric identifier in the form (*group-name.user-name*) number. For example:

```
9} CHANGEUSER NB.USER password
9} ADD ATTACHMENT-SET #CURRENT
Attachment-set (NB.USER)4 added
```

- Create automatically an attachment set for a job when the job's ATTACHMENT-SET attribute specifies #CURRENT, and #CURRENT has a null value. The set has a scheduler-generated ID and these attributes:

Attribute	Value
ASSIGN	ASSIGN attributes from working-attributes set. For information on the working-attributes set, see the descriptions of the SET ATTACHMENT-SET Command on page 6-130, SHOW ATTACHMENT-SET Command on page 6-142, rptype on page 6-116, and ADD ATTACHMENT-SET Command on page 6-34.
DEFINE	DEFINE attributes from working-attributes set.
PARAM	PARAM attributes from working-attributes set.
SECURITY	UUUU unless overridden by SECURITY attribute in working-attributes set.
TEMPORARY	ON unless overridden by TEMPORARY attribute in working-attributes set.

For example:

```
2} CHANGEUSER FPP.QA password
2} SET ATTACHMENT-SET (DEFINE =_DEFAULTS, CLASS DEFAULTS,
VOLUME \MELBDEV.$BIG2.QA), (ASSIGN LOG,
\MELBORN.$FPP.QA.LOG)
3} SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
\MELBDEV.$BIG2.QA
ASSIGN LOG, \MELBORN.$FPP.QA.LOG
4} INFO ATTACHMENT-SET #CURRENT, DETAIL
-^-0345-
E Undefined substitution
5} SUBMIT JOB QA-TEST, ATTACHMENT-SET #CURRENT
Attachment-set (FPP.QA)4 added
Job QA-TEST job number 6 submitted
6} INFO JOB QA-TEST, ATTACHMENT-SET
JOB ATTRIBUTES for QA-TEST
jobnumber: 6
attachment-set: (FPP.QA)4
7} INFO ATTACHMENT-SET #CURRENT, DETAIL
ATTACHMENT-SET ATTRIBUTES for (FPP.QA)4
security: "UUUU"
temporary: On
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
\MELBDEV.$BIG2.QA
ASSIGN LOG, \MELBORN.$FPP.QA.LOG
```

Specifying a Job's Description

The DESCRIPTION job attribute contains ASCII text that describes a job. The text appears in the scheduler's log file each time the job runs.

Example

This example shows the ALTER JOB command adding a description to a job. The percent (%) characters force line breaks.

```
9} ALTER JOB PAYROLL, DESCRIPTION "--------- Payroll Run ----
------% On failure: Check line #7% Restore EMP9% Rerun%
See $SDOC.DOCS.PAY for details%-----------------------------
---
--"
Job PAYROLL Jobnumber 19 altered
10} INFO JOB 19, DESCRIPTION
JOB ATTRIBUTES for PAYROLL
jobnumber: 19
description:
---------- Payroll Run -----------
On failure: Check line #7
Restore EMP9
Rerun
See $SDOC.DOCS.PAY for details
---------------------------------
```

Managing Jobs

This subsection describes various tasks for managing jobs:

Topic	Page
Displaying Job Status	4-42
Altering Job Attributes	4-44
Overriding Job Dependencies, Timing Attributes, and Selection Priority	4-44
Suspending and Reactivating Job Processes	4-45
Stopping and Deleting Jobs	4-46
Dealing With Job Output	4-47

Displaying Job Status

You can display the status of all jobs or specified jobs only by using the STATUS JOB command. The command displays the number, name, owner, processing state, and class for each job listed. It also displays the spooler-job number of the log file of jobs whose OUT attributes specify spooler locations. For example:

```
32} STATUS JOB *
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
755  COBOL-COMPILE    255,205 23:30:00 COMPILES
756  SPOOLER-CLEANUP  205,70  184   EXECUTING  DEFAULT
757  USERS             133,2    186   EXECUTING   DEFAULT
```

You can display the jobs' SELPRI attributes instead of their log file numbers by including the SELPRI qualifier in the command. For example:

```
33} STATUS JOB *, SELPRI
JOB  JOBNAME          USERID  SEL  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
755  COBOL-COMPILE    255,205 1 23:30:00 COMPILES
756  SPOOLER-CLEANUP  205,70  6    EXECUTING   DEFAULT
757  USERS             133,2    5    EXECUTING    DEFAULT
```

Processing States

The State column of the STATUS JOB display shows job states. These states (see Table 4-7) indicate whether jobs are running and, if not, why. For detailed information on the states, see STATUS JOB Command on page 6-165.

Table 4-7. Job State (page 1 of 2)

State	Description
EVENT	The job has the WAITON attribute and does not run until released.
EXECUTING	The job is running.
OVER LIMIT	The job exceeded its execution time limit but is still running.

Table 4-7. Job State (page 2 of 2)

State	Description
READY	The job is available for execution.
RUNNEXT	The job was the subject of a RUNNEXT JOB command and runs as soon as an executor associated with its class is available.
RUNNOW	The job was the subject of a RUNNOW JOB command or has the AT attribute. It is in this state momentarily until execution begins.
SPECIAL-*n*	The job is on hold for one of these reasons: ● The job's SUBMIT JOB command specified the HOLD ON attribute. ● An ALTER JOB command altered the job's HOLD attribute from OFF to ON before the job ran. ● The scheduler put the job on hold after it ran and stopped because the job had the attribute HOLDAFTER ON. ● The scheduler put the job on hold during execution because of a problem requiring user intervention. *n* indicates the reason the job is on hold and has one of these values: 1 1:Hold The job is on hold for one of the first three reasons listed in the preceeding description of SPECIAL- n. 2 2:NB failed The job was running when an event other than execution of an ABORT SCHEDULER or SHUTDOWN SCEDULER command stopped its scheduler's processes. 3 3:NEWP.err The scheduler tried to create a new process for the job's executor program but failed. 4 4:Start err The scheduler successfully created a new process for the job's executor program, but the program failed during startup. 5 5:Fail RsOn The job has the RESTART ON attribute and abended, but did not restart because its attributes include IFFAILS OFF. This state applies only to recurrent jobs. 6 6:Fail RsOff The job has the RESTART OFF attribute and abended, but did not restart because its attributes include IFFAILS OFF. This state appliesonly to recurrent jobs. 7 7:CAL error An error occurred when the scheduler tried to open the job's BATCHCAL calendar file. 8 8:CAL exprd The job's BATCHCAL calendar has run out of dates. 9 9:STALL The job has the attribute STALL ON and failed while running.
SUSPENDED	The job was the subject of a SUSPEND JOB command.
TAPE	The job has the TAPEDRIVES attribute and requires more drives than are available.
TIME (*hh:mm:ss)*	The job has one of these attributes delaying execution: AFTER, AT, CALENDAR, EVERY, or WAIT.

Altering Job Attributes

The ALTER JOB command changes a job's attributes. For example:

```
3} INFO JOB COBOL-COMPILE, OUT, PRI
JOB ATTRIBUTES for COBOL-COMPILE
jobnumber: 17
out: \MELBDEV.$S.#COBOL
pri: 119
4} ALTER JOB 17, OUT $DATA7.PURGE.CMPL3, PRI 149
Job COBOL-COMPILE Jobnumber 17 altered
5} INFO JOB COBOL-COMPILE, OUT, PRI
JOB ATTRIBUTES for COBOL-COMPILE
jobnumber: 17
out: \MELBDEV.$DATA7.PURGE.CMPL3
pri: 149
```

The ALTER JOB command updates a job's description in the scheduler database. Whether the updated description applies to the job when it runs, however, depends on the job's processing state.

- If the job's state is EVENT, READY, RUNNEXT, RUNNOW, SPECIAL-*n*, TAPE, or TIME, the updated description applies when the job runs.
- If the job's state is EXECUTING, OVER LIMIT, or SUSPENDED, the only attributes that can apply are DESCRIPTION, HOLDAFTER, IFFAILS, LIMIT, PURGE-IN-FILE, RESTART, SAVEABEND, STALL, and STOP-ON-ABEND. If the job is also recurrent, the updated description applies in full the next time the job runs. (A recurrent job has the CALENDAR or EVERY attribute.)

Overriding Job Dependencies, Timing Attributes, and Selection Priority

NetBatch supervisors can use the RUNNEXT JOB and RUNNOW JOB commands to override jobs' dependencies, timing attributes, and selection priorities. The commands have the effect of promoting the specified jobs to run before all others in their respective classes.

RUNNEXT JOB

The RUNNEXT JOB command makes the scheduler run a job immediately when an executor associated with the job's class is available. This example shows the command making the scheduler run job C before job B instead of after it:

```
47} STATUS JOB *, SELPRI
JOB  JOBNAME          USERID  SEL  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
758  A                255,205 7    EXECUTING   DEFAULT
759  B                255,205 6    READY       DEFAULT
760  C                255,205 1    READY       DEFAULT
48} RUNNEXT JOB C
Job C job number 760 will run next
```

```
49} STATUS JOB *, SELPRI
JOB  JOBNAME          USERID  SEL  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
758  A                255,205 7    EXECUTING   DEFAULT
759  B                255,205 6    READY       DEFAULT
760  C                255,205 1    RUNNEXT     DEFAULT
50} STATUS JOB *, SELPRI
JOB  JOBNAME          USERID  SEL  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
759  B                255,205 6    READY       DEFAULT
760  C                 255,205  1    EXECUTING    DEFAULT
```

RUNNOW JOB

The RUNNOW JOB command makes the scheduler run a job immediately. The command makes the scheduler create a temporary executor for the job. The scheduler deletes the temporary executor when the job finishes. For example:

```
69} STATUS JOB *, SELPRI
JOB  JOBNAME          USERID  SEL  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
761  D                255,205 3    READY       DEFAULT
762  E                255,205 7    EXECUTING   DEFAULT
763  F                255,205 7    READY       DEFAULT
70} STATUS EXECUTOR *
EXECUTOR                  CPU STATE  JOB  CLASS
------------------------- --- ------ ---- -------
EX1                       1   ACTIVE 762  DEFAULT
71} RUNNOW JOB 761
Job D job number 761 running
72} STATUS JOB *, SELPRI
JOB  JOBNAME          USERID  SEL  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
761  D                255,205 3    EXECUTING   DEFAULT
762  E                255,205 7    EXECUTING   DEFAULT
763  F                255,205 7    READY       DEFAULT
73} STATUS EXECUTOR *
EXECUTOR                  CPU STATE  JOB  CLASS
------------------------- --- ------ ---- -------
EX1                        1  ACTIVE 762  DEFAULT
__TEMP_EXEC_761 3 DELETE 761 DEFAULT
74} STATUS EXECUTOR *
EXECUTOR                  CPU STATE  JOB  CLASS
------------------------- --- ------ ---- -------
EX1                         1   ACTIVE 762   DEFAULT
```

Suspending and Reactivating Job Processes

The SUSPEND JOB command suspends executing processes associated with a job. The ACTIVATE JOB command reactivates a job's suspended processes. For example:

```
28> BATCHCOM $ZBAT; STATUS JOB 766
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
```

```
766  H               205,70         EXECUTING   DEFAULT
29> STATUS *, GMOMJOBID $ZBAT.766
Process …
$X466 1,51 …
1,52 …
30> BATCHCOM $ZBAT; SUSPEND JOB 766
Job H job number 766 suspended
31> BATCHCOM $ZBAT; STATUS JOB 766
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
766  H                205,70       SUSPENDED   DEFAULT
32> STATUS *, GMOMJOBID $ZBAT.766, DETAIL
Pid: 1,51 ($X466) Primary
Process States: … SUSPENDED
GMOMJOBID: $ZBAT.766
Pid: 1,52
Process States: … SUSPENDED
GMOMJOBID: $ZBAT.766
33> BATCHCOM $ZBAT; ACTIVATE JOB 766
Job H job number 766 activated
34> BATCHCOM $ZBAT; STATUS JOB 766
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
766  H                205,70       EXECUTING   DEFAULT
35> STATUS *, GMOMJOBID $ZBAT.766, DETAIL
Pid: 1,51 ($X466) Primary
Process States: … RUNNABLE
GMOMJOBID: $ZBAT.766
Pid: 1,52
Process States: … RUNNABLE
GMOMJOBID: $ZBAT.766
```

Stopping and Deleting Jobs

The STOP JOB command terminates processes associated with an executing, over-limit, or suspended job. The DELETE JOB command deletes from the scheduler a job that is not executing, over limit, or suspended. For example:

```
38> BATCHCOM $ZBAT; SUBMIT JOB I, IN DELAY04, HOLDAFTER ON
Job I job number 767 submitted
39> BATCHCOM $ZBAT; STATUS JOB 767
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
767  I                205,70       EXECUTING   DEFAULT
40> STATUS *, GMOMJOBID $ZBAT.767, DETAIL
Pid: 1,51 ($X467) Primary
Process States: … RUNNABLE
GMOMJOBID: $ZBAT.767
Pid: 1,52
Process States: … RUNNABLE
GMOMJOBID: $ZBAT.767
41> BATCHCOM $ZBAT; STOP JOB 767
Job I job number 767 stopped
42> STATUS *, GMOMJOBID $ZBAT.767, DETAIL
```

```
43> BATCHCOM $ZBAT; STATUS JOB 767
JOB  JOBNAME          USERID  LOG  STATE       CLASSNAME
---- ---------------- ------- ---- ----------- ---------
767  I                205,70  1:Hold           DEFAULT
44> BATCHCOM $ZBAT; DELETE JOB 767
Job I job number 767 deleted
45> BATCHCOM $ZBAT; STATUS JOB 767
2099-E JOB does not exist
```

Dealing With Job Output

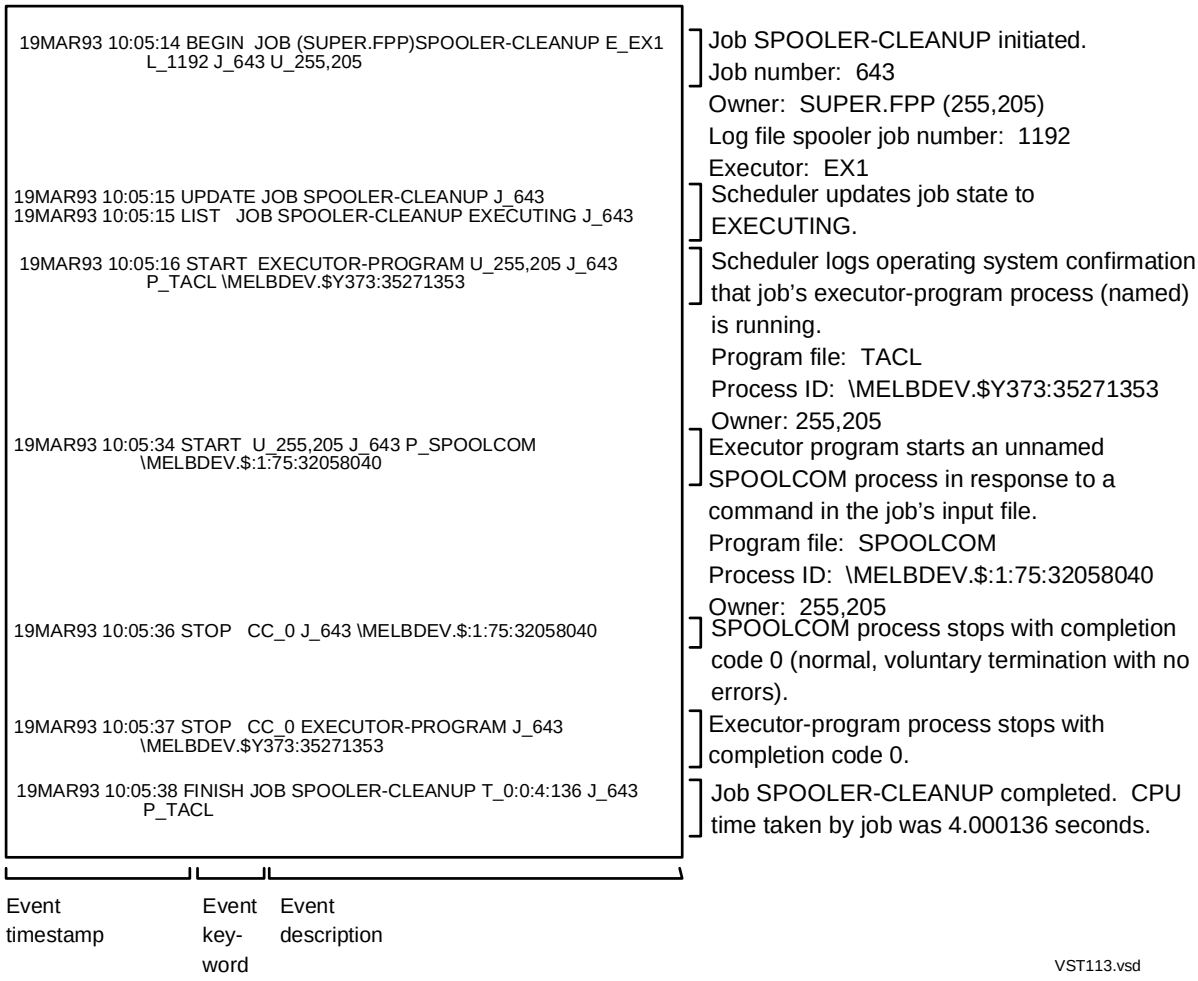
Job output is data produced by an executing job and written to the output file specified by the job's OUT attribute. The data can include the job's log file, executor-program output, and output from processes started by the executor program.

Job Log Files

Job log files record details of events occurring during job execution. The scheduler creates and opens the files as specified by the jobs' JOB-LOG and OUT attributes.

Figure 4-11 on page 4-48 shows a commented example of a job log file. The example lists events that occur when a job's TACL executor-program process executes a SPOOLCOM command. (The examples in Figure 4-12 on page 4-49 and Figure 4-13 on page 4-49 relate to this job.)

Figure 4-11. Sample Job Log File



Interpreting the Contents of a Job Log File

The scheduler records events in a job log file in the same way that it records events in a scheduler log file. Events appear in chronological order, and each has a timestamp and keyword identifying its type.

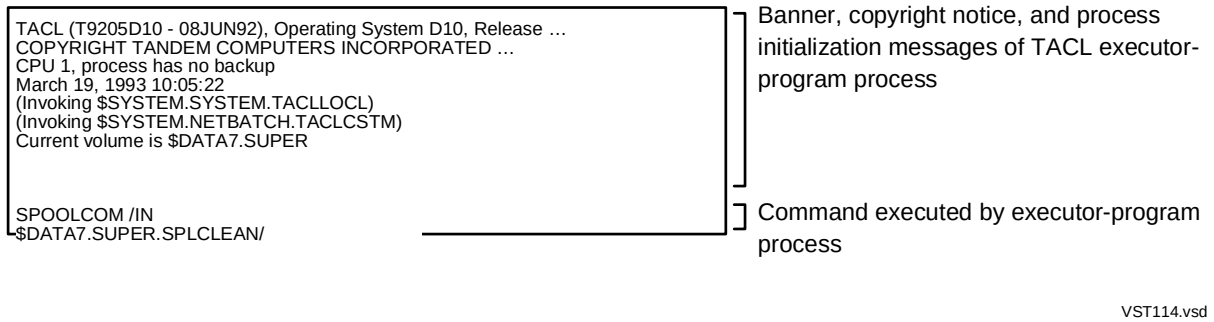
For information about event keywords and logging formats in event descriptions, see Dealing With Scheduler Log Files on page 3-38.

Executor-Program Output

A job's executor-program process writes the commands it executes to the job's output file. If the file is a spooler collector process, the scheduler creates a new spooler job for the output.

Figure 4-12 on page 4-49 shows a commented example of executor-program output. The output relates to the job whose log file appears in the previous figure.

Figure 4-12. Example of Executor-Program Output



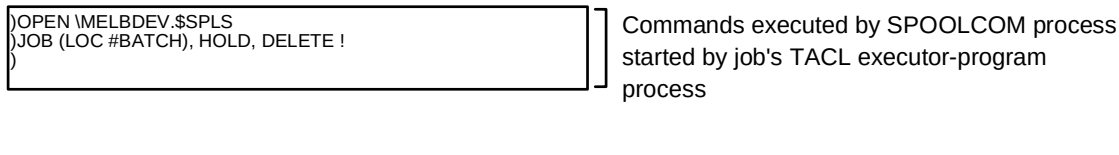
Output From Other Processes

Processes started by an executor-program process also write the commands they execute to the job's output file, depending on the output file's type. This table indicates whether you can expect output for a given file type:

Output File Type	Nonexecutor-program Output?
Terminal device	Yes
Nonterminal device	Same as disk file
Spooler collector process	Yes (new spooler job for output of each process)
Nonspooler collector process	Yes, if the output file is different from that of the executor program
Disk file	No, if the output file is the same as that of the executor program (the program has exclusive access to the file). In this case, the process might abend if it tries to open the file, depending on how it handles "file in use" errors.

Figure 4-13 shows a commented example of output from a process started by an executor-program process. The output is from the SPOOLCOM process started in the previous figure.

Figure 4-13. Example of Nonexecutor-Program Output



Manipulating Job Output in the Spooler

For information on how spoolers handle job output and for details of the commands used to manipulate that output, see the manuals for the spooler products you are using.

5

Run Calendar Generation and Display

A run calendar is a disk file generated from user-supplied source data by the BATCHCAL program. It contains a series of dates and times called run times. You can schedule a job to run automatically at those times by using the CALENDAR attribute to assign the file to the job. For more information, see CALENDAR Job Attribute on page 7-25.

Topic	Page
Running BATCHCAL	5-2
Entering Source Data	5-5
Using BATCHCAL Commands	5-13
Generating a Run Calendar	5-14
Displaying Run Times	5-16
Reformatting an Old Calendar File	5-22

Example

This example illustrates the use of a run calendar in scheduling a job:

1. BATCHCAL generates a calendar file (file code 848) based on source data in an EDIT file.
2. BATCHCAL displays the next run time from the generated file.
3. BACTCHAL displays the submission and next run time of a job whose CALENDAR attribute specifies the BATCHCAL file:

```
13> FILEINFO
$DATA7.CALFILES
Code EOF Last Modification …
SRCFRI95 101 2200 26-Oct-94 9:49:37 …
14> FUP COPY SRCFRI95
$DATA7.CALFILES.CALFRI95 == Name of calendar file
14:30 == Default job start time
1995 * FRIDAY == All Fridays during 1995
15> BATCHCAL /IN SRCFRI95/
$DATA7.CALFILES.CALFRI95 == Name of calendar file
14:30 == Default job start time
1995 * FRIDAY == All Fridays during 1995
16> FILEINFO
$DATA7.CALFILES
Code EOF Last Modification …
CALFRI95 848 210 26-Oct-94 9:52:12 …
SRCFRI95 101 2200 26-Oct-94 9:49:37 …
```

```
17> BATCHCAL /IN CALFRI95/ NEXT-DATE
1995-01-06 14:30:00
18> BATCHCOM $ZBAT; SUBMIT JOB FIRST-FRIDAY, CALENDAR
CALFRI95
Job FIRST-FRIDAY Jobnumber 23 submitted
22> BATCHCOM $ZBAT; INFO JOB 23, AFTER, CALENDAR
JOB ATTRIBUTES for FIRST-FRIDAY
jobnumber: 23
calendar: \QA.$DATA7.CALFILES.CALFRI95
next-runtime: 06JAN95 14:30:00
```

Running BATCHCAL

The BATCHCAL program lets you generate a run calendar, display run times, and reformat an old calendar file to the current format. To run the program, use the TACL RUN command:

```
[ RUN ] [ \node. ] [ volume. ] BATCHCAL
[/ IN source-file [ , run-option ]… /]
[/ IN calendar-file [ , run-option ]… /]
{N[EXT-DATE]};
F[ROM-DATE] [yy]yy mmm ] [ ; T[O-DATE] [yy]yy mmm ],
R[E-FORMAT]
{S[HOW-DATES]};
F[ROM-DATE] [yy]yy mmm ] [ ; T[O-DATE] [yy]yy mmm ],
[R[E-FORMAT]]
```

node

is the name of the node where the BATCHCAL program file resides. If the node is remote, you must have remote access to it (that is, you must have remote passwords set up on both local and remote nodes). The default is the node where the process that creates the BATCHCAL process is running.

volume

is one of:

$*volume-name.subvolume-name*

specifies the volume and subvolume containing the BATCHCAL program file.

subvolume-name

specifies the subvolume containing the BATCHCAL program file.

The default volume varies, depending on whether the RUN command is explicit or implicit:

- For an explicit RUN command, enter RUN followed by the BATCHCAL program-file name. If you enter a partially qualified file name, the TACL

program expands it by using the current default node, volume, and subvolume names.

- For an implicit RUN command, enter the BATCHCAL program-file name only. If you enter a partially qualified file name, the TACL program searches for the file in the subvolumes specified by the TACL #PMSEARCHLIST variable.

source-file

is the name of an EDIT file (file code 101) containing source data for a BATCHCAL run calendar. The data structure and format must conform to the rules listed in Entering Source Data on page 5-5.

run-option

is one of these TACL RUN command options. For descriptions of these options, see the RUN[D] command description in the *TACL Reference Manual.*

CPU	INLINE	LIB	OUT	STATUS
DEBUG	INSPECT	MEM	OUTV	SWAP
DEFMODE	INV	NAME	PFS	TERM
EXTSWAP	JOBID	NOWAIT	PRI	WINDOW
HIGHPIN				

calendar-file

is the name of a BATCHCAL-generated calendar file (file code 848) whose run times you want to display.

F[ROM-DATE]

causes BATCHCAL to display run times from the beginning of the specified month.

[*yy*]*yy*

is a two-digit or four-digit number specifying the year in which *mmm* occurs.

mmm

is a character string specifying the month of [*yy*] *yy*. The options are:

JAN[UARY]	APR[IL]	JUL[Y]	OCT[OBER]
FEB[RUARY]	MAY	AUG[UST]	NOV[EMBER]
MAR[CH]	JUN[E]	SEP[TEMBER]	DEC[EMBER]

T[O-DATE]

causes BATCHCAL to finish the run times display at the end of the specified month.

N[EXT-DATE]

causes BATCHCAL to display the next run time after the beginning of the month specified by FROM-DATE. If you omit FROM-DATE, BATCHCAL displays the next run time after the current time. (The current time is the system date and time on the node where the BATCHCAL process runs.)

S[HOW-DATES]

causes BATCHCAL to display run times from the beginning of the month specified by FROM-DATE to the end of the month specified by TO-DATE. If you omit FROM-DATE, BATCHCAL displays run times from the first run time after the current time. If you omit TO-DATE, BATCHCAL displays run times to the end of the file.

R[E-FORMAT]

causes BATCHCAL to convert a calendar file created by a version of BATCHCAL earlier than C22 to a format compatible with versions C22 through D21. For more information, see Reformatting an Old Calendar File on page 5-22.

BATCHCAL's High PIN Capabilities

Table 5-1 lists the high PIN capabilities of BATCHCAL version D30. BATCHCAL versions earlier than version D20 do not have high PIN capabilities.

Table 5-1. High PIN Capabilities in BATCHCAL (page 1 of 2)

Feature	Implementation	Comments
Can have a high PIN creator?	Yes	None
Can communicate with high PIN requesters?	NA	None
Can communicate with high PIN servers?	NA	None
Can create high PIN processes?	NA	BATCHCAL cannot create processes.

Table 5-1. High PIN Capabilities in BATCHCAL (page 2 of 2)

Feature	Implementation	Comments
Can run at a high PIN?	Yes	None
Can recognize high PIN process IDs?	NA	None
Defaults to run at a high PIN?	Yes	A BATCHCAL process runs at a high PIN by default when all these conditions exist: ● The operating system on the node where the process runs is a D-series system configured to handle high PIN processes. ● The BATCHCAL process creator (for example, a D-series TACL process) can create high PIN processes. ● The BATCHCAL process creator does not force the process to run at a low PIN. ● A high PIN is available in the target CPU. To prevent BATCHCAL from running at a high PIN, specify run option HIGHPIN OFF in the TACL RUN command. For example: `RUN BATCHCAL /HIGHPIN OFF, …/`

Entering Source Data

The BATCHCAL program generates a run calendar from source data supplied in an EDIT input file or during an interactive BATCHCAL session. Source data comprises a number of statements that specify:

- A calendar file name. The file holds up to 10,000 run times.
- A default job start time.
- Information that enables BATCHCAL to calculate run times. BATCHCAL can generate multiple run times for each day specified.

Whether you enter source data in an EDIT file or interactively, it must conform to the structure and format rules in Formatting Source Data on page 5-6.

Entering Source Data in a File Versus Entering It Interactively

The decision to enter a run calendar's source data in a file or interactively depends on:

- The number of source data statements needed to generate the calendar. The minimum number is three—one naming the calendar file, one specifying the default start time, and one specifying a run time or range of run times.
- The likelihood that you will need to regenerate the calendar in the future (for example, when the calendar expires). A calendar generated from source data in a file is easier to regenerate than a calendar generated interactively.

This table describes which source data entry method to use:

If the calendar requires …	And it is…	Then …
Five or more source data statements	Likely you will need to regenerate the calendar	Generate the calendar from source data in an EDIT file
Fewer than five source data statements	Unlikely you will need to regenerate the calendar	Generate the calendar interactively

Formatting Source Data

Source data entered in an EDIT input file or interactively during a BATCHCAL session must comply with this structure and format rules:

```
calendar-file [ ! ] == First noncomment line
default-start-time == Second noncomment line
run-time == Third and subsequent noncomment lines
```

calendar-file [!]

specifies a BATCHCAL calendar file. *calendar-file* must be the first noncomment line entered in the EDIT input file or interactively.

calendar-file

is the name of a disk file to which BATCHCAL writes calendar information. If the file does not exist, BATCHCAL creates it when generating the calendar. If the file exists, it must be a BATCHCAL-generated file (file code 848).

!

specifies, when *calendar-file* exists, that BATCHCAL overwrite the file with new calendar information. If you omit ! and *calendar-file* exists, calendar generation fails with file-system error 10 (file or record already exists).

default-start-time

specifies the default start time for run-time entries that do not specify time. *default-start-time* must be the second noncomment line entered in the EDIT input file or interactively. The form is:

```
[ h] h:[ m] m[:[ s] s]
```

`[ h] h`

is a number in the range 0 through 23 specifying the hour of the day.

`[ m] m`

is a number in the range 0 through 59 specifying the minute of the hour.

`[ s] s`

is a number in the range 0 through 59 specifying seconds. The default is 0.

run-time

specifies when a job using the calendar can or cannot run. *run-time* must be the third noncomment line entered in the EDIT input file or interactively. It also can appear on subsequent noncomment lines. Enter *run-time* in any of these forms:

```
[ + | - ] year month day [ time ]
[ + | - ] / option [ / option ]… [ time ]
[ + | - ] MERGE-DATES calendar-file
[ + | - ] year month day [ time ]
```

`+`

adds the specified run times to the times so far recorded.

`-`

deletes the specified run times from the times so far recorded.

year

specifies the year in any of these forms:

`[ yy] yy`

is a two-digit or four-digit number specifying the year.

`*`

specifies all years.

month

specifies the month in any of these forms:

`[ m] m`

is a number in the range 1 through 12 specifying the month.

mmm

is a character string specifying the month. The options are:

JAN[UARY]	APR[IL]	JUL[Y]	OCT[OBER]
FEB[RUARY]	MAY	AUG[UST]	NOV[EMBER]
MAR[CH]	JUN[E]	SEP[TEMBER]	DEC[EMBER]

*

specifies all months.

day

specifies the day in any of these forms:

[*d*] *d*

is a number in the range 1 through 31 specifying the day of the month.

ddd

is a character string specifying the day. The options are:

MON[DAY]	THU[RSDAY]	SAT[URDAY]
TUE[SDAY]	FRI[DAY]	SUN[DAY]
WED[NESDAY]		

LAST

specifies the last day of the month.

WEEKDAY

specifies all Mondays, Tuesdays, Wednesdays, Thursdays, and Fridays.

WEEKEND

specifies all Saturdays and Sundays.

*

specifies all days.

time

specifies a time that overrides the default start time specified on the first noncomment line. For the format of time, see default-start-time . Omitting + and - makes BATCHCAL replace all previous times recorded for a date with the specified time.

```
[ + | - ] / option [ / option ]… [ time ]
```

`+`

adds the specified run times to the times so far recorded.

`-`

deletes the specified run times from the times so far recorded.

option

specifies a run-time frequency. The options are:

n

specifies a day of the month where *n* is a number in the range 1 through 31. For example, /1 specifies the first day; /15 specifies the fifteenth day; /31 specifies the last day (not necessarily the thirty-first).

*n*M

specifies a frequency in months from the day before the current system date. *n* is a number in the range 1 through 12. For example, /3M specifies a quarterly frequency.

*n*Y

specifies a frequency in years from the day before the current system date. *n* is a number in the range 1 through 11. For example, /5Y specifies a frequency of once every five years.

day

specifies a day of the week where day is the first three characters of the day name; for example, /MON, /TUE. You cannot combine other frequency options with this option if it is the first option.

H[*n*]

specifies a half-yearly frequency occurring at month-end *n* months after each half-year (30 June and 31 December). *n* is a number in the range 1–5. For example, /H specifies a six-monthly frequency occurring at the end of June and December; /H4 specifies the end of October and April.

M

specifies a monthly frequency occurring at month-end (31 January, 28 February [leap years 29 February], 31 March, … 31 December). You can also specify this frequency with option *n*.

mth

specifies the first of the month where *mth* is the first three characters of the month name. For example, /JAN specifies the first of January; /DEC specifies the first of December. You can combine this option with option *n* to specify a day other than the first of the month. For example, /JAN/18 specifies the eighteenth of January.

Q[*n*]

specifies a quarterly frequency occurring at month-end *n* months after the end of each calendar-year quarter (31 March, 30 June, 30 September, and 31 December). *n* can be 1 or 2. For example, /Q specifies a quarterly frequency occurring at the end of March, June, September, and December; /Q2 specifies the end of February, May, August, and November.

Y[*n*]

specifies a yearly frequency occurring at month-end *n* months after year-end (31 December). *n* is a number in the range 1 through 11. For example, /Y specifies a yearly frequency occurring at the end of December; /Y6 specifies the end of June.

+*n*

specifies a frequency in days from the day before the current system date. *n* is a number in the range 1 through 32767. For example, /+7 specifies a frequency of once a week; /+14 specifies a fortnightly frequency. You cannot combine other frequency options with this option if it is the first option.

-*n*

specifies the number of days to subtract from another frequency option. *n* is a number in the range 1 through 32767. For example, /M/-5 specifies a monthly frequency occurring five days before month-end. You cannot specify this option on its own.

You can combine frequency options when you want to specify a frequency you cannot specify by entering an option on its own. For example, there is no single option to specify the last Friday of each month. To specify these Fridays, combine options M, -*n*, and day in the form /M/-6/FRI.

When combining frequency options, you must specify them in order of frequency, with longer frequencies coming before shorter frequencies. For example, year must come before month, month before day of the month, and day of the month before day of the week. The exceptions are +*n* and -*n*, which you can use to qualify any option. You must separate combined options with a diagonal (/).

More examples of option combinations:

/M/MON	Mondays occurring on the last day of the month and, for a month that does not end on a Monday, the first Monday of these months: 1995-01-02, 1995-02-06, 1995-03-06, 1995-04-03, and 1995-05-01
/10/FRI	Fridays occurring on the tenth day of the month and, for a month in which no Friday occurs on the tenth day, the first Friday following the tenth day: 1995-01-13, 1995-02-10, 1995-03-10, 1995-04-14, and 1995-05-12
/5/-7/WED	The Wednesday in the seven-day period before the fifth of the month: 1995-01-04, 1995-02-01, 1995-03-01, 1995-03-29 (for April 1995), and 1995-05-03
/APR/26	The twenty-sixth of April: 1995-04-26, 1996-04-26, 1997-04-26, 1998-04-26, and 1999-04-26

Note. HP recommends that you check the results of frequency-option combinations before committing to your production environment calendars containing run times generated by those combinations.

time

specifies a time that overrides the default start time specified on the first noncomment line. For the format of time, see *default-start-time*. Omitting + and - makes BATCHCAL replace all previous times recorded for a date with the specified time.

```
[ + | - ] MERGE-DATES calendar-file
```

+

adds calendar-file run times to the times so far recorded.

-

deletes calendar-file run times from the times so far recorded.

calendar-file

is the name of a BATCHCAL-generated calendar file (file code 848) whose run times you want to merge with the calendar you are creating.

Omitting + and - makes BATCHCAL replace all previous times recorded for a date with the specified time.

Examples

- This example shows source data entered in an EDIT file:

```
42> FUP COPY WKDYIN
== ************************************************
```

```
== * The source data in this EDIT file will enable *
== * BATCHCAL to generate a run calendar containing *
== * run times for weekdays through 1995. *
== * The calendar will include special run times for *
== * June 30 and December 31 that override the weekday *
== * run times where necessary. It also will exclude *
== * run times on New Year's Day and Christmas Day. *
== **************************************************
$DATA7.CALFILES.WEEKDAY ! == File name of run calendar
22:45 == Default job start time
1995 * WEEKDAY == Run daily Mon-Fri at 22:45
1995 6 30 12:00 == Run at midday on 30JUN95
1995 12 LAST 23:59 == Run at 11:59 p.m. on 31DEC95
- 1995 1 1 == Do not run on 01JAN95
- 1995 12 25 == Do not run on 25DEC95
```

- This example shows source data entered during an interactive BATCHCAL session:

```
> BATCHCAL
1} $DATA7.CALFILES.WEEKEND !
2} 17:30
3} 1995 * WEEKEND
4} EXIT
>
```

- This example shows BATCHCAL generating end-of-quarter run times:

```
> BATCHCAL
1} QUARTERS
2} 12:00
3} /Q
4} EXIT
> BATCHCAL /IN QUARTERS/ SHOW-DATES; FROM-DATE 1995 JAN; TO
-DATE
1995 DEC
\MELBDEV.$DATA7.CALFILES.QUARTERS
1995-03-31 12:00:00
1995-06-30 12:00:00
1995-09-30 12:00:00
1995-12-31 12:00:00
```

- This example shows the merging of run times from an existing calendar file with a new file:

```
> BATCHCAL /IN WEEKENDS/ SHOW-DATES
\MELBDEV.$DATA7.CALFILES.WEEKENDS
1995-11-04 17:30:00
1995-11-05 17:30:00 .
.
> BATCHCAL
1} ALLDAYS !
2} 06:45
3} 1995 NOV WEEKDAY
4} + MERGE-DATES WEEKENDS
5} EXIT
```

```
> BATCHCAL /IN ALLDAYS/ SHOW-DATES
\MELBDEV.$DATA7.CALFILES.ALLDAYS
1995-11-01 06:45:00
1995-11-02 06:45:00
1995-11-03 06:45:00
1995-11-04 17:30:00
1995-11-05 17:30:00 .
.
```

Using BATCHCAL Commands

During an interactive BATCHCAL session, you can use four BATCHCAL commands:

- ==
- EXIT
- FC
- HELP

== Command

To cause BATCHCAL to ignore text from the double equals sign to the end of the command line, use the == (comment) command:

```
== [ comment-text ]
```

`comment-text`

is any text, including braces and square brackets.

EXIT Command

To end an interactive BATCHCAL session, use the EXIT command. The command stops the BATCHCAL process and returns you to the process from which you started the session. (Creating an end-of-file (EOF) condition by pressing CTRL/Y has the same effect as the EXIT command.)

```
E[XIT]
```

FC Command

To redisplay, edit, and reexecute the previous command line, use the FC (Fix Command) command. The command operates similarly to BATCHCOM's FC command. For more information, see FC Command on page 6-92.

```
FC
```

HELP Command

To display information about running the BATCHCAL program and about source data structure and format, use the HELP command.

```
H[ELP]
```

Generating a Run Calendar

To generate a run calendar:

- From source data in an EDIT file, see Generating a Run Calendar From an EDIT Source File on page 5-14
- During an interactive BATCHCAL session, see Generating a Run Calendar During an Interactive BATCHCAL Session on page 5-15

The BATCHCAL program does not let you alter the run times in a run calendar other than by regenerating that calendar.

Generating a Run Calendar From an EDIT Source File

Step 1: Create and Open an EDIT File for the Calendar's Source Data

Create and open an EDIT file for the calendar's source data by using the text editing program EDIT:

```
> EDIT SRC9303 !
CURRENT FILE IS $NB.CALFILES.SRC9303
```

Step 2: Add the Calendar's Source Data

Add the calendar's source data to the file, then close the file and stop the EDIT program:

Note. The data structure and format must conform to the rules in Entering Source Data on page 5-5.

```
*ADD
1 CAL9303 == File ID of run calendar
2 18:15 == Default start time for jobs
3 1993 MARCH WEEKDAY == Run daily Mon-Fri at 18:15
4 1993 MARCH WEEKEND 09:00 == Run Sats and Suns at 09:00
5 - 1993 MARCH 01 == Not run on March 01
6 - 1993 MARCH LAST == Not run on March 31
7 //
*EXIT
63> FILEINFO
$NB.CALFILES
```

```
Code EOF Last Modification …
SRC9303 101 310 22-Dec-92 11:27:24 …
64>
```

Step 3: Generate the Calendar

Run the BATCHCAL program, specifying the calendar's source file as the program's input file:

```
64> BATCHCAL /IN SRC9303/
CAL9303 == File ID of run calendar
18:15 == Default start time for jobs
1993 MARCH WEEKDAY == Run daily Mon-Fri at 18:15
1993 MARCH WEEKEND 09:00 == Run Sats and Suns at 09:00
- 1993 MARCH 01 == Not run on March 01
- 1993 MARCH LAST == Not run on March 31
65> FILEINFO
$NB.CALFILES
Code EOF Last Modification …
CAL9303 848 160 22-Dec-92 11:27:42 …
SRC9303 101 310 22-Dec-92 11:27:24 …
66>
```

Step 4: Display and Check the Calendar's Run Times

Display the calendar's run times and check them. For more information, see Displaying Run Times on page 5-16.

```
> BATCHCAL /IN CAL9303/
1993 - March - CAL9303
-------------------------------------------------------------
|Sunday |Monday |Tuesday |Wednesday |Thursday |Friday |Saturday |
=============================================================
|            | |1 | |2 |18:15 |3 |18:15 |4 |18:15 |5 |18:15 |6 | 9:00 |
|--|------|--|------|--|------|--|------|--|------|--|------|--|------|
|7 | 9:00 |8 |18:15 |9 |18:15 |10|18:15 |11|18:15 |12|18:15 |13| 9:00 |
|--|------|--|------|--|------|--|------|--|------|--|------|--|------|
|14| 9:00 |15|18:15 |16|18:15 |17|18:15 |18|18:15 |19|18:15 |20| 9:00 |
|--|------|--|------|--|------|--|------|--|------|--|------|--|------|
|21| 9:00 |22|18:15 |23|18:15 |24|18:15 |25|18:15 |26|18:15 |27| 9:00 |
|--|------|--|------|--|------|--|------|--|------|--|------|--|------|
|28| 9:00 |29|18:15 |30|18:15 |31|      |  |      |  |      |  |      |
-------------------------------------------------------------
```

Generating a Run Calendar During an Interactive BATCHCAL Session

Step 1: Start a BATCHCAL Session

Start a BATCHCAL session by running the BATCHCAL program:

```
4> BATCHCAL
1}
```

Step 2: Create the Calendar

Enter the calendar's source data at the BATCHCAL prompts, then stop the BATCHCAL program to create the calendar:

Note. The data structure and format must conform to the rules in Entering Source Data on page 5-5.

```
1}CAL9304 == File ID of run calendar
2}10:00 == Default start time for jobs
3}1993 APRIL * == Run daily at 10:00
4}EXIT
> FILEINFO
$DATA7.RUNTIMES
Code EOF Last Modification …
CAL9304 848 222 22-Dec-92 11:41:11 …
>
```

Step 3: Display and Check the Calendar's Run Times

Display the calendar's run times and check them. For more information, see Displaying Run Times on page 5-16.

```
> BATCHCAL /IN CAL9304/
1993 - April - CAL9304
------------------------------------------------------------
|Sunday |Monday |Tuesday |Wednesday|Thursday |Friday |Saturday |
============================================================
| | | | | | | |1 |10:00 |2 |10:00 |3 |10:00 |
|--|------|--|------|--|------|--|------|--|------|--|------|------|
|4 |10:00 |5 |10:00 |6 |10:00 |7 |10:00 |8 |10:00 |9 |10:00 |10|10:00 |
|--|------|--|------|--|------|--|------|--|------|--|------|------|
|11|10:00 |12|10:00 |13|10:00 |14|10:00 |15|10:00 |16|10:00 |17|10:00 |
|--|------|--|------|--|------|--|------|--|------|--|------|------|
|18|10:00 |19|10:00 |20|10:00 |21|10:00 |22|10:00 |23|10:00 |24|10:00 |
|--|------|--|------|--|------|--|------|--|------|--|------|------|
|25|10:00 |26|10:00 |27|10:00 |28|10:00 |29|10:00 |30|10:00 |    | |
------------------------------------------------------------
```

Displaying Run Times

The BATCHCAL program can display run times in list form and in chart form:

- A run-times list is a single column of run times listed one per line in ascending date order. BATCHCAL displays run times in list form when you include the NEXT-DATE or SHOW-DATES program parameter in the RUN BATCHCAL command. For example:

  ```
  > BATCHCAL /IN FEB1993/ SHOW-DATES
  $DATA7.CALFILES.FEB1993
  1993-02-01 12:00:00
  1993-02-02 12:00:00
  1993-02-03 12:00:00 .
  .
  1993-02-26 12:00:00
  ```

```
1993-02-27 12:00:00
1993-02-28 12:00:00
```

- A run-times chart is a table of all run times for a given month. BATCHCAL displays run times in chart form when you omit the NEXT-DATE and SHOW-DATES program parameters from the RUN BATCHCAL command. For example:

```
31> BATCHCAL /IN FEB1993/
1993 - February - FEB1993
-------------------------------------------------------------
|Sunday |Monday |Tuesday |Wednesday|Thursday |Friday |Saturday |
=============================================================================
| | |1 |12:00 |2 |12:00 |3 |12:00 |4 |12:00 |5 |12:00 |6 |12:00 |
|--|------|--|------|--|------|--|------|--|------|--|------|--|------|
|7 |12:00 |8 |12:00 |9 |12:00 |10|12:00 |11|12:00 |12|12:00 |13|12:00 |
|--|------|--|------|--|------|--|------|--|------|--|------|--|------|
|14|12:00 |15|12:00 |16|12:00 |17|12:00 |18|12:00 |19|12:00 |20|12:00 |
|--|------|--|------|--|------|--|------|--|------|--|------|--|------|
|21|12:00 |22|12:00 |23|12:00 |24|12:00 |25|12:00 |26|12:00 |27|12:00 |
|--|------|--|------|--|------|--|------|--|------|--|------|--|------|
|28|12:00 |  |      |  |      |  |      |  |      |  |      |  |      |
-------------------------------------------------------------
```

A plus sign (+) appended to a run time indicates multiple run times exist on that day. To display all run times for the day, display a run-times list using the NEXT-DATE or SHOW-DATES program parameter in the RUN BATCHCAL command.

Displaying Run Times in List Form

Table 5-2 on page 5-17 lists the procedures for displaying run times in list form. The examples relate to the calendar generated by the source data CALQ393:

```
21:30
1993 APRIL 01 09:30
1993 APRIL LAST
1993 MAY 01 09:30
1993 MAY LAST
1993 JUNE 01 09:30
1993 JUNE LAST
```

Table 5-2. Displaying Run Times in List Form (page 1 of 2)

To display the next run time after the current time...	**Example**
BATCHCAL / IN *calendar-file* / N[EXT-DATE]	> **TIME** January 12, 1993 14:44:31 > **BATCHCAL /IN CALQ393/ NEXT-DATE** 1993-04-01 09:30:00
To display the next run time after the beginning of a specified month ...	**Example**
BATCHCAL / IN *calendar-file* / N[EXT-DATE] ; F[ROM-DATE] [*yy*]*yy mmm*	> **BATCHCAL /IN CALQ393/ NEXT-DATE; FROM-DATE 1993 JUNE**1993-06-01 09:30:00

Table 5-2. Displaying Run Times in List Form (page 2 of 2)

To display all run times after the current time …	**Example**
BATCHCAL / IN *calendar-file* /S[HOW-DATES]	> **TIME** January 12, 1993 14:48:54 > **BATCHCAL /IN CALQ393/ SHOW-DATES $DATA7.RUNTIMES.CALQ393** 1993-04-01 09:30:00 1993-04-30 21:30:00 1993-05-01 09:30:00 1993-05-31 21:30:00 1993-06-01 09:30:00 1993-06-30 21:30:00
To display all run times after the beginning of a specified month …	**Example**
BATCHCAL / IN *calendar-file* / S[HOW-DATES] ; F[ROM-DATE] [*yy*]*yy mmm*	> **BATCHCAL /IN CALQ393/ SHOW-DATES; FROM-DATE 1993 MAY** $DATA7.RUNTIMES.CALQ393 1993-05-01 09:30:00 1993-05-31 21:30:00 1993-06-01 09:30:00 1993-06-30 21:30:00
To display all run times in a specified range of months...	**Example**
BATCHCAL / IN *calendar-file* / S[HOW-DATES] ; F[ROM-DATE] [*yy*]*yy mmm* ; T[O-DATE] [*yy*]*yy mmm*	> BATCHCAL /IN CALQ393/ SHOW-DATES; FROM-DATE 1993 APRIL; TO-DATE 1993 MAY $DATA7.RUNTIMES.CALQ393 1993-04-01 09:30:00 1993-04-30 21:30:00 1993-05-01 09:30:00 1993-05-31 21:30:00
To display all run times from the current time to the end of a specified month …	**Example**
BATCHCAL / IN *calendar-file* / S[HOW-DATES] ; T[O-DATE] [*yy*]*yy mmm*	> **TIME** January 12, 1993 14:58:37 61> **BATCHCAL /IN CALQ393/ SHOW-DATES; TO-DATE 1993 APRIL** $DATA7.RUNTIMES.CALQ393 1993-04-01 09:30:00 1993-04-30 21:30:00

Displaying Run Times in Chart Form

Table 5-3 on page 5-19 lists the procedures for displaying run times in chart form. The examples relate to the calendar generated by this source data:

```
CALQ493
12:00
1993 JULY WEEKDAY
1993 JULY WEEKEND 06:45
1993 AUGUST WEEKDAY 17:45
1993 AUGUST WEEKEND
1993 SEPTEMBER WEEKDAY 22:30
1993 SEPTEMBER WEEKEND 03:15
```

The BATCHCAL program pauses indefinitely after each month when diplaying time in chart form. To continue the display, press RETURN. To end the display, press CTRL/Y.

Table 5-3. Displaying Run Times in Chart Form (page 1 of 3)

To display all run times after the current time...	Example
BATCHCAL / IN *calendar-file* /	> TIME January 13, 1993 11:49:55 > BATCHCAL /IN CALQ493/ 1993 - July - CALQ493 ---------- |Sunday |Monday |Tuesday |Wednesday|Thursday |Friday |Saturday | ========== | | | | | | | |1 |12:00 |2 |12:00 |3 | 6:45 | |--|---|--|---|--|---|--|---|--|---|--|---|--|--- . ---------- 1993 - August - CALQ493 ---------- |Sunday |Monday |Tuesday |Wednesday|Thursday |Friday |Saturday | ========== |1 |12:00 |2 |17:45 |3 |17:45 |4 |17:45 |5 |17:45 |6 |17:45 |7 |12:00 | |--|---|--|---|--|---|--|---|--|---|--|---|--|--- . ---------- 1993 - September - CALQ493 ---------- |Sunday |Monday |Tuesday |Wednesday|Thursday |Friday |Saturday | ========== | | | | | | |1 |22:30 |2 |22:30 |3 |22:30 |4 | 3:15 | |--|--- |--|---|--|---|--|---|--|---|--|---|--|--- . ----------

Table 5-3. Displaying Run Times in Chart Form (page 2 of 3)

To display all run times after the beginning of a specified month...	Example
BATCHCAL / IN *calendar-file* / F[ROM-DATE] [*yy*]*yy mmm*	> BATCHCAL /IN CALQ493/ FROM-DATE 1993 AUGUST 1993 - August - CALQ493 ---------- |Sunday |Monday |Tuesday |Wednesday|Thursday |Friday |Saturday | ========== |1 |12:00 |2 |17:45 |3 |17:45 |4 |17:45 |5 |17:45 |6 |17:45 |7 |12:00 | |--|--------|--|----------|--|---------|--|------------|--|---------|--|---------|- . ---------- 1993 - September - CALQ493 ---------- |Sunday |Monday |Tuesday |Wednesday|Thursday |Friday |Saturday | ========== | | | | | |1 |22:30 |2 |22:30 |3 |22:30 |4 | 3:15 | |--|--------|--|----------|--|---------|--|------------|--|---------|--|--------|- . ----------

Table 5-3. Displaying Run Times in Chart Form (page 3 of 3)

To display all run times in a specified range of months...	Example
BATCHCAL / IN *calendar-file* / F[ROM-DATE] [*yy*]*yy mmm* ; T[O-DATE] [*yy*]*yy mmm*	> BATCHCAL /IN CALQ493/ FROM-DATE JULY; TO-DATE AUGUST 1993 - July - CALQ493 ---------- |Sunday |Monday |Tuesday |Wednesday|Thursday |Friday |Saturday | ========== | | | | | | | |1 |12:00 |2 |12:00 |3 | 6:45 | |--|--------|--|--------|--|---------|--|----------|--|--------|--|-------|--|---- . ---------- 1993 - August - CALQ493 ---------- |Sunday |Monday |Tuesday |Wednesday|Thursday |Friday |Saturday | ========== |1 |12:00 |2 |17:45 |3 |17:45 |4 |17:45 |5 |17:45 |6 |17:45 |7 |12:00 |--|--------|--|---------|--|-------|--|-----------|--|-------|--|--------|--|---- . ----------

To display all run times from the current time to the end of a specified month...	Example
BATCHCAL / IN *calendar-file* / T[O-DATE] [*yy*]*yy mmm*	> TIME January 13, 1993 12:12:50 > BATCHCAL /IN CALQ493/ TO-DATE 1993 JULY 1993 - July - CALQ493 ---------- |Sunday |Monday |Tuesday |Wednesday|Thursday |Friday |Saturday | ========== | | | | | | | |1 |12:00 |2 |12:00 |3 | 6:45 | |--|--------|--|---------|--|---------|--|----------|--|--------|--|-------|--|---- . ----------

Reformatting an Old Calendar File

Versions of the BATCHCAL program from C22 through D21 cannot read calendar files created by earlier versions of the program. An attempt to read such a file fails with this message:

```
Old calendar should be rebuilt or converted by RE-FORMAT
```

To convert an old calendar file to the current format, use the RE-FORMAT program parameter in the RUN BATCHCAL command (see Table 5-4). The scheduler and the D30 version of BATCHCAL can read all calendar files, regardless of when they were created.

Table 5-4. Reformatting an Old Calendar File

To reformat a pre-C22 calendar file...	Example
BATCHCAL / IN *calendar-file* / R[E-FORMAT]	> **FILEINFO CALOLD** $DATA7.CALENDAR Code EOF Last Modification … CALOLD 848 2038 21-Nov-89 02:27:48 … > **BATCHCAL /IN CALOLD/ NEXT-DATE** ?Old calendar should be rebuilt or converted by RE-FORMAT > **BATCHCAL /IN CALOLD/ RE-FORMAT** > **BATCHCAL /IN CALOLD/ NEXT-DATE** 1993-01-31 12:00:00 > **FILEINFO CALOLD** $DATA7.CALENDAR Code EOF Last Modification … CALOLD 848 674 13-Jan-93 15:27:01 …

6 Commands

This section describes how to use BATCHCOM commands:

Topic	Page
Running BATCHCOM	6-1
BATCHCOM's High PIN Capabilities	6-6
Command Reference Summary	6-11
Command Security	6-23
Command Descriptions	6-28

Running BATCHCOM

Use the TACL RUN command to run the BATCHCOM program and start a BATCHCOM session. A session lasts until the program stops.

There are two types of BATCHCOM sessions:

- Interactive (also known as conversational), during which BATCHCOM responds to each command you type at its prompt (a single, closing brace)
- Noninteractive, during which BATCHCOM automatically executes commands specified in an input file or in the TACL RUN command

This subsection describes TACL RUN command syntax as it applies to running BATCHCOM. Included are brief discussions of factors to consider before you run BATCHCOM. Examples illustrate various run options and program parameters.

```
[ RUN ] [ \node. ] [ volume. ] BATCHCOM
[ /run-option [ , run-option ]… / ]
[ [ \node. ] $schd ] [ , SYNTAX ] [ ; command-list ]
```

node

is the name of the node where the BATCHCOM program file resides. If the node is remote, you must have remote access to it (that is, you must have remote passwords set up on both local and remote nodes). The default is the node where the process that creates the BATCHCOM process is running.

volume

is one of:

$*volume-name.subvolume-name*

specifies the volume and subvolume containing the BATCHCOM program file.

subvolume-name

specifies the subvolume containing the BATCHCOM program file. The default volume varies, depending on whether the RUN command is explicit or implicit.

- For an explicit RUN command, enter RUN followed by the BATCHCOM program-file name. If you enter a partially qualified file name, the TACL program expands it by using the current default node, volume, and subvolume names.
- For an implicit RUN command, enter the BATCHCOM program-file name only. If you enter a partially qualified file name, the TACL program searches for the file in the subvolumes specified by the TACL #PMSEARCHLIST variable.

run-option

is one of these TACL RUN command options.

CPU	HIGHPIN	INV	NAME	PFS	TERM
DEBUG	IN	JOBID	NOWAIT	PRI	WINDOW
DEFMODE	INLINE	LIB	OUT	STATUS	
EXTSWAP	INSPECT	MEM	OUTV	SWAP	

For descriptions of these options, see the RUN[D] command description in the *TACL Reference Manual*.

schd

is a scheduler process name. If you omit *schd*, the first scheduler-related command BATCHCOM opens scheduler process $ZBAT if it exists. If $ZBAT does not exist, the command fails with a message similar to:

```
0014-E Device does not exist, \MELBDEV.$ZBAT
```

When BATCHCOM runs as a job's executor-program process, the default scheduler is the scheduler controlling the job.

SYNTAX

causes BATCHCOM to check the syntax of, but not execute, the commands in *command-list* or in the file specified by *run-option* IN. When you specify SYNTAX, you also must specify $ *schd* unless scheduler process $ZBAT exists.

command-list

is a list of semicolon-separated BATCHCOM commands.

Considerations

- BATCHCOM maintains a working-attributes set that supplies default attributes to attachment sets, classes, executors, jobs, and schedulers added during a session.

You can manipulate the set's contents by using the SET, SHOW, and RESET commands:

- SET commands add attributes to or alter the values of attributes in the set.
- SHOW commands display attributes in the set.
- RESET commands remove attributes from the set except these defaults:

Object	Default Attributes
ATTACHMENT-SET	DEFINE =_DEFAULTS specifying the node and volume set by the last SET ATTACHMENT-SET (DEFINE =_DEFAULTS, VOLUME …), SYSTEM, or VOLUME command. If there was no such command, the DEFINE specifies the current node and volume from the TACL environment.
CLASS	INITIATION ON class attribute.
EXECUTOR	None
JOB	JOBID-ZERO OFF and VOLUME job attributes. The VOLUME attribute specifies the node, volume, subvolume, and security defaults set by the last SET JOB VOLUME command. If there was no such command, the attribute specifies the current defaults from the TACL environment.
SCHEDULER	None

- BATCHCOM's initial working-attributes set contains attachment-set attributes inherited from the TACL working set. These attributes include all ASSIGNs and PARAMs from the TACL set, and all DEFINEs except those whose names begin with =_ZBAT. (The NetBatch product reserves DEFINEs whose names begin with =_ZBAT for use by its own processes.)

  The first online help request during a session causes BATCHCOM to open one of these IMMU files containing NetBatch help text:

  Note. IMMU (Information Message Management Utilities) is an HP internal product used for handling message and help text, and keyword substitution. It consists of a utility that loads and unloads key-sequenced message tables and the procedures needed to access those tables.

  - If map DEFINE =_ZBAT_IMMU_FILE exists in the TACL working set, BATCHCOM opens the file specified by that DEFINE. To add the DEFINE to the TACL set before starting a session, use TACL's ADD DEFINE command. For example:

    ```
    5> ADD DEFINE =_ZBAT_IMMU_FILE, CLASS MAP, FILE NB.HELP
    ```

    Note. BATCHCOM does not propagate DEFINEs whose names begin with =_ZBAT from the TACL working set to its own working-attributes set. Therefore, you cannot change the file specified by DEFINE =_ZBAT_IMMU_FILE after you start a session.

  - If map DEFINE =_ZBAT_IMMU_FILE does not exist in the TACL working set, BATCHCOM opens file BATCHIMU in the BATCHCOM program-file

subvolume. (This file contains the standard help text supplied with NetBatch software.) When BATCHIMU is not in the program-file subvolume, the help request fails, and BATCHCOM displays a message similar to this:

```
0011-E File or record does not exist, BATCHIMU
```

Examples

- The explicit RUN command in this example starts an interactive BATCHCOM session. The STATUS SCHEDULER command, being the first scheduler-related command after the session begins, opens scheduler $ZBAT by default and displays its details.

```
> RUN $SYSTEM.SYSTEM.BATCHCOM
BATCHCOM - T9190D30 …
1} STATUS SCHEDULER
NETBATCH SERVER - T9190D30 … Time: 26AUG94 10:05:08
SCHEDULER STATUS
Process : \MELBDEV.$ZBAT Primary : 0,45 Backup : 1,77 .
2}
```

- The implicit RUN command in this example starts an interactive BATCHCOM session and causes BATCHCOM to open scheduler $SCHD on node \MELBQAT:

```
> BATCHCOM \MELBQAT.$SCHD
BATCHCOM - T9190D30 …
NETBATCH SERVER - T9190C23 … Time: 26AUG94 10:10:07
1}
```

- The implicit RUN command in this example starts a noninteractive BATCHCOM session during which BATCHCOM executes a STATUS SCHEDULER command. Because the RUN command does not specify a scheduler, the STATUS command opens and displays details of scheduler $ZBAT.

```
> BATCHCOM; STATUS SCHEDULER
SCHEDULER STATUS
Process : \MELBDEV.$ZBAT Primary : 0,45 Backup : 1,77 .
```

- The implicit RUN command in this example is the same as in the previous example, but the STATUS SCHEDULER command fails because $ZBAT is not running:

```
> BATCHCOM; STATUS SCHEDULER
STATUS SCHEDULER
-^-0014-
E Device does not exist, \MELBDEV.$ZBAT
```

- The implicit RUN command in this example causes BATCHCOM to check, but not execute, the two commands in file STATUS. The first command is syntactically correct, but the second contains a spelling error.

```
> BATCHCOM /IN STATUS/ \MELBQAT.$SCHD, SYNTAX
BATCHCOM - T9190D30 …
NETBATCH SERVER - T9190C23 … Time: 26AUG94 10:26:04
STATUS EXECUTOR *
```

```
STTUS JOB *
STTUS JOB *
^--
0290-E Invalid command
```

- The implicit RUN command in this example causes BATCHCOM to execute the OPEN, SUBMIT JOB, and INFO JOB commands in the file BACKUP:

```
> BATCHCOM /IN BACKUP/
BATCHCOM - T9190D30 …
OPEN \MELBQAT.$SCHD
NETBATCH SERVER - T9190C23 … Time: 26AUG94 10:29:26
SUBMIT JOB BACKUP, IN $A.OPS.BACKUP, OUT $S.#OPS, AFTER 17:30
Job BACKUP Jobnumber 7 submitted
INFO JOB BACKUP
JOB ATTRIBUTES for BACKUP
jobnumber: 7
volume: $A.OPS, "NCNC"
in: \MELBDEV.$A.OPS.BACKUP
out: \MELBDEV.$S.#OPS
executor-program: \MELBDEV.$SYSTEM.SYSTEM.TACL
pri: 120
selpri: 3
maxprintlines: None
maxprintpages: None
class: STANDARD-CLASS
stall: Off
stop-on-abend: Off
after: 17:30:00
user: 255,255
next-runtime: 26AUG94 17:30:00
```

- This example shows ASSIGNs, DEFINEs, and PARAMs from the TACL working set becoming attachment-set attributes in BATCHCOM's initial working-attributes set. BATCHCOM does not propagate the DEFINE =_ZBAT_IMMU_FILE to the set.

```
> ASSIGN CREDITS-FILE, $DATA6.ACCOUNTS.CREDITS
> PARAM DATE 26AUG94
> ADD DEFINE =_ZBAT_IMMU_FILE, CLASS MAP, FILE NB.HELP
> ALTER DEFINE =_DEFAULTS, VOLUME $A.NB
> BATCHCOM; SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
ASSIGN CREDITS-FILE, $DATA6.ACCOUNTS.CREDITS
PARAM DATE 26AUG94
```

- The SHOW commands in this example display the contents of BATCHCOM's working-attributes set at the start of a session. Note the source of attachment-set attributes (the TACL working set) and the absence of executor and scheduler attributes. The value of the VOLUME job attribute comes from the TACL environment's current defaults.

```
> ASSIGN
BACKUP-FILE
```

```
Physical file: $BIG1.OPS.BACKUP
> INFO DEFINE **
Define Name =_DEFAULTS
CLASS DEFAULTS
VOLUME $A.NB
> PARAM
DEPT .OPERATIONS.
SHIFT .DAY.
HOURS .0900-1700.
> WHO .
Current volume: $A.NB .
Userid: 133,2 Username: NB.USER Security: "AAAA"
> BATCHCOM; SHOW ATTACHMENT-SET; SHOW CLASS; SHOW EXECUTOR;
SHOW JOB; SHOW SCHEDULER
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
ASSIGN BACKUP-FILE, $BIG1.OPS.BACKUP
PARAM DEPT OPERATIONS
PARAM SHIFT DAY
PARAM HOURS 0900-1700
CLASS ATTRIBUTE
initiation: On
EXECUTOR ATTRIBUTES
JOB ATTRIBUTES
volume: \MELBDEV.$A.NB, "NNNN"
user: 133,2
SCHEDULER ATTRIBUTES
```

BATCHCOM's High PIN Capabilities

Table 6-1 lists the high PIN capabilities of BATCHCOM versions D20 and later. BATCHCOM versions earlier than D20 do not have high PIN capabilities.

Table 6-1. High PIN Capabilities in BATCHCOM (page 1 of 2)

Feature	Implementation	Comments
Can have a high PIN creator?	Yes	None
Can communicate with high PIN requesters?	Yes	None
Can communicate with high PIN servers?	Yes	None
Can create high PIN processes?	Yes	None

Table 6-1. High PIN Capabilities in BATCHCOM (page 2 of 2)

Feature	Implementation	Comments
Can run at a high PIN?	Yes	None
Can recognize high PIN process IDs?	Yes	None
Defaults to run at a high PIN?	Yes	A BATCHCOM process runs at a high PIN by default when all these conditions exist: ● The operating system on the node where the process runs is a D-series system configured to handle high PIN processes. ● The BATCHCOM process creator (for example, a D-series TACL process) can create high PIN processes. ● The BATCHCOM process creator does not force the process to run at a low PIN. ● A high PIN is available in the target CPU. To prevent BATCHCOM from running at a high PIN, specify run option HIGHPIN OFF in the TACL RUN command. For example: `RUN BATCHCOM /HIGHPIN OFF, …/ …`

Keywords

This subsection describes and lists BATCHCOM keywords, explains the rules applying to keyword abbreviation, and lists keyword and command aliases.

Keyword Types

The five types of keywords are: object keywords, command keywords, command qualifier keywords, attribute keywords, and attribute flag keywords.

Object Keywords

An object keyword identifies the object of a command. Object keywords are:

ATTACHMENT-SET	EXECUTOR	SCHEDULER
CLASS	JOB	

The default object keyword when you start a BATCHCOM session is JOB.

Command Keywords

A command keyword specifies a task BATCHCOM is to perform. Command keywords are:

ABORT	HISTORY	START
ACTIVATE	INFO	STATUS
ADD	OBEY	STATUS-HISTORY
ALLOW	OPEN	STOP
ALTER	RELEASE-WAITON	SUBMIT
ASSUME	RESET	SUSPEND
CHANGEUSER	RUN	SWITCHCPU
COMMENT	RUNNEXT	SWITCHLOG
DELETE	RUNNOW	SYSTEM
DISPLAY-SPI	SET	VOLUME
EXIT	SHOW	!
FC	SHUTDOWN	?
HELP		

Command Qualifier Keywords

A command qualifier keyword modifies the meaning of a command (for example, by specifying the format of or restricting its output). Command qualifier keywords are:

AFTER	LIB	OWNER
ALL	LIKE	REMOVE
ATTACHMENT-SET	NB-LOG	RUN
CLASS	NOT	SELPRI
DETAIL	OBEY-FORM	STATE
ERRORS	OFF	USER
FROM	ON	WAITON
IN	OUT	#CURRENT
JB-LOG		

Attribute Keywords

An attribute keyword specifies a characteristic of an attachment set, class, executor, job, or scheduler. Attribute keywords are:

AFTER	EMS	OUT
ASSIGN	EVERY	PARAM
AT	EXECUTOR-PROGRAM	PFS
AT-ALLOWED	EXTSWAP	PRI
ATTACHMENT-SET	HIGHPIN	PURGE-IN-FILE
BACKUPCPU	HOLD	RESTART
CALENDAR	HOLDAFTER	RUND
CATCHUP	IFFAILS	SAVEABEND
CLASS	IN	SECURITY
CPU	INITIATION	SELPRI
DEFAULT-CLASS	JOB-LOG	STALL
DEFAULT-EXECUTOR-PROGRAM	JOBID-ZERO	STARTUP
DEFAULT-HIGHPIN	LIB	STOP-ON-ABEND
DEFAULT-MAXPRINTLINES	LIMIT	SUBMIT-ALLOWED
DEFAULT-MAXPRINTPAGES	LOCALNAMES	SWAP
DEFAULT-OUT	MAX-CONCURRENT-JOBS	TAPEDRIVES
DEFAULT-PRI	MAX-PRI	TEMPORARY
DEFAULT-SELPRI	MAX-PRINTLINES	TERM
DEFAULT-STALL	MAX-PRINTPAGES	VOLUME
DEFAULT-STOP-ON-ABEND	MEM	WAIT
DEFINE-DESCRIPTION	NAME	WAITON

Attribute Flag Keywords

An attribute flag keyword sets an attribute flag such as ON or OFF. Attribute flag keywords are:

NONE	OFF	ON

Keyword Abbreviations

BATCHCOM lets you abbreviate its keywords, thereby reducing the keystrokes necessary to enter commands. As a general rule, you can abbreviate any keyword by omitting characters from right to left. The resulting abbreviation, however, must be unique. For example, the acceptable abbreviations of keywords ASSIGN and ASSUME are ASSI and ASSU. The exceptions to the rule and other considerations are:

- You can abbreviate the object keywords ATTACHMENT-SET, CLASS, EXECUTOR, and SCHEDULER, but only when the preceding keyword is ASSUME. For example, you could abbreviate SCHEDULER to SC in the ASSUME SCHEDULER command (ASSU SC). In the STATUS SCHEDULER command, however, you must enter SCHEDULER in full (S SCHEDULER). You cannot abbreviate the object keyword JOB.
- You can abbreviate each word in a hyphen-separated keyword, as long as the resulting abbreviation is unique and retains all hyphens. For example, you could abbreviate DEFAULT-STOP-ON-ABEND to D---.
- You can abbreviate the names of ASSIGN attributes, but not those of DEFINE attributes. For example, you could abbreviate the ASSIGN attribute names I-O and SHARED to I- and S.
- Single-character abbreviations of frequently used command keywords are:

Abbreviation	Keyword
A	ADD
D	DELETE
E	EXIT
F	FC
H	HELP
I	INFO
O	OBEY
S	STATUS
V	VOLUME

- For information on abbreviating specific keywords, see the syntax diagrams in this section and in Section 7, Attributes.

Examples

Each of these examples shows a BATCHCOM command and its abbreviated form:

```
ALTER JOB 35, EXECUTOR-PROGRAM COBOL85, AFTER 18:45, HOLD OFF
ALT JOB 35, E-P COBOL85, AF 18:45, HOLD OF
SUBMIT JOB X, IN TEMP, OUT $S, PURGE-IN-FILE ON, HOLDAFTER ON
SUB JOB X, IN TEMP, OU $S, P-- ON, HOLDA ON
ALTER SCHEDULER, DEFAULT-MAXPRINTPAGES 100, AT-ALLOWED OFF
ALT SCHEDULER, D-MAXPRINTP 100, AT-A OF
ADD EXECUTOR X-0, CPU 0, CLASS *; START EXECUTOR X-0
A EXECUTOR X-0, CP 0, CL *; START EXECUTOR X-0
SHOW ATTACHMENT-SET /OUT Y/ DEFINE, PARAM, OBEY-FORM
SHO ATTACHMENT-SET /OU Y/ DEFI, PA, O-
```

Keyword and Command Aliases

Aliases exist for some BATCHCOM keywords and commands:

Alias	Command	Comments
== (double equals)	COMMENT	
ADD	SUBMIT	ADD JOB and ADD-ALLOWED are acceptable variants of the SUBMIT JOB command and SUBMIT-ALLOWED scheduler attribute.
ADP (abbreviation of ASSIGNs, DEFINEs, PARAMs)	ATTACHMENT-SET	For example, you could enter the command SHOW ADP instead of SHOW ATTACHMENT-SET.
JOBCLASS	CLASS	
S-M	Attribute keyword STARTUP	S-M is an abbreviation of STARTUP-MESSAGE.
S[TATUS]-E[XECUTING]	STATUS JOB (STATE EXECUTING)	
S[TATUS]-R[EADY]	STATUS JOB (STATE READY)	
S[TATUS]-U[SER] [*user-ID*]	STATUS JOB (U[SER] *user-ID*)	

Command Reference Summary

This section includes the syntax of the TACL RUN commands that run the NETBATCH and BATCHCOM programs.

Command to Run NETBATCH

```
[ RUN ] [ \node. ] [ volume. ] NETBATCH
/ NAME [ $schd ] , NOWAIT [ , run-option ]… /
[ schd-database-subvol ] [ ! ] [ \remote-node ]… [ EMS ]
[DISPLAY-SPI ]
```

Command to Run BATCHCOM

```
[ RUN ] [ \node. ] [ volume. ] BATCHCOM
[ /run-option [ , run-option ]… / ]
[ [ \node. ] $schd ] [ , SYNTAX ] [ ; command-list ]
```

Attachment-Set Commands

- ADD ATTACHMENT-SET adds attachment sets to a scheduler:

```
A[DD] [ ATTACHMENT-SET ] [ / OU[T] [ file-name ] / ]
{( user-ID ) ] attachment-set-name |#CURRENT) }
[ , attribute ]…
```

attribute is one of:

```
( ASSI[GN] ASSIGN-name , ASSIGN-attributes )
( DEFI[NE] DEFINE-name-1 , [ LIK[E] DEFINE-name-2 , ]
[ CLASS DEFINE-class , ]
[, DEFINE-attribute [ , DEFINE-attribute ]… )
( PA[RAM] PARAM-name PARAM-value )
SEC[URITY] " security "
TEM[PORARY] {OF[F]|ON) }
```

- ALTER ATTACHMENT-SET changes attachment-set attributes:

```
ALT[ER] [ ATTACHMENT-SET ] [ / OU[T] [ file-name ] / ]
{ [ ( user-ID ) ] attachment-set-ID |#CURRENT}
, attribute [ , attribute ]…
```

For a description of *attribute*, see ADD ATTACHMENT-SET Command on page 6-34.

- ASSUME ATTACHMENT-SET sets the default object keyword to ATTACHMENT-SET:

```
ASSU[ME] A[TTACHMENT]-S[ET]
```

- DELETE ATTACHMENT-SET deletes attachment sets and deletes specified ASSIGNs, DEFINEs (except =_DEFAULTS), and PARAMs from attachment sets:

```
D[ELETE] [ ATTACHMENT-SET ] [ / OU[T] [ file-name ] / ]
[ ( user-ID ) ] attachment-set-IDB
[ [ attribute ]…| , DET[AIL] ]
```

attribute is one of:

```
ASSI[GN] [ ASSIGN-name ]
DEFI[NE] [ DEFINE-name ]
PA[RAM] [ PARAM-name ]
```

- INFO ATTACHMENT-SET lists the attributes of attachment sets:

```
I[NFO] [ ATTACHMENT-SET ] [ / OU[T] [ file-name ] / ]
{ [ ( user-ID ) ] attachment-set-ID |#CURRENT }
[ [ , attribute ]…|, DET[AIL] ] [ , O[BEY]-[FORM] ]
```

attribute is one of:

```
ASSI[GN] [ ASSIGN-name ]
DEFI[NE] [ DEFINE-name ]
PA[RAM] [ PARAM-name ]
SEC[URITY]
TEM[PORARY]
```

- RESET ATTACHMENT-SET removes all attachment-set attributes (except DEFINE =_DEFAULTS) from BATCHCOM's working-attributes set:

  ```
  RESE[T] [ ATTACHMENT-SET ] [ attribute [ , attribute ]… ]
  ```

  For a description of *attribute*, see INFO ATTACHMENT-SET Command on page 6-99.

- SET ATTACHMENT-SET specifies attachment-set attributes in BATCHCOM's working-attributes set:

  ```
  SET [ ATTACHMENT-SET ] attribute [ , attribute ]…
  ```

  For a description of *attribute*, see ALTER ATTACHMENT-SET Command on page 6-52.

- SHOW ATTACHMENT-SET lists attachment-set attributes from BATCHCOM's working-attributes set:

  ```
  SHO[W] [ ATTACHMENT-SET ] [ / OU[T] [ file-name ] / ]
  [ attribute [ , attribute ]… ] [ , O[BEY]- [FORM] ]
  [O[BEY]-[FORM] ]
  ```

  For a description of *attribute*, see INFO ATTACHMENT-SET Command on page 6-99.

- STATUS ATTACHMENT-SET lists attachment sets and the names and owners of jobs using those sets:

  ```
  S[TATUS] [ ATTACHMENT-SET ] [ / OU[T] [ file-name ] / ]
  { [ ( user-ID ) ] attachment-set-name | #CURRENT }
  ```

Class Commands

- ADD CLASS adds classes to a scheduler:

  ```
  A[DD] [ CLASS ] [ / OU[T] [ file-name ] / ]
  class-name [ , attribute ]
  ```

  attribute is:

  ```
  INI[TIATION] { OF[F] | ON }
  ```

- ALTER CLASS changes class attributes:

  ```
  ALT[ER] [ CLASS ] [ / OU[T] [ file-name ] / ]
  class-name , attribute
  ```

  For a description of *attribute*, see ADD CLASS Command on page 6-40.

- ASSUME CLASS sets the default object keyword to CLASS:

  ```
  ASSU[ME] CL[ASS]
  ```

- DELETE CLASS deletes classes from a scheduler:

  ```
  D[ELETE] [ CLASS ] [ / OU[T] [ file-name ] / ] class-name
  ```

- INFO CLASS lists the attributes of classes:

```
I[NFO] [ CLASS ] [ / OU[T] [ file-name ] / ]
class-ID [ , O[BEY]-[FORM] ]
```

- RESET CLASS restores the default value of the INITIATION class attribute in BATCHCOM's working-attributes set:

```
RESE[T] [ CLASS ]
```

- SET CLASS specifies the value of the INITIATION class attribute in BATCHCOM's working-attributes set:

```
SET [ CLASS ] attribute
```

  For a description of *attribute*, see ADD CLASS Command on page 6-40.

- SHOW CLASS displays the INITIATION class attribute from BATCHCOM's working-attributes set:

```
SHO[W] [ CLASS ] [ / OU[T] [ file-name ] / ]
[ O[BEY]-[FORM] ]
```

Executor Commands

- ADD EXECUTOR adds executors to a scheduler:

```
A[DD] [ EXECUTOR ] [ / OU[T] [ file-name ] / ]
executor-name-1 [ , LIK[E] executor-name-2 ]
[ , attribute ]…
```

  attribute is one of:

```
CL[ASS] BBC{( AALVS1( class-name,( class-name [ , class-name
]… ),*) )
CP[U] cpu-number
```

- ALTER EXECUTOR changes executor attributes:

```
ALT[ER] [ EXECUTOR ] [ / OU[T] [ file-name ] / ]
executor-name-1 ,
{ LIK[E] executor-name-2 [ , attribute ]… }
{ attribute [ , attribute ] }
```

  For a description of *attribute*, see ADD EXECUTOR Command on page 6-42.

- ASSUME EXECUTOR sets the default object keyword to EXECUTOR:

```
ASSU[ME] EXE[CUTOR]
```

- DELETE EXECUTOR deletes executors from a scheduler:

```
D[ELETE] [ EXECUTOR ] [ / OU[T] [ file-name ] / ]
executor-name
```

- INFO EXECUTOR lists the attributes of executors:

```
I[NFO] [ EXECUTOR ] [ / OU[T] [ file-name ] / ]
executor-ID [ [ , attribute ]… | , O[BEY]-[FORM] ]
```

attribute is one of:

```
CL[ASS]
CP[U]
```

- RESET EXECUTOR removes executor attributes from BATCHCOM's working-attributes set:

  ```
  RESE[T] [ EXECUTOR ] [ attribute [ , attribute ] ]
  ```

  For a description of *attribute*, see INFO EXECUTOR Command on page 6-104.

- SET EXECUTOR specifies executor attributes in BATCHCOM's working-attributes set:

  ```
  SET [ EXECUTOR ] [ LIK[E] executor-name , ]
  attribute [ , attribute ]
  ```

  For a description of *attribute*, see ADD EXECUTOR Command on page 6-42.

- SHOW EXECUTOR lists executor attributes from BATCHCOM's working-attributes set:

  ```
  SHO[W] [ EXECUTOR ] [ / OU[T] [ file-name ] / ]
  [ attribute [ , attribute ] [ , O[BEY]-[FORM] ]
  [O[BEY]-[FORM]]
  ```

  For a description of *attribute*, see INFO EXECUTOR Command on page 6-104.

- START EXECUTOR starts executors whose state is OFF or STOP, thus making the executors available for use by jobs:

  ```
  START [ EXECUTOR ] [ / OU[T] [ file-name ] / ] executor-ID
  ```

- STATUS EXECUTOR displays executors' names, CPUs, and states. If an executor is in use by a job, the command also displays the job's number and its class name:

  ```
  S[TATUS] [ EXECUTOR ] [ / OU[T] [ file-name ] / ] executor-ID
  ```

- STOP EXECUTOR stops executors whose state is ACTIVE or ON, thus making the executors unavailable for use by jobs:

  ```
  STO[P] [ EXECUTOR ] [ / OU[T] [ file-name ] / ] executor-ID
  ```

Job Commands

- ACTIVATE JOB reactivates suspended processes associated with jobs:

  ```
  AC[TIVATE] [ JOB ] [ / OU[T] [ file-name ] / ]
  { job-ID, }
  { ( [ [NOT] job-ID ,]… [NOT] filter [ , [NOT] filter ]… )}
  ```

  For a description of *filter*, see STOP JOB Command on page 6-180.

- ALTER JOB changes job attributes:

  ```
  ALT[ER] [ JOB ] [ / OU[T] [ file-name ] / ]
  { job-ID }
  ```

```
{ ( [ [NOT] job-ID , ]… [NOT] filter [ , [NOT] filter]… )
{ , LIK[E] job-ID-2 [ , attribute ]… }
{, attribute [ , attribute ]…}
[ { IN } ]
[, REM[OVE] { J[OB]-L[OG] } ] …
[ { LIB } ]
```

For a description of *attribute*, see SUBMIT JOB Command on page 6-183.

For a description of *filter*, see STATUS JOB Command on page 6-165.

- ASSUME JOB sets the default object keyword to JOB:

  ```
  ASSU[ME] JOB
  ```

- DELETE JOB deletes jobs that are not executing, over-limit, or suspended:

  ```
  D[ELETE] [ JOB ] [ / OU[T] [ file-name ] / ] { job-ID }
  {( [ [NOT] job-ID ,]… [NOT] filter [ , [NOT] filter ]… )}
  ```

  For a description of *filter*, see STATUS JOB Command on page 6-165.

- INFO JOB lists the attributes of jobs:

  ```
  I[NFO] [ JOB ] [ / OU[T] [ file-name ] / ] { job-ID }
  {( [ [NOT] job-ID , ]… [NOT] filter [ , [NOT] filter ]… )}
  [ , attribute ]… [ , OW[NER] ] [ , O[BEY]-[FORM] ]
  ```

  attribute is one of:

  ```
  AF[TER]
  AT
  A[TTACHMENT]-S[ET]
  CA[LENDAR]
  CL[ASS]
  DES[CRIPTION]
  EV[ERY]
  E[XECUTOR]-P[ROGRAM
  EXT[SWAP]
  HIG[HPIN]
  HOLD
  HOLDA[FTER]
  IF[FAILS]
  IN
  J[OB]-L[OG]
  J[OBID]-Z[ERO]
  LIB
  LIM[IT]
  MAXPRINTL[INES]
  MAXPRINTP[AGES]
  ME[M]
  NA[ME]
  OU[T]
  PF[S]
  PR[I]
  P[URGE]-[IN]-[FILE]
  REST[ART]
  RUND
  ```

```
SA[VEABEND]
SEL[PRI]
STAL[L]
STARTU[P]
S[TOP]-[ON]-[ABEND]
SWA[P]
TA[PEDRIVES]
V[OLUME]
WAIT
WAITO[N]
```

For a description of *filter*, see SUSPEND JOB Command on page 6-190.

- RELEASE-WAITON releases dependent jobs from their masters, without running the masters:

  ```
  R[ELEASE]-W[AITON] dependent-job-ID FR[OM] master-job-ID
  ```

- REPORT JOB generates reports of jobs that have been submitted and run:

  ```
  REPORT JOB

  [ { job-ID } ]
  [ { CLASS class-name } ]
  , TYPE rptype
  [ , START-TIME rptime ]
  [ , END-TIME rptime  ]
  ```

  For a detailed description of *rptype* and *rptime*, see REPORT JOB Command on page 6-116.

- RESET JOB removes job attributes (except JOBID-ZERO and VOLUME) from BATCHCOM's working-attributes set:

  ```
  RESE[T] [ JOB ] [ attribute [ , attribute ]… ]
  ```

  For a description of *attribute*, see INFO JOB Command on page 6-106.

- RUNNEXT JOB makes the scheduler run a job immediately when an executor associated with the job's class is available. The command overrides the job's dependencies, timing attributes, and selection priority.

  ```
  RUNNE[XT] [ JOB ] [ / OU[T] [ file-name ] / ]
  { job-ID }
  {( [ [NOT] job-ID , ]… [NOT] filter [ , [NOT] filter ]… )}
  ```

  For a description of *filter*, see STATUS JOB Command on page 6-165.

- RUNNOW JOB makes the scheduler run a job immediately. The command overrides job dependencies, timing attributes, and selection priority and makes the scheduler create a temporary executor for the job.

  ```
  RUNNO[W] [ JOB ] [ / OU[T] [ file-name ] / ]
  { job-ID }
  {( [ [NOT] job-ID , ]… [NOT] filter [ , [NOT] filter ]… )}
  ```

  For a description of *filter*, see STATUS JOB Command on page 6-165.

- SET JOB specifies job attributes in BATCHCOM's working-attributes set:

  ```
  SET [ JOB ] [ LIK[E] job-ID , ] attribute [ , attribute ]…
  ```

  For a description of *attribute*, see SUBMIT JOB Command on page 6-183.

- SHOW JOB lists job attributes from BATCHCOM's working-attributes set and displays the ID of the current user:

  ```
  SHO[W] [ JOB ] [ / OU[T] [ file-name ] / ]
  [ attribute [ , attribute ]… [ , O[BEY]-[FORM] ]
  [O[BEY]-[FORM]
  ```

  For a description of *attribute*, see INFO JOB Command on page 6-106.

- STATUS JOB displays jobs' numbers, names, owners, processing states, classes, SELPRI attributes, and log-file spooler-job numbers. The command also displays run statistics and information about executor-program processes.

  ```
  S[TATUS] [ JOB ] [ / OU[T] [ file-name ] / ]
  {job-ID }
  {( [ [NOT] job-ID , ]… [NOT] filter [ , [NOT] filter ]… ) }
  [ , SEL[PRI] ] [ , DET[AIL] ]
  ```

  filter is one of:

  ```
  A[TTACHMENT]-S[ET] [[ ( user-ID ) ] attachment-setID ]
  [#CURRENT ]
  CL[ASS] class-ID
  IN file-ID
  STATE state
  U[SER] user-ID
  WAITO[N] master-job
  ```

- STATUS-HISTORY lists scheduler log-file events generated by a job:

  ```
  S[TATUS]-H[ISTORY] [ / OU[T] [ file-name ] / ] job-ID
  [ , N[B]-[LOG] log-file ] [ , RUN number ]
  [ , AF[TER] [ date ] [ time ] ]
  ```

- STOP JOB stops executing, over-limit, or suspended jobs:

  ```
  STO[P] [ JOB ] [ / OU[T] [ file-name ] / ]
  { job-ID }
  {( [ [NOT] job-ID , ]… [NOT] filter [ , [NOT] filter ]… )}
  ```

  For a description of *filter*, see STATUS JOB Command on page 6-165.

- SUBMIT JOB submits jobs to a scheduler:

  ```
  SUB[MIT] [ JOB ] [ / OU[T] [ file-name ] / ]
  [ job-name-1 ] [ , LIK[E] job-name-2 ] [ , attribute ]…
  ```

  attribute is one of:

  ```
  AF[TER] [ date ] [ time ]
  AT [ date ] [ time ]
  A[TTACHMENT]-S[ET]
  [ [( user-id) } attachment-set-ID | #CURRENT ]
  ```

```
CA[LENDAR] [ file-name ]
CL[ASS] class-name
DES[CRIPTION] " [ string ] "
EV[ERY] [ weeks WEEK[S] ]
[ days D[AYS] ]
[ hours [ : mins ] [HOURS] ]
[ hours H[OURS] [ mins MIN[UTES] ]
[ crontab-entry ]
E[XECUTOR]-P[ROGRAM] file-name
EXT[SWAP] { $volume |file-name }
HIG[HPIN] { OF[F] | ON }
HOLD { OF[F] | ON }
HOLDA[FTER] { OF[F] | ON }
IF[FAILS] { OF[F] | ON }
IN [ file-name ]
J[OB]-L[OG] [ log-file ]
J[OBID]-Z[ERO] { OF[F] | ON }
LIB [ file-name ]
LIM[IT] hours [ : mins ]
MAXPRINTL[INES] { number | NON[E] }
MAXPRINTP[AGES] { number | NON[E] }
ME[M] number
NA[ME] $process-name
OU[T] [ file-name ]
PF[S] number
PR[I] number
P[URGE]-[IN]-[FILE] { OF[F] | ON }
REST[ART] { OF[F] | ON }
RUND { OF[F] | ON }
SA[VEABEND] { OF[F] | ON }
SEL[PRI] number
STAL[L] { OF[F] | ON }
STARTU[P] " param-set "
S[TOP]-[ON]-[ABEND] { OF[F] | ON }
SWA[P] { $volume | file-name }
TA[PEDRIVES] number
TERM $process-name
V[OLUME] { \node. [$volume ] [ , “security” ] }
{ [ \node.$volume [ , “security” ] }
{ [ \node.[$volume ] , “security”
WAIT hours [ : mins ]
WAITO[N] [ job-name [ case ] ]
[ ( job-name [ case ] [ , job-name [ case ] ]… ) ]
```

- SUSPEND JOB suspends executing processes associated with jobs:

  ```
  SUS[PEND] [ JOB ] [ / OU[T] [ file-name ] / ]
  { job-ID }
  {( [ [NOT] job-ID , ]… [NOT] filter [ , [NOT] filter ]… )}
  ```

  For a description of *filter*, see STATUS JOB Command on page 6-165.

Scheduler Commands

- ABORT SCHEDULER immediately stops all executing and suspended processes associated with jobs and shuts down the scheduler:

  ```
  AB[ORT] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
  ```

- ADD SCHEDULER creates and initializes the scheduler's database during a cold start:

  ```
  A[DD] [ SCHEDULER ] [ / OU[T] [ file-name ] / ] [ ,
  attribute ]…
  ```

  attribute is one of:

  ```
  AT-A[LLOWED] { OF[F] | ON }
  B[ACKUPCPU] { cpu-number-1 [ , cpu-number-2 ] |*}
  CAT[CHUP] { OF[F] | ON }
  D[EFAULT]-C[LASS] class-name
  D[EFAULT]-E[XECUTOR]-[PROGRAM] file-name
  D[EFAULT]-H[IGHPIN] { OF[F] | ON }
  D[EFAULT]-MAXPRINTL[INES] { number | NON[E]}
  D[EFAULT]-MAXPRINTP[AGES] { number | NON[E]}
  D[EFAULT]-O[UT] file-name
  D[EFAULT]-P[RI] number
  D[EFAULT]-SE[LPRI] number
  D[EFAULT]-ST[ALL] { OF[F] | ON }
  D[EFAULT]-[STOP]-[ON]-[ABEND] { OF[F] | ON }
  EM[S] BBC{( AALVS1(ER[RORS],OF[F],ON) )
  INI[TIATION] { OF[F] | ON }
  LO[CALNAMES] [ \ remote-node ]
  [ ( \ remote-node [ , \ remote-node ]… ) ]
  M[AX]-[CONCURRENT]-[JOBS] max-concurrent-jobs
  [ , max-temporary-executors ]
  M[AX]-[PRI] number
  S[UBMIT]-A[LLOWED]BBC{( AALVS1(OF[F],ON) )
  TA[PEDRIVES] number
  ```

- ALTER SCHEDULER changes scheduler attributes.

  ```
  ALT[ER] [ SCHEDULER ] [ / OU[T] [ file-name ] / ] ,
  attribute [ , attribute ]…
  ```

  For a description of *attribute*, see ADD SCHEDULER Command on page 6-46.

- ASSUME SCHEDULER sets the default object keyword to SCHEDULER.

  ```
  ASSU[ME] SC[HEDULER]
  ```

- INFO SCHEDULER lists the attributes of schedulers.

  ```
  I[NFO] [ SCHEDULER ] [ / OU[T] [ file-name ] / ] [ ,
  attribute ]…
  [ , O[BEY]-[FORM] ]
  ```

attribute is one of:

```
AT-A[LLOWED]
B[ACKUPCPU]
CAT[CHUP]
D[EFAULT]-C[LASS]
D[EFAULT]-E[XECUTOR]-[PROGRAM]
D[EFAULT]-H[IGHPIN]
D[EFAULT]-MAXPRINTL[INES]
D[EFAULT]-MAXPRINTP[AGES]
D[EFAULT]-O[UT]
D[EFAULT]-P[RI]
D[EFAULT]-SE[LPRI]
D[EFAULT]-ST[ALL]
D[EFAULT]-[STOP]-[ON]-[ABEND]
EM[S]
INI[TIATION]
LO[CALNAMES]
M[AX]-[CONCURRENT]-[JOBS]
M[AX]-[PRI]
S[UBMIT]-A[LLOWED]
TA[PEDRIVES]
```

- OPEN specifies the target scheduler for later commands:

  ```
  OP[EN] [ \node. ] $schd
  ```

- RESET SCHEDULER removes scheduler attributes from BATCHCOM's working-attributes set:

  ```
  RESE[T] [ SCHEDULER ] [ attribute [ , attribute ]… ]
  ```

  For a description of *attribute*, see INFO SCHEDULER Command on page 6-110.

- SET SCHEDULER specifies scheduler attributes in BATCHCOM's working-attributes set:

  ```
  SET [ SCHEDULER ] attribute [ , attribute ]…
  ```

  For a description of *attribute*, see ADD SCHEDULER Command on page 6-46.

- SHOW SCHEDULER lists scheduler attributes from BATCHCOM's working-attributes set:

  ```
  SHO[W] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
  [ attribute [ , attribute ]… [ , O[BEY]-[FORM] ]
  [ O[BEY]-[FORM]
  ```

  For a description of *attribute*, see INFO SCHEDULER Command on page 6-110.

- SHUTDOWN SCHEDULER shuts down a scheduler. The command lets executing and over-limit jobs finish before shutdown, but it immediately stops suspended jobs and their processes.

  ```
  SHU[TDOWN] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
  ```

- START SCHEDULER makes available for use a scheduler you are cold starting or warm starting:

  ```
  START [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
  ```

- STATUS SCHEDULER displays information about a scheduler:

  ```
  S[TATUS] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
  ```

- SWITCHCPU SCHEDULER makes a scheduler's primary process run in the CPU of its backup process, and the backup process run in the CPU of its primary process:

  ```
  SWITCHC[PU] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
  ```

- SWITCHLOG SCHEDULER closes the current scheduler log file and opens another:

  ```
  SWITCHL[OG] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
  [ log-file ]
  ```

Other Commands

- ALLOW ERRORS specifies the number of errors allowed during execution of commands contained in a BATCHCOM input file:

  ```
  ALLO[W] num ER[RORS]
  ```

- CHANGEUSER lets you log on as a different user during a BATCHCOM session:

  ```
  CH[ANGEUSER] user-ID [ password ]
  ```

- COMMENT causes BATCHCOM to ignore the input-file line on which the command occurs:

  ```
  COM[MENT] [ comment-text ]
  ```

- DISPLAY-SPI displays the contents of SPI-format command and response buffers that BATCHCOM sends to and receives from the scheduler:

  ```
  D[ISPLAY]-SP[I] { OF[F] | ON }
  ```

- EXIT ends an interactive BATCHCOM session:

  ```
  E[XIT]
  ```

- FC retrieves, and enables the editing and reexecution of, command lines from BATCHCOM's history buffer:

  ```
  F[C] [ num | - num | text ]
  ```

- HELP displays information about objects, commands, and attributes and about other general topics such as keyword abbreviation:

  ```
  H[ELP] [ / OU[T] [ file-name ] / ] [ topic | ALL ]
  ```

- HISTORY displays a specified number of the most recent command lines in BATCHCOM's history buffer:

  ```
  HIS[TORY] [ / OU[T] [ file-name ] / ] [ num ]
  ```

- OBEY causes BATCHCOM to execute commands from a specified disk file:

  ```
  O[BEY] file-name
  ```

- RUN runs a program during a BATCHCOM session:

  ```
  RUN program-file [ / OU[T] [ list-file ] / ] [ param-set ]
  ```

- SYSTEM specifies the default node for a BATCHCOM session:

  ```
  SY[STEM] [ / OU[T] [ file-name ] / ] [ \node ]
  ```

- VOLUME specifies the default node, volume, and subvolume for a BATCHCOM session, and the default security for disk files created during the session:

  ```
  V[OLUME] [ / OU[T] [ file-name ] / ] [ [ \node. ] volume ]
  [ , " security " ]
  ```

- ! reexecutes command lines from BATCHCOM's history buffer:

  ```
  ! [ num | - num | text) )
  ```

- ? displays command lines from BATCHCOM's history buffer:

  ```
  ? [ num | - num | text) )
  ```

Command Security

Table 6-2 lists BATCHCOM commands. The table indicates the commands that are restricted for use by NetBatch supervisors (users with execute access to the NETBATCH program file) and, where applicable, lists the conditions of use. Commands not restricted to NetBatch supervisors are available to all users.

Table 6-2. Command Security (page 1 of 4)

Command	NetBatch Supervisors Only?	Conditions	Page
ABORT SCHEDULER	Yes		6-30
ACTIVATE JOB	No	NetBatch supervisors can reactivate jobs belonging to any user. Users who are not NetBatch supervisors can reactivate any job whose input file is a disk file to which they have write access. If the input file does not exist or is a device or process, only the owner and NetBatch supervisors can reactivate the job.	6-32
ADD ATTACHMENT-SET	No		6-34
ADD CLASS	Yes		6-40
ADD EXECUTOR	Yes		6-42
ADD SCHEDULER	Yes		6-46
ALLOW ERRORS	No		6-51
ALTER ATTACHMENT-SET	No	You must have write access to the set you want to alter.	6-52
ALTER CLASS	Yes		6-58
ALTER EXECUTOR	Yes		6-59
ALTER JOB	No	You can alter all attributes of a job if the job has a disk input file to which you have write access or, if you are the job's owner, if the job's input file does not exist or is a device or a process. NetBatch supervisors can alter all but these attributes of any job: ATTACHMENT-SET, DESCRIPTION, EXECUTOR-PROGRAM, HIGHPIN, IN, JOB-LOG, JOBID-ZERO, LIB, NAME, OUT, PURGE-IN-FILE, RUND, STARTUP, STOP-ON-ABEND, and VOLUME.	6-62
ALTER SCHEDULER	Yes		6-67
ASSUME ATTACHMENT-SET	No		6-69
ASSUME CLASS	No		6-70
ASSUME EXECUTOR	No		6-71

Table 6-2. Command Security (page 2 of 4)

Command	NetBatch Supervisors Only?	Conditions	Page
ASSUME JOB	No		6-72
ASSUME SCHEDULER	No		6-73
CHANGEUSER	No		6-74
COMMENT	No		6-76
DELETE ATTACHMENT-SET	No	To delete an attachment set, you must have purge access to the set. To delete ASSIGNs, DEFINEs, and PARAMs from an attachment set, you must have write access to the set.	6-77
DELETE CLASS	Yes		6-81
DELETE EXECUTOR	Yes		6-82
DELETE JOB	No	NetBatch supervisors can delete jobs belonging to any user. Users who are not NetBatch supervisors can delete any job whose input file is a disk file to which they have write access. If the input file does not exist or is a device or process, only the owner and NetBatch supervisors can delete the job.	6-84
DISPLAY-SPI	No		6-86
EXIT	No		6-92
FC	No		6-92
HELP	No		6-95
HISTORY	No		6-98
INFO ATTACHMENT-SET	No	To display more than an attachment set's SECURITY and TEMPORARY attributes, you must have read access to the set.	6-99
INFO CLASS	No		6-103
INFO EXECUTOR	No		6-104
INFO JOB	No		6-106
INFO SCHEDULER	No		6-110
OBEY	No		6-112
OPEN	No		6-113

Table 6-2. Command Security (page 3 of 4)

Command	NetBatch Supervisors Only?	Conditions	Page
RELEASE-WAITON	No	NetBatch supervisors can release dependent jobs belonging to any user. Users who are not NetBatch supervisors can release a dependent job whose input file is a disk file to which they have write access. If the input file does not exist or is a device or process, only the owner and NetBatch supervisors can release the job.	6-115
REPORT JOB	No		6-116
RESET ATTACHMENT-SET	No		6-117
RESET CLASS	No		6-119
RESET EXECUTOR	No		6-120
RESET JOB	No		6-121
RESET SCHEDULER	No		6-122
RUN	No		6-124
RUNNEXT JOB	Yes		6-125
RUNNOW JOB	Yes		6-128
SET-ATTACHMENT-SET	No		6-130
SET CLASS	No		6-133
SET EXECUTOR	No		6-134
SET JOB	No		6-136
SET SCHEDULER	No		6-140
SHOW ATTACHMENT-SET	No		6-142
SHOW CLASS	No		6-145
SHOW EXECUTOR	No		6-146
SHOW JOB	No		6-147
SHOW SCHEDULER	No		6-150
SHUTDOWN SCHEDULER	Yes		6-152
START EXECUTOR	Yes		6-155

Table 6-2. Command Security (page 4 of 4)

Command	NetBatch Supervisors Only?	Conditions	Page
START SCHEDULER	Yes		6-157
STATUS ATTACHMENT-SET	No		6-160
STATUS EXECUTOR	No		6-163
STATUS JOB	No		6-165
STATUS SCHEDULER	No		6-173
STATUS-HISTORY	No	You must have read access to a log file for BATCHCOM to display events from it.	6-175
STOP EXECUTOR	Yes		6-178
STOP JOB	No	NetBatch supervisors can stop jobs belonging to any user. Users who are not NetBatch supervisors can stop any job whose input file is a disk file to which they have write access. If the input file does not exist or is a device or process, only the owner and NetBatch supervisors can stop the job.	6-180
SUBMIT JOB	No		6-183
SUSPEND JOB	No	NetBatch supervisors can suspend jobs belonging to any user. Users who are not NetBatch supervisors can suspend any job whose input file is a disk file to which they have write access. If the input file does not exist or is a device or process, only the owner and NetBatch supervisors can suspend the job.	6-190
SWITCHCPU SCHEDULER	Yes		6-192
SWITCHLOG SCHEDULER	Yes		6-193
SYSTEM	No		6-195
VOLIUME	No		6-196
!	No		6-198
?	No		6-200

Command Descriptions

This subsection contains descriptions and examples of the syntax, operation, and results of all BATCHCOM commands. These commands let you:

- Set up, monitor, control, and maintain schedulers:

ABORT SCHEDULER	SET SCHEDULER
ADD SCHEDULER	SHOW SCHEDULER
ALTER SCHEDULER	SHUTDOWN SCHEDULER
ASSUME SCHEDULER	START SCHEDULER
INFO SCHEDULER	STATUS SCHEDULER
OPEN	SWITCHCPU SCHEDULER
RESET SCHEDULER	SWITCHLOG SCHEDULER

- Set up, inquire about, and maintain classes:

ADD CLASS	INFO CLASS
ALTER CLASS	RESET CLASS
ASSUME CLASS	SET CLASS
DELETE CLASS	SHOW CLASS

- Set up, monitor, control, and maintain executors:

ADD EXECUTOR	SET EXECUTOR
ALTER EXECUTOR	SHOW EXECUTOR
ASSUME EXECUTOR	START EXECUTOR
DELETE EXECUTOR	STATUS EXECUTOR
INFO EXECUTOR	STOP EXECUTOR
RESET EXECUTOR	

- Set up, inquire about, and maintain attachment sets:

ADD ATTACHMENT-SET	RESET ATTACHMENT-SET
ALTER ATTACHMENT-SET	SET ATTACHMENT-SET
ASSUME ATTACHMENT-SET	SHOW ATTACHMENT-SET
DELETE ATTACHMENT-SET	STATUS ATTACHEMENT-SET
INFO ATTACHMENT-SET	

- Submit, monitor, control, and maintain jobs:

  ACTIVATE JOB
  ALTER JOB
  ASSUME JOB
  DELETE JOB
  INFO JOB
  RELEASE-WAITON
  RESET JOB
  RUNNEXT JOB
  RUNNOW JOB
  SET JOB
  SHOW JOB
  STATUS JOB
  STATUS-HISTORY
  STOP JOB
  SUBMIT JOB
  SUSPEND JOB

- Set current defaults for a BATCHCOM session:

  SYSTEM
  VOLUME

- Execute commands in a BATCHCOM input file:

  ALLOW ERRORS
  COMMENT
  OBEY

- Manipulate command lines:

  FC
  HISTORY
  !
  ?

- Display help text, change the current user, and end a session:

  CHANGEUSER
  EXIT
  HELP

- Display the contents of SPI-format command and response buffers that BATCHCOM sends to and receives from the scheduler:

  DISPLAY-SP

- Run a program during a BATCHCOM session:

  RUN

Wild-Card Characters

Some attachment-set, class, executor, and job commands enable you to specify a range of those objects by using either or both the asterisk (*) and question mark (?) wild-card characters. The characters matched by these wild-card characters are:

*

matches zero or more characters. For example, A*D matches character strings beginning with A and ending in D (such as ABCD and AD, but not CAD or ADE). You can use multiple asterisks if you separate them by at least one other character.

For example, *CD* matches strings containing CD (such as ABCDEF, XYZCD, and CD21, but not BC3D or DCA). * by itself matches all strings.

?

matches a single character. For example, ABC?? matches five-character strings beginning with ABC (such as ABCDE and ABC12, but not ABCDEF).

ABORT SCHEDULER Command

To immediately stop all executing and suspended processes associated with jobs and to shut down the scheduler, use the ABORT SCHEDULER command.

The scheduler deletes from its database jobs whose processes it stops. The only jobs not deleted are recurrent jobs (jobs with the CALENDAR or EVERY attribute) and jobs with the HOLDAFTER ON attribute.

```
AB[ORT] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

Considerations

- The ABORT SCHEDULER command is available to NetBatch supervisors only.
- ABORT SCHEDULER is one of two commands you can use to shut down a scheduler. The other command, SHUTDOWN SCHEDULER, lets executing and over-limit jobs finish before shutdown. Like ABORT SCHEDULER, it stops suspended jobs immediately. For more information, see SHUTDOWN SCHEDULER Command on page 6-152.
- No scheduler or job control commands are effective after execution of an ABORT SCHEDULER command.
- Shutting down a scheduler results in the closure of its database files and log file. The scheduler keeps details of its configuration at shutdown in its BATCHCTL file for use during a warm start.
- To restart a shutdown scheduler, do either of:
  - Warm start the scheduler:
    1. Run the scheduler program NETBATCH.
    2. Start the scheduler by using the START SCHEDULER command.
  - Cold start the scheduler:

1. Run the scheduler program NETBATCH.
2. Create and initialize the scheduler's database by using the ADD SCHEDULER command.
3. Start the scheduler by using the START SCHEDULER command.

For more information about scheduler cold-start and warm-start procedures, see Section 3, Scheduler Planning, Configuration, and Management.

- You can omit the object keyword SCHEDULER from the ABORT SCHEDULER command only when SCHEDULER is the current assumed object. For more information, see ASSUME SCHEDULER Command on page 6-73.

Examples

- To shut down a scheduler after stopping executing and suspended jobs, use the ABORT SCHEDULER command:

```
28} CHANGEUSER 255,205 FPP
28} OPEN $ZBAT
NETBATCH SERVER - T9190D30 … Time: 26AUG94 15:06:23
29} STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
6 ABC 205,70 2998 EXECUTING CL1
7 XYZ 205,70 2999 SUSPENDED CL2
8 PQR 255,205 READY CL2
9 EFG 133,2 23:59:00 DEFAULT
30} ABORT SCHEDULER
Scheduler aborted
```

To confirm the results of the shutdown command:

```
31} STATUS JOB *
0201-E Process gone or path down; operation failed, \M.$ZBAT
0014-E Device does not exist, \M.$ZBAT
```

- To list ABORT SCHEDULER events recorded in the log file of the scheduler shut down in the previous example:

```
ABORT SCHEDULER \M.$:0:62:52568356 U_255,205
H_\M.$ZTN0.#PTY4
ABORT JOB ABC by NetBatch J_6 \M.$Y985:54284837 U_255,205
H_\M.$ZTN0.#PTY4
DOSTOP \M.$Y985:54284837 J_6 U_255,205 H_\M.$ZTN0.#PTY4
UPDATE EXECUTOR EXEC-01 S_ON U_255,205 H_\M.$ZTN0.#PTY4
FINISH JOB ABC T_0:0:0:0 J_6 P_DELAY U_255,205
H_\M.$ZTN0.#PTY4
DELETE JOB ABC J_6 U_255,205 H_\M.$ZTN0.#PTY4
ABORT JOB XYZ by NetBatch J_7 \M.$Y986:54285093 U_255,205
H_\M.$ZTN0.#PTY4
DOSTOP \M.$Y986:54285093 J_7 U_255,205 H_\M.$ZTN0.#PTY4
UPDATE EXECUTOR EXEC-02 S_ON U_255,205 H_\M.$ZTN0.#PTY4
```

```
FINISH JOB XYZ T_0:0:0:0 J_7 P_DELAY U_255,205
H_\M.$ZTN0.#PTY4
DELETE JOB XYZ J_7 U_255,205 H_\M.$ZTN0.#PTY4
DOSTOP Myself
```

- To warm start a scheduler shut down by the ABORT SCHEDULER command:

```
14> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 2,37 0,80 $C2
15> BATCHCOM $ZBAT; ABORT SCHEDULER
Scheduler aborted
16> PPD $ZBAT
(Process does not exist)
17> NETBATCH /NAME $ZBAT, NOWAIT/ $DATA7.ZBAT
18> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 2,34 $C2
19> BATCHCOM $ZBAT; START SCHEDULER
Scheduler started
20> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 2,34 3,17 $C2
```

ACTIVATE JOB Command

To reactivate suspended processes associated with jobs, use the ACTIVATE JOB command. The command does not reactivate suspended processes dissociated from a job by the TACL RUN command option JOBID or by the JOBID-ZERO job attribute.

```
AC[TIVATE] [ JOB ] [ / OU[T] [ file-name ] / ]
{ job-ID }
{ ( [ [NOT] job-ID ,]… [NOT] filter [ , [NOT] filter ]… )
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

job-ID

specifies a job name or number, or a range of job names. To specify a range of job names, use the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all jobs.

NOT

excludes jobs that satisfy the specified job-selection criterion.

filter

specifies one of these job-selection criteria (see STATUS JOB Command on page 6-165):

```
A[TTACHMENT]-S[ET]BBC[( AALVS1([ ( user-ID ) ] attachment-
set-
ID,#CURRENT) )
CL[ASS] class-ID
IN file-ID
STATE state
U[SER] user-ID
WAITO[N] master-job
```

Considerations

- The ACTIVATE JOB command is available to all users, but:
  - NetBatch supervisors can reactivate jobs belonging to any user.
  - Users who are not NetBatch supervisors can reactivate any job whose input file is a disk file to which they have write access. If the input file does not exist or is a device or process, only the owner and NetBatch supervisors can reactivate the job.

    If you are not a NetBatch supervisor, an ACTIVATE JOB command that specifies job-ID with wild-card characters reactivates only your own jobs. To reactivate another user's job if you have write access to its input file, specify the job's full name or number in job-ID.
- A suspended job whose attributes include EVERY accumulates a run backlog if suspended for longer than the EVERY interval and the scheduler has the attribute CATCHUP ON. When you reactivate the job, it runs repeatedly until the backlog clears. For example, an hourly 10-minute job submitted at 0900 should run three times before 1200 (at 0900, 1000, and 1100). If suspended at 0905, then reactivated at 1105, the job finishes executing and runs twice more before running again at 1200.

  To stop a suspended job's processes and to stop its backlog running:

  1. Change the suspended job's HOLDAFTER attribute to HOLDAFTER ON by using the ALTER JOB command.
  2. Stop the suspended job's processes and delete the job from its scheduler by using the STOP JOB command.
  3. Change the scheduling attributes of the recurring job by using the ALTER JOB command. (Alternatively, delete the job by using the DELETE JOB command.)
  4. Change the recurring job's HOLD and HOLDAFTER attributes to OFF.
- A suspended job whose attributes include CALENDAR does not accumulate a run backlog while suspended. When you reactivate the job, it finishes executing but does not run again until the next future CALENDAR time.

- You can omit the object keyword JOB from the ACTIVATE JOB command only when JOB is the current assumed object. For more information, see ASSUME JOB Command on page 6-72.

Example

To use SUBMIT JOB, SUSPEND JOB, and ACTIVATE JOB commands to submit, suspend, and reactivate a job:

```
> BATCHCOM $ZBAT; SUBMIT JOB, EXECUTOR-PROGRAM DELAY, STARTUP
"4 MINS"
Job ZBAT-0099 Jobnumber 99 submitted
> BATCHCOM $ZBAT; SUSPEND JOB 99
Job ZBAT-0099 Jobnumber 99 suspended
> BATCHCOM $ZBAT; ACTIVATE JOB 99
Job ZBAT-0099 Jobnumber 99 activated
```

The resulting log-file events are:

```
> FUP COPY LOGABO,,SHARE .
.
ADD JOB ZBAT-0099 C_DEFAULT:3 J_99 U_255,255
H_\MELBDEV.$ZTN0.#PTY4
BEGIN JOB (SUPER.SUPER)ZBAT-0099:1 E_E1 L_700 J_99 P_DELAY
\MELBDEV.$Z729:17398829 U_255,255
UPDATE EXECUTOR E1 S_ACTIVE
LIST JOB ZBAT-0099 EXECUTING J_99
START EXECUTOR-PROGRAM U_255,255 J_99 P_DELAY
\MELBDEV.$Z729:17398829
SUSPND JOB ZBAT-0099 J_99 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
LIST JOB ZBAT-0099 SUSPENDED J_99 U_255,255
H_\MELBDEV.$ZTN0.#PTY4
A'VATE JOB ZBAT-0099 J_99 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
LIST JOB ZBAT-0099 EXECUTING J_99 U_255,255
H_\MELBDEV.$ZTN0.#PTY4
STOP CC_0 EXECUTOR-PROGRAM J_99 \MELBDEV.$Z729:17398829
UPDATE EXECUTOR E1 S_ON
FINISH JOB ZBAT-0099 T_0:0:0:12 J_99 P_DELAY
DELETE JOB ZBAT-0099 J_99
```

ADD ATTACHMENT-SET Command

Use the ADD ATTACHMENT-SET command to add attachment sets to a scheduler.

```
A[DD] [ ATTACHMENT-SET ] [ / OU[T] [ file-name ] / ]
{ ( user-ID ) ] attachment-set-name | #CURRENT }
[ , attribute ]…
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists but creates an EDIT

file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

user-ID

specifies the user ID of the attachment-set owner. (*user-ID* must be in *group-name.user-name* or *group-ID,user-ID* form.) The default is the ID of the current user.

attachment-set-name

is the name of an attachment set. The name can contain from 1 to 24 letters and numbers. It also can contain hyphens but must begin with a letter and end with a letter or number. The name cannot contain spaces.

Note. To avoid confusion, do not use these keywords and keyword aliases as attachment-set names: ADP, ATTACHMENT-SET, CLASS, EXECUTOR, JOB, JOBCLASS, and SCHEDULER.

#CURRENT

causes the scheduler to assign attachment-set ownership automatically to the current user and to generate a number as the set identifier.

attribute

is one of these attachment-set attributes (see Section 7, Attributes):

```
( ASSI[GN] ASSIGN-name , ASSIGN-attributes )
( DEFI[NE] DEFINE-name-1 , [ LIK[E] DEFINE-name-2 , ]
[ CLASS DEFINE-class , ]
DEFINE-attribute [ , DEFINE-attribute ]… )
( PA[RAM] PARAM-name PARAM-value )
SEC[URITY] " security "
TEM[PORARY]BBC{( AALVS1(OF[F],ON) )
```

attribute overrides attributes of the same type and, for ASSIGNs, DEFINEs, and PARAMs, name in the working-attributes set. It also overrides attachment-set defaults (see Table 6-3) when the defaults do not appear in the working set. If you omit *attribute*, the attachment set adopts its attributes from the working-attributes set and attachment-set defaults.

Table 6-3. Attachment-Set Defaults (page 1 of 2)

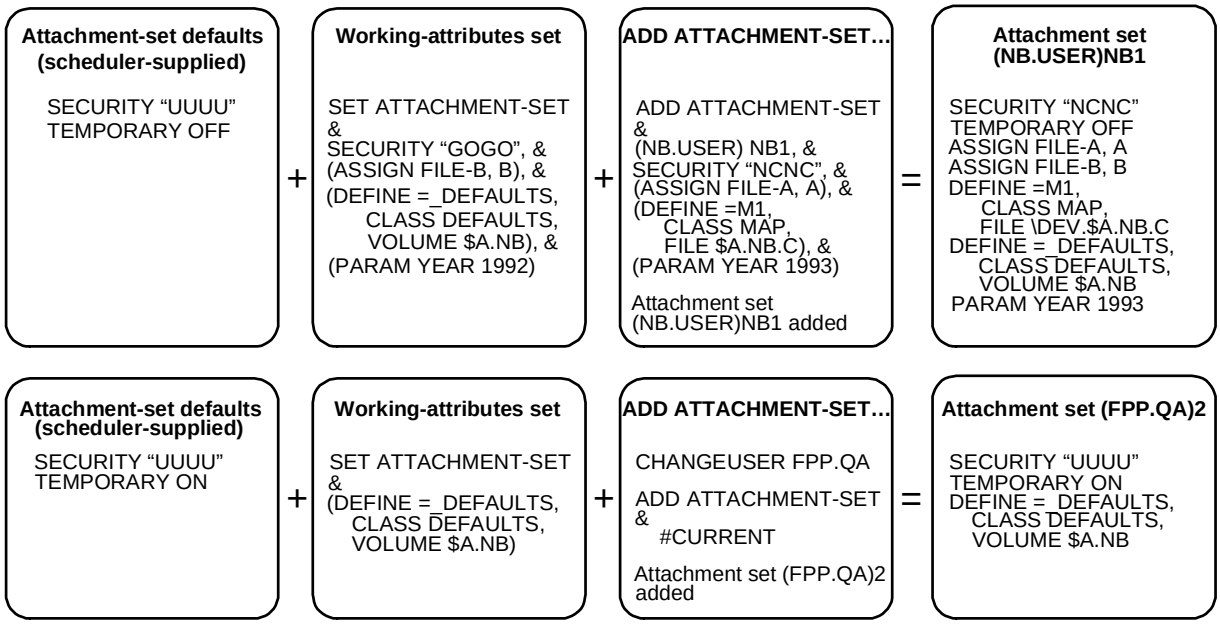
Attribute	Default Value
ASSIGN	No ASSIGN attribute.
DEFINE	DEFINE =_DEFAULTS specifying the node and volume set by the last SET ATTACHMENT-SET (DEFINE =_DEFAULTS, VOLUME …), SYSTEM, or VOLUME command. If there was no such command, the DEFINE specifies the current node and volume from the TACL environment.

Table 6-3. Attachment-Set Defaults (page 2 of 2)

Attribute	Default Value
PARAM	No PARAM attribute.
SECURITY	UUUU.
TEMPORARY	OFF for attachment sets added by ADD ATTACHMENT-SET commands that specify an ID of *attachment-set-name*. ON for attachment sets added by ADD ATTACHMENT-SET commands that specify an ID of #CURRENT.

Figure 6-1 illustrates the attribute conditions described earlier.

Figure 6-1. Examples of Attribute Defaulting-Attachment Set



Considerations

- The ADD ATTACHMENT-SET command is available to all users.
- BATCHCOM supports all classes of DEFINE available in the TACL environment. The default DEFINE class is MAP.
- The LIKE qualifier is available for DEFINEs only. Use it in the ADD ATTACHMENT-SET command to specify DEFINEs whose attributes match those of DEFINEs specified earlier in the command.
- The ADD ATTACHMENT-SET command changes the attachment set specified by the #CURRENT variable to the added attachment set. To illustrate this consideration, this example shows the state of the variable after various ADD

ATTACHMENT-SET commands. BATCHCOM displays an “undefined substitution” message when the variable has a null value:

```
6} CHANGEUSER NB.USER psswrd
7} RESET ATTACHMENT-SET
8} INFO ATTACHMENT-SET #CURRENT
-^-0345-
E Undefined substitution
9} ADD ATTACHMENT-SET DAYLOG, …
Attachment-set (NB.USER)DAYLOG added
10} INFO ATTACHMENT-SET #CURRENT
ATTACHMENT-SET ATTRIBUTES for (NB.USER)DAYLOG .
.
11} ADD ATTACHMENT-SET #CURRENT, …
Attachment-set (NB.USER)46 added
12} INFO ATTACHMENT-SET #CURRENT
ATTACHMENT-SET ATTRIBUTES for (NB.USER)46 .
.
13} RESET ATTACHMENT-SET
14} INFO ATTACHMENT-SET #CURRENT
-^-0345-
E Undefined substitution
```

- ADP (an abbreviation of ASSIGNs, DEFINEs, and PARAMs) is an alias of the object keyword ATTACHMENT-SET.
- You can omit the object keyword ATTACHMENT-SET from the ADD ATTACHMENT-SET command only when ATTACHMENT-SET is the current assumed object. For more information, see ASSUME ATTACHMENT-SET Command on page 6-69.

Examples

- This example shows the ADD ATTACHMENT-SET command adding an attachment set to scheduler $ZBAT. Note the sources of the set’s attributes.

```
> LOGON NB.USER, psswrd
> ASSIGN SALES-LOG, $A.NB.SALES
> ALTER DEFINE =_DEFAULTS, VOLUME $A.NB
> PARAM DAY TUESDAY
> BATCHCOM $ZBAT
1} SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
ASSIGN SALES-LOG, $A.NB.SALES
PARAM DAY TUESDAY
2} SET ATTACHMENT-SET SECURITY "AGGO", TEMPORARY OFF, &
}} (ASSIGN ORDERS-LOG, $A.NB.ORDERS), &
}} (DEFINE =OUT, CLASS SPOOL, LOC \A.$S.#DEFAULT), &
}} (PARAM DAY WEDNESDAY)
3} SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
security: "AGGO"
```

```
temporary: Off
attachments: DEFINE =OUT, CLASS SPOOL, LOC \A.$S.#DEFAULT
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
ASSIGN ORDERS-LOG, $A.NB.ORDERS
ASSIGN SALES-LOG, $A.NB.SALES
PARAM DAY WEDNESDAY
4} ADD ATTACHMENT-SET MARKETING, &
}} (ASSIGN STOCK-LOG, $A.NB.STOCK), &
}} (DEFINE =OUT, LOC \A.$S.#MRKTNG), (PARAM DAY THURSDAY)
Attachment-set (NB.USER)MARKETING added
5} INFO ATTACHMENT-SET MARKETING, DETAIL
ATTACHMENT-SET ATTRIBUTES for (NB.USER)MARKETING
security: "AGGO"
temporary: Off
attachments: DEFINE =OUT, CLASS SPOOL, LOC \A.$S.#MRKTNG
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
ASSIGN ORDERS-LOG, $A.NB.ORDERS
ASSIGN SALES-LOG, $A.NB.SALES
ASSIGN STOCK-LOG, $A.NB.STOCK
PARAM DAY THURSDAY
```

- This example shows an ADD ATTACHMENT-SET command that includes the #CURRENT qualifier. #CURRENT causes the scheduler to assign attachment-set ownership to the current user and to generate a number as the set identifier. Note the sources of the set's attributes.

```
8} CHANGEUSER SUPER.FPP psswrd
8} SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =MAP, CLASS MAP, FILE
\MELBDEV.$A.NB.CALENDAR
DEFINE =SQLCAT, CLASS CATALOG, SUBVOL
\MELBDEV.$DATA7.SQLCAT
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
ASSIGN DATES-FILE, DATES
ASSIGN TIMES-FILE, TIMES
PARAM DAY 23
PARAM MONTH JUNE
PARAM YEAR 1993
9} ADD ATTACHMENT-SET #CURRENT, TEMPORARY OFF
Attachment-set (SUPER.FPP)48 added
10} INFO ATTACHMENT-SET 48, DETAIL
ATTACHMENT-SET ATTRIBUTES for (SUPER.FPP)48
security: "UUUU"
temporary: Off
attachments: DEFINE =MAP, CLASS MAP, FILE
\MELBDEV.$A.NB.CALENDAR
DEFINE =SQLCAT, CLASS CATALOG, SUBVOL
\MELBDEV.$DATA7.SQLCAT
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
ASSIGN DATES-FILE, DATES
```

```
ASSIGN TIMES-FILE, TIMES
PARAM DAY 23
PARAM MONTH JUNE
PARAM YEAR 1993
```

- This example shows the effect of the LIKE qualifier in an ADD ATTACHMENT-SET (DEFINE ...) command:

```
28} ADD ATTACHMENT-SET (SUPER.NB)OUT, &
}} (DEFINE =LOC1, CLASS SPOOL, BATCHNAME ACCOUNTS, COPIES 2,
FORM INVOICES, HOLDAFTER ON, LOC \MELBDEV.$S.#DEFAULT,
MAXPRINTLINES 132, OWNER "205,255", SELPRI 7), &
}} (DEFINE =LOC2, LIKE =LOC1, LOC \MELBQAT.$S.#DEFAULT), &
}} (DEFINE =LOC3, LIKE =LOC2, LOC \MELBORN.$S.#DEFAULT)
Attachment-set (SUPER.NB)OUT added
29} INFO ATTACHMENT-SET (SUPER.NB)OUT, DEFINE =LOC*
ATTACHMENT-SET ATTRIBUTES for (SUPER.NB)OUT
attachments: DEFINE =LOC1, CLASS SPOOL, LOC
\MELBDEV.$S.#DEFAULT, COPIES 2, FORM
INVOICES, HOLDAFTER ON, OWNER
"205,255", SELPRI 7, BATCHNAME
ACCOUNTS, MAXPRINTLINES 132
DEFINE =LOC2, CLASS SPOOL, LOC
\MELBQAT.$S.#DEFAULT, COPIES 2, FORM
INVOICES, HOLDAFTER ON, OWNER
"205,255", SELPRI 7, BATCHNAME
ACCOUNTS, MAXPRINTLINES 132
DEFINE =LOC3, CLASS SPOOL, LOC
\MELBORN.$S.#DEFAULT, COPIES 2, FORM
INVOICES, HOLDAFTER ON, OWNER
"205,255", SELPRI 7, BATCHNAME
ACCOUNTS, MAXPRINTLINES 132
```

- This example shows execution of an ADD ATTACHMENT-SET command and lists the event relating to that command from the scheduler log file:

```
139} CHANGEUSER 205,70 psswrd
139} ADD ATTACHMENT-SET (133,2)ADP-1, (ASSIGN X, $A.NB.X),
(PARAM DAY FRIDAY), (DEFINE =_DEFAULTS, VOLUME $A.NB),
TEMPORARY OFF, SECURITY "AAAA"
Attachment-set (NB.USER)ADP-1 added
140} RUN FUP COPY $DATA7.ZBAT.LOGABR,,SHARE .
.
ADD ATT-SET (133,2)ADP-1 SECURITY "AAAA" TEMPORARY OFF
U_205,70 H_\MELBDEV.$ZTN0.#PTY4
```

ADD CLASS Command

Use the ADD CLASS command to add classes to a scheduler.

```
A[DD] [ CLASS ] [ / OU[T] [ file-name ] / ]
class-name [ , attribute ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

class-name

is the name of a class. The name can contain from 1 to 24 letters and numbers. It also can contain hyphens but must begin with a letter and end with a letter or number. The name cannot contain spaces.

Note. To avoid confusion, do not use these keywords and keyword aliases as class names: ADP, ATTACHMENT-SET, CLASS, EXECUTOR, JOB, JOBCLASS, and SCHEDULER.

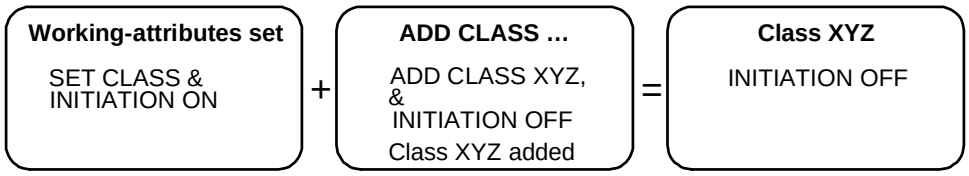
attribute

specifies this class attribute (see Section 7, Attributes):

`INI[TIATION]BBC{( AALVS1(OF[F],ON) )`

attribute overrides the INITIATION attribute in the working-attributes set. If you omit *attribute*, the class adopts its attribute from the set. Figure 6-2 illustrates these conditions.

Figure 6-2. Example of Attribute Defaulting-Class



Considerations

- The ADD CLASS command is available to NetBatch supervisors only.
- After adding a class, you must assign it to at least one executor by using the ADD EXECUTOR command or ALTER EXECUTOR command. If you do not do this, the

scheduler never scans the class for jobs. You cannot assign a class to an executor until you add the class to the scheduler.

- JOBCLASS is an alias of the object keyword CLASS.
- You can omit the object keyword CLASS from the ADD CLASS command only when CLASS is the current assumed object. For more information, see ASSUME CLASS Command on page 6-70.

Examples

- To add a class with the attribute INITIATION OFF:

```
1} ADD CLASS DEVELOPMENT, INITIATION OFF
Class DEVELOPMENT added
```

- This example shows a class adopting its INITIATION attribute from the working-attributes set:

```
2} SHOW CLASS
CLASS ATTRIBUTE
initiation: On
3} ADD CLASS PRODUCTION
Class PRODUCTION added
4} INFO CLASS PRODUCTION
CLASS ATTRIBUTE for PRODUCTION
initiation: On
```

- This example shows BATCHCOM creating an input file (CLASSES) containing ADD CLASS commands. The commands reflect the configuration of scheduler $ZBAT's classes. The example then shows BATCHCOM executing the commands from the file to configure the classes of scheduler $BANK.

```
> BATCHCOM $ZBAT; INFO CLASS /OUT CLASSES/ *, OBEY-FORM
> FUP COPY CLASSES
ADD CLASS DEVELOPMENT, INITIATION Off
ADD CLASS PRODUCTION, INITIATION On
> BATCHCOM /IN CLASSES/ $BANK
ADD CLASS DEVELOPMENT, INITIATION Off
Class DEVELOPMENT added
ADD CLASS PRODUCTION, INITIATION On
Class PRODUCTION added
```

- This example shows execution of an ADD CLASS command and lists the event relating to that command from the scheduler log file:

```
> LOGON SUPER.SUPER, password
> BATCHCOM $SCHD; ADD CLASS CL1, INITIATION OFF
Class CL1 added
> FUP COPY LOGAAF,,SHARE .
.
ADD CLASS CL1 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
```

ADD EXECUTOR Command

Use the ADD EXECUTOR command to add executors to a scheduler.

```
A[DD] [ EXECUTOR ] [ / OU[T] [ file-name ] /] executor-name-1
[ , LIK[E] executor-name-2 ] [ , attribute ]…
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

executor-name-1

is the name of the executor you want to add. The name can contain from 1 to 24 letters and numbers. It also can contain hyphens but must begin with a letter and end with a letter or number. The name cannot contain spaces.

Note. To avoid confusion, do not use these keywords and keyword aliases as executor names: ADP, ATTACHMENT-SET, CLASS, EXECUTOR, JOB, JOBCLASS, and SCHEDULER.

LIKE

specifies that all attributes are to match those of executor *executor-name-2*. LIKE overrides executor attributes in the working-attributes set and executor defaults (see Table 6-4 on page 6-43). *attribute* overrides *executor-name-2* attributes of the same type.

executor-name-2

is the name of an existing executor whose attributes you want executor *executor-name-1* to match.

attribute

is one of these executor attributes (see Section 7, Attributes):

```
CL[ASS] BBC{( AALVS1( class-name,( class-name [ , class-name
]…),*) )
CP[U] cpu-number
```

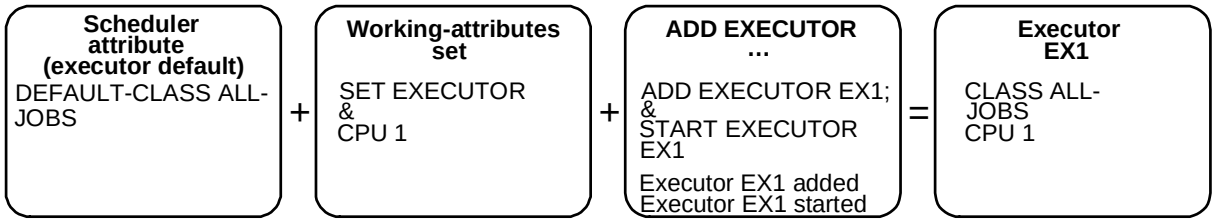
attribute overrides attributes of the same type in the working-attributes set or specified by *executor-name-2*. It also overrides executor defaults (see Table 6-4) when the defaults do not appear in the working set. If you omit *attribute* and LIKE, the executor adopts its attributes from the working-attributes set and executor defaults.

Table 6-4. Executor Defaults

Attributes	Default Values
CLASS	DEFAULT-CLASS scheduler attribute
CPU	No default

Figure 6-3 illustrates some of the *attribute* conditions.

Figure 6-3. Example of Attribute Defaulting-Executor



Considerations

- The ADD EXECUTOR command is available to NetBatch supervisors only.
- The CLASS executor attribute lets you assign up to eight existing classes to an executor. (An existing class is one added to the scheduler by the ADD CLASS command.)
- The order in which you assign classes to an executor determines the order in which the scheduler scans the classes for jobs. For example, assigning classes B and A to an executor (in that order) makes the scheduler scan class B before it scans class A. Jobs in class B therefore run before jobs in class A.
- You must specify the CPU attribute for each executor you add, but you can specify only one CPU per executor.
- After adding an executor, to make it available for use by jobs, you must start it by using the START EXECUTOR command.
- You can omit the object keyword EXECUTOR from the ADD EXECUTOR command only when EXECUTOR is the current assumed object. For more information, see ASSUME EXECUTOR Command on page 6-71.

Examples

- To add an executor with the attributes CPU 0 and CLASS DEFAULT:

```
35} ADD EXECUTOR ZBAT-EXEC-0, CPU 0, CLASS DEFAULT
Executor ZBAT-EXEC-0 added
```

To make the executor available for use by jobs:

```
36} START EXECUTOR ZBAT-EXEC-0
Executor ZBAT-EXEC-0 started
```

- The ADD EXECUTOR command in this example adds an executor that adopts its CPU attribute from the working-attributes set. The executor's CLASS attribute comes from the scheduler's DEFAULT-CLASS attribute. The START EXECUTOR command makes the executor available for use by jobs.

```
48} SHOW EXECUTOR
EXECUTOR ATTRIBUTES
cpu: 1
49} INFO SCHEDULER, DEFAULT-CLASS
SCHEDULER ATTRIBUTES
default-class: ZBAT-CLASS
50} ADD EXECUTOR ZBAT-EXEC-1
Executor ZBAT-EXEC-1 added
51} INFO EXECUTOR ZBAT-EXEC-1
EXECUTOR ATTRIBUTES for ZBAT-EXEC-1
cpu: 1
classes: ZBAT-CLASS
52} START EXECUTOR ZBAT-EXEC-1
Executor ZBAT-EXEC-1 started
```

- This example shows BATCHCOM creating an input file (ZBATEXEC) containing ADD EXECUTOR commands. The commands reflect the executor configuration of local scheduler $ZBAT. The example then shows BATCHCOM executing the commands from the file to configure the executors of scheduler \REMOTE.$ZBAT. The START EXECUTOR * command makes the executors available for use by jobs.

```
> BATCHCOM $ZBAT; INFO EXECUTOR /OUT ZBATEXEC/ *, OBEY-FORM
> FUP COPY ZBATEXEC
ADD EXECUTOR ZBAT-EXEC-0, CPU 0, CLASS(DEFAULT)
ADD EXECUTOR ZBAT-EXEC-1, CPU 1, CLASS(ZBAT-CLASS)
ADD EXECUTOR ZBAT-EXEC-2, CPU 2, CLASS*
ADD EXECUTOR ZBAT-EXEC-3, CPU 3, CLASS*
> BATCHCOM /IN ZBATEXEC/ \REMOTE.$ZBAT
ADD EXECUTOR ZBAT-EXEC-0, CPU 0, CLASS(DEFAULT)
Executor ZBAT-EXEC-0 added
ADD EXECUTOR ZBAT-EXEC-1, CPU 1, CLASS(ZBAT-CLASS)
Executor ZBAT-EXEC-1 added
ADD EXECUTOR ZBAT-EXEC-2, CPU 2, CLASS*
Executor ZBAT-EXEC-2 added
ADD EXECUTOR ZBAT-EXEC-3, CPU 3, CLASS*
Executor ZBAT-EXEC-3 added
> BATCHCOM \REMOTE.$ZBAT; START EXECUTOR *
Executor ZBAT-EXEC-0 started
Executor ZBAT-EXEC-1 started
Executor ZBAT-EXEC-2 started
Executor ZBAT-EXEC-3 started
```

- This example shows the use of the ADD EXECUTOR command's LIKE qualifier:

```
37} ADD EXECUTOR EXEC01, CPU 1, CLASS (CL1, CL2, CL3, CL4)
Executor EXEC01 added
38} ADD EXECUTOR EXEC02, LIKE EXEC01, CPU 2
Executor EXEC02 added
39} INFO EXECUTOR *
EXECUTOR ATTRIBUTES for EXEC01
cpu: 1
classes: CL1
CL2
CL3
CL4
EXECUTOR ATTRIBUTES for EXEC02
cpu: 2
classes: CL1
CL2
CL3
CL4
40} START EXECUTOR *
Executor EXEC01 started
Executor EXEC02 started
```

- This example shows scheduler log-file events relating to the execution of ADD EXECUTOR and START EXECUTOR commands:

```
> LOGON 255,205, password
> BATCHCOM $SCHD; ADD EXECUTOR EX1, CPU 0, CLASS DEFAULT;
START EXECUTOR EX1
Executor EX1 added
Executor EX1 started
> FUP COPY LOGAAL,,SHARE .
.
ADD EXECUTOR EX1 CPU 0 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
UPDATE CLASS DEFAULT S_OFF U_255,255 H_\MELBDEV.$ZTN0.#PTY4
START EXECUTOR EX1 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
UPDATE EXECUTOR EX1 S_ON U_255,255 H_\MELBDEV.$ZTN0.#PTY4
```

ADD SCHEDULER Command

Use the ADD SCHEDULER command in a scheduler cold start to create and initialize the scheduler's database. Before using the command, run the scheduler program NETBATCH. Afterward, start the scheduler with the START SCHEDULER command.

△ **Caution.** The ADD SCHEDULER command purges existing scheduler database files (except BATCHCTL and log files) before creating and initializing new files. Do not use the command unless you are sure that loss of the existing files will not affect your organization. If in doubt, back up the files before using the command.

```
A[DD] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
[ , attribute ]…
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

attribute

is one of these scheduler attributes (see Section 7, Attributes):

```
AT-A[LLOWED]BBC { OF[F] | ON }
B[ACKUPCPU]BBC { cpu-number-1 [ , cpu-number-2 ] | *
CAT[CHUP] { OF[F] | ON }
D[EFAULT]-C[LASS] class-name
D[EFAULT]-E[XECUTOR]-[PROGRAM] file-name
D[EFAULT]-H[IGHPIN] { OF[F] | ON }
D[EFAULT]-MAXPRINTL[INES] { number | NON[E] }
D[EFAULT]-MAXPRINTP[AGES { number | NON[E] }
D[EFAULT]-O[UT] file-name
D[EFAULT]-P[RI] number
D[EFAULT]-SE[LPRI] number
D[EFAULT]-ST[ALL] { OF[F] | ON }
D[EFAULT]-[STOP]-[ON]-[ABEND] { OF[F] | ON }
EM[S] { ER[RORS] | OF[F] | ON }
INI[TIATION] { OF[F] | ON }
LO[CALNAMES] [\ remote-node | ( \ remote-node [ , \ remote-
node ]… ) ]
M[AX]-[CONCURRENT]-[JOBS] max-concurrent-jobs
[ , max-temporary-executors ]
M[AX]-[PRI] number
S[UBMIT]-A[LLOWED] { OF[F] | ON }
TA[PEDRIVES] number
```

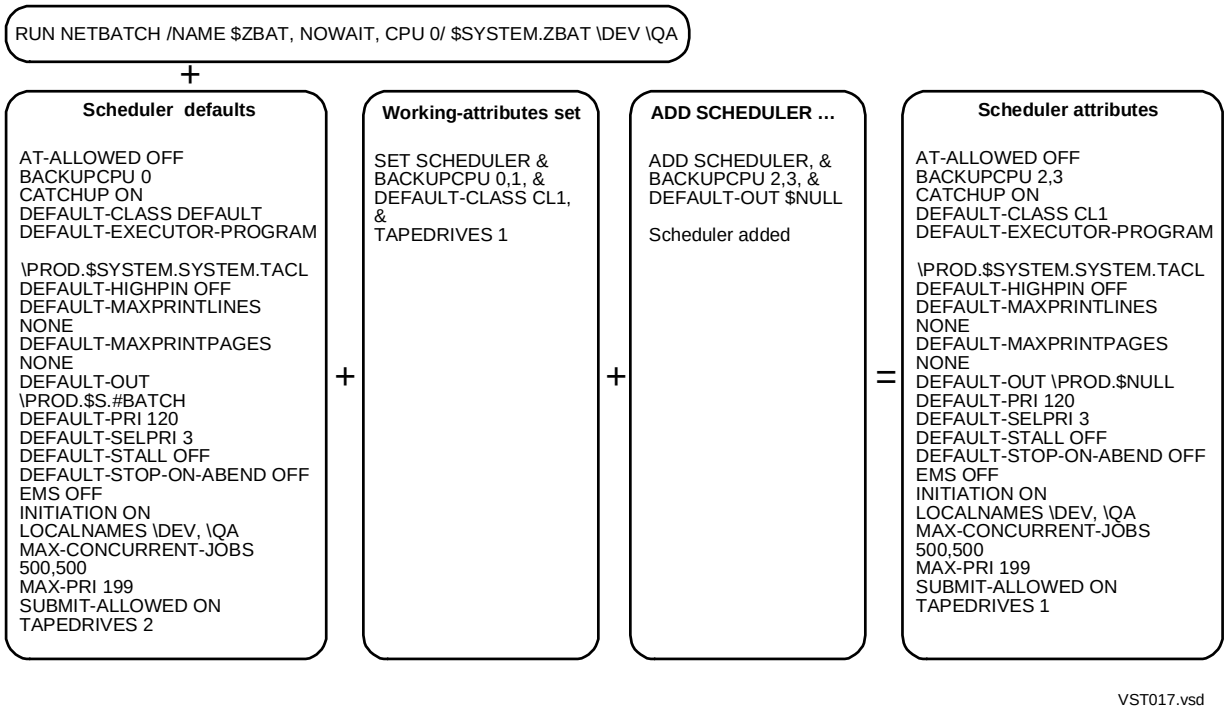
attribute overrides attributes of the same type in the working-attributes set. It also overrides scheduler defaults (see Table 6-5 on page 6-47) when the defaults do not appear in the working set. If you omit *attribute*, the scheduler adopts its attributes from the working-attributes set and scheduler defaults.

Table 6-5. Scheduler Defaults

Attribute	Default Value
AT-ALLOWED	OFF
BACKUPCPU	CPU of scheduler's primary process
CATCHUP	ON
DEFAULT-CLASS	DEFAULT
DEFAULT-EXECUTOR-PROGRAM	$SYSTEM.SYSTEM.TACL
DEFAULT-HIGHPIN	OFF
DEFAULT-MAXPRINTLINES	NONE
DEFAULT-MAXPRINTPAGES	NONE
DEFAULT-OUT	$S.#BATCH
DEFAULT-PRI	120
DEFAULT-SELPRI	3
DEFAULT-STALL	OFF
DEFAULT-STOP-ON-ABEND	OFF
EMS	OFF
INITIATION	ON
LOCALNAMES	RUN NETBATCH *remote-node* nodes
MAX-CONCURRENT-JOBS	500,500
MAX-PRI	199
SUBMIT-ALLOWED	On
TAPEDRIVES	2

Figure 6-4 on page 6-48 illustrates the *attribute* conditions.

Figure 6-4. Example of Attribute Defaulting-Scheduler



Considerations

- The ADD SCHEDULER command is available to NetBatch supervisors only.
- Table 6-6 lists the scheduler database files created or initialized by the ADD SCHEDULER command. The scheduler program secures the files “OOOO” to the super ID (255,255). The program also secures the JOB file with the PROGID option of the FUP SECURE command. This option limits file access to the local super ID.

Table 6-6. Files Created or Initialized by ADD SCHEDULER Command (page 1 of 2)

File ID	Created?	Initialized?	Purpose
ATTACH	Yes	Yes	Contains attachment sets.
ATTACH0	Yes	Yes	The ATTACH file’s alternate key file.
BATCHCTL	No	Yes	Contains scheduler attributes; created by RUN NETBATCH.
CHKQUE	Yes	Yes	The scheduler’s working file.
CHKQUE0	Yes	Yes	The CHKQUE file’s alternate key file.
EXECUTOR	Yes	Yes	Contains information about executors and their attributes.
EXECUT00	Yes	Yes	The EXECUTOR file’s alternate key file.

Table 6-6. Files Created or Initialized by ADD SCHEDULER Command (page 2 of 2)

File ID	Created?	Initialized?	Purpose
JOB	Yes	Yes	Contains information about jobs and their attributes.
JOBCLASS	Yes	Yes	Contains information about classes and their attributes.
JOBCLAS0	Yes	Yes	The JOBCLASS file's alternate key file.
NBATTX	Yes	Yes	Contains information linking attachment sets to jobs.
NBATTX0	Yes	Yes	The NBATTX file's alternate key file.

- You can omit the object keyword SCHEDULER from the ADD SCHEDULER command only when SCHEDULER is the current assumed object. For more information, see ALTER SCHEDULER Command on page 6-67.

Examples

- This example shows the result of an ADD SCHEDULER command that tries to act on a nonexistent scheduler:

```
> PPD $SCHD
(Process does not exist)
> BATCHCOM $SCHD; ADD SCHEDULER
0014-E Device does not exist, \MELBDEV.$SCHD
```

- This example shows the result of an ADD SCHEDULER command that tries to act on a started scheduler:

```
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 2,36 3,24 $C2
> BATCHCOM $ZBAT; ADD SCHEDULER
2055-E Scheduler is already started; command ignored
```

- This example shows a cold start of a previously active but now shutdown scheduler. The example shows the effect of the ADD SCHEDULER command in the cold-start process.

```
$DATA7 SCHD 71> FILEINFO
$DATA7.SCHD
Code EOF …
ATTACH 847 24576 …
ATTACH0 847 3072 …
BATCHCTL 847 552 …
CHKQUE 847 12288 …
CHKQUE0 847 12288 …
EXECUTO0 847 12288 …
EXECUTOR 847 12288 …
JOB 847P 16384 …
```

```
JOBCLAS0 847 12288 …
JOBCLASS 847 12288 …
LOGAAA 847 16104 …
NBATTX 847 3072 …
NBATTX0 847 3072 …
$DATA7 SCHD 73> NETBATCH /NAME $SCHD, NOWAIT/ $DATA7.SCHD !
$DATA7 SCHD 74> FILEINFO
$DATA7.SCHD
Code EOF …
BATCHCTL O 847 552 …
LOGAAA 847 16500 …
LOGAAB O 847 396 …
$DATA7 SCHD 75> BATCHCOM $SCHD; ADD SCHEDULER
Scheduler added
$DATA7 SCHD 76> FILEINFO
$DATA7.SCHD
Code EOF …
ATTACH 847 0 …
ATTACH0 847 0 …
BATCHCTL O 847 552 …
CHKQUE 847 0 …
CHKQUE0 847 0 …
EXECUTO0 847 0 …
EXECUTOR 847 0 …
JOB 847P 0 …
JOBCLAS0 847 0 …
JOBCLASS 847 0 …
LOGAAA 847 16500 …
LOGAAB O 847 792 …
NBATTX 847 0 …
NBATTX0 847 0 …
$DATA7 SCHD 77> BATCHCOM $SCHD; START SCHEDULER
Scheduler started
```

- This example lists cold-start events recorded in the log file of the scheduler started in the previous example:

```
CREATE Scheduler Primary \QA.$SCHD CPU 0
COLD Purge any existing scheduler database files
OPEN U_255,255 P_BATCHCOM \QA.$:0:35:17105964
ADD SCHEDULER \QA.$:0:35:17105964 U_255,255
H_\QA.$ZTN0.#PTY4
CLOSE \QA.$:0:35:17105964 U_255,255
OPEN U_255,255 P_BATCHCOM \QA.$:0:59:17106988
START SCHEDULER \QA.$:0:59:17106988 U_255,255
H_\QA.$ZTN0.#PTY4
CREATE Scheduler Backup \QA.$SCHD CPU 1 U_255,255
H_\QA.$ZTN0.#PTY4
START \QA.$SCHD:17381165
CLOSE \QA.$:0:59:17106988 U_255,255
```

ALLOW ERRORS Command

Use the ALLOW ERRORS command to specify the number of errors allowed during execution of commands contained in a BATCHCOM input file. BATCHCOM stops when the number of errors exceeds the specified limit.

```
ALLO[W] num ER[RORS]
```

num

is an integer in the range 0 through 65535 specifying the number of errors allowed.

Considerations

- The ALLOW ERRORS command is available to all users.
- The ALLOW ERRORS command is effective only when included in a BATCHCOM input file. The command has no effect when entered interactively.
- If you do not specify an explicit ALLOW ERRORS command in a BATCHCOM input file, the implicit command ALLOW 0 ERRORS applies.
- The ALLOW ERRORS command applies only to events generating error messages (prefix *nnnn*-E). It does not apply to events generating informational messages (prefix *nnnn*-I) or warning messages (prefix *nnnn*-W). For details of NetBatch messages, see Appendix A, Messages.

When no more than *num* errors occur during command execution, BATCHCOM stops with completion code 1 (normal, voluntary termination with warning diagnostics). When the number of errors exceeds *num*, BATCHCOM stops with completion code 2 (abnormal, voluntary termination with fatal errors or diagnostics).

Note. HP recommends you specify an error limit for all BATCHCOM processes that are part of multiprocess jobs whose attributes include STOP-ON-ABEND ON. By specifying a limit covering all expected errors, you can prevent the scheduler from stopping jobs unnecessarily.

Example

This example shows BATCHCOM executing or trying to execute these commands from the file INFILE:

```
ALLOW 2 ERRORS
SUBMIT JOB A, IN A, OUT $S.#A, AT 20:30
SUBMIT JOB B, IN B, OUT $S.#B, CLASS OPS
SUBMIT JOB C, IN C, OUT $S.#C, AFTER 23:00
SUBMIT JOB D, IN D, OUT $S.#D, TAPDRIVES 1
SUBMIT JOB E, IN E, OUT $S.#E
```

The errors generated by the SUBMIT JOB commands show the effect of the ALLOW ERRORS command. The SHOW ERRORS command allows two errors, which causes

the BATCHCOM process to stop when a third error occurs. Note the completion code (2) when the process stops and the reported warnings (1) and errors (3).

```
> BATCHCOM /IN INFILE/ $ZBAT
ALLOW 2 ERRORS
SUBMIT JOB A, IN A, OUT $S.#A, AT 20:30
2056-E AT-ALLOWED is currently OFF; submit AFTER time
SUBMIT JOB B, IN B, OUT $S.#B, CLASS OPS
2105-E CLASS OPS does not exist
SUBMIT JOB C, IN C, OUT $S.#C, AFTER 23:00
Job C job number 125 submitted
0513-W IN file does not exist; create and secure as required
SUBMIT JOB D, IN D, OUT $S.#D, TAPDRIVES 1
SUBMIT JOB D, IN D, OUT $S.#D, TAPDRIVES 1
-^-0290-
E Invalid command SUBMIT JOB
STOPPED: 1,51
CPU time 0:00:00.350
2: Process terminated with fatal errors or diagnostics
Termination Info 2056
Subsystem TANDEM.9.D20
BATCHCOM: 1 warnings and 3 errors
>
```

ALTER ATTACHMENT-SET Command

Use the ALTER ATTACHMENT-SET command to change attachment-set attributes.

```
ALT[ER] [ ATTACHMENT-SET ] [ / OU[T] [ file-name ] / ]
{ [ ( user-ID ) ] attachment-set-ID | #CURRENT }
, attribute [ , attribute ]…
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

user-ID

specifies a user ID or a range of user IDs specified with the asterisk (*) and question mark (?) wild-card characters. (*user-ID* must be in *group-name.user-name* or *group-ID*,*user-ID* form.) The default is the user ID of the current user.

attachment-set-ID

is one of:

attachment-set-name

specifies an attachment-set name or a range of names when specified with the asterisk (*) and question mark (?) wild-card characters.

attachment-set-number

is a scheduler-generated number identifying an attachment set created by means of the #CURRENT variable.

*

specifies all attachment sets.

#CURRENT

is the attachment set specified by the #CURRENT variable. If #CURRENT has a null value, BATCHCOM displays this message:

```
0345-E Undefined substitution
```

attribute

is one of these attachment-set attributes (see Section 7, Attributes):

```
( ASSI[GN] ASSIGN-name , ASSIGN-attributes )
( DEFI[NE] DEFINE-name-1 , [ LIK[E] DEFINE-name-2 , ]
[ CLASS DEFINE-class , ]
DEFINE-attribute [ , DEFINE-attribute ]… )
( PA[RAM] PARAM-name PARAM-value )
SEC[URITY] " security "
TEM[PORARY] { OF[F] | ON }
```

Considerations

- The ALTER ATTACHMENT-SET command is available to all users, but you must have write access to the set you want to alter. For example, SUPER.FPP could alter set (SUPER.SUPER)P secured “NGNO” but not set (SUPER.NB)Q secured “AOAO.”
- The ALTER ATTACHMENT-SET command enables you to change an existing attachment set:
  - By adding ASSIGNs, DEFINEs, and PARAMs. For example:

```
1} CHANGEUSER SUPER.NB psswrd
1} INFO ATTACHMENT-SET STANDARD, ASSIGN, DEFINE, PARAM
ATTACHMENT-SET ATTRIBUTES for (SUPER.NB)STANDARD
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS,
VOLUME $A.NB
ASSIGN FILE-X, X
PARAM A 1
2} ALTER ATTACHMENT-SET STANDARD, (ASSIGN FILE-Y, Y),
(DEFINE =LOC, CLASS SPOOL, LOC $S), (PARAM B 2)
Attachment-set (SUPER.NB)STANDARD altered
```

```
3} INFO ATTACHMENT-SET STANDARD, ASSIGN, DEFINE, PARAM
ATTACHMENT-SET ATTRIBUTES for (SUPER.NB)STANDARD
attachments: DEFINE =LOC, CLASS SPOOL, LOC \SYS1.$S
DEFINE =_DEFAULTS, CLASS DEFAULTS,
VOLUME $A.NB
ASSIGN FILE-X, X
ASSIGN FILE-Y, Y
PARAM A 1
PARAM B 2
```

- By resetting ASSIGNs and PARAMs. For example:

```
7} CHANGEUSER FPP.MANAGER psswrd
7} INFO ATTACHMENT-SET OPS, ASSIGN, PARAM
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)OPS
attachments: ASSIGN BCKPS-DLY, $A.NB.DLY, EXCLUSIVE,
INPUT, EXT 256, BLOCK 4096
PARAM TIME 14:30
8} ALTER ATTACHMENT-SET OPS, (ASSIGN BCKPS-DLY,
$BIG1.NB.DLY), (PARAM TIME 15:15)
Attachment-set (FPP.MANAGER)OPS altered
9} INFO ATTACHMENT-SET OPS, ASSIGN, PARAM
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)OPS
attachments: ASSIGN BCKPS-DLY, $BIG1.NB.DLY
PARAM TIME 15:15
```

- By altering or removing attributes of DEFINEs. For example:

```
20} CHANGEUSER NB.USER psswrd
20} INFO ATTACHMENT-SET 19, DEFINE =TAPE
ATTACHMENT-SET ATTRIBUTES for (NB.USER)19
attachments: DEFINE =TAPE, CLASS TAPE, VOLUME
SCRATCH, LABELS ANSI, FILEID
CUSTRECORDS, DEVICE
\MELBDEV.$TAPE
21} ALTER ATTACHMENT-SET 19, (DEFINE =TAPE, FILEID,
DEVICE \MELBQAT.$TAPE1)
Attachment-set (NB.USER)19 altered
22} INFO ATTACHMENT-SET 19, DEFINE =TAPE
ATTACHMENT-SET ATTRIBUTES for (NB.USER)19
attachments: DEFINE =TAPE, CLASS TAPE, VOLUME
SCRATCH, LABELS ANSI, DEVICE
\MELBQAT.$TAPE1
```

  By omitting the value of a DEFINE attribute from the ALTER ATTACHMENT-SET command, you remove that attribute from the DEFINE.

- By altering SECURITY and TEMPORARY attributes. For example:

```
5} CHANGEUSER 255,255 psswrd
5} INFO ATTACHMENT-SET 23, SECURITY, TEMPORARY
ATTACHMENT-SET ATTRIBUTES for (SUPER.SUPER)23
security: "UUUU"
temporary: On
6} ALTER ATTACHMENT-SET 23, SECURITY "GOGO", TEMPORARY
OFF
Attachment-set (SUPER.SUPER)23 altered
```

```
7} INFO ATTACHMENT-SET 23, SECURITY, TEMPORARY
ATTACHMENT-SET ATTRIBUTES for (SUPER.SUPER)23
security: "GOGO"
temporary: Off
```

- BATCHCOM supports all classes of DEFINE available in the TACL environment. The default DEFINE class is MAP.
- The LIKE qualifier is available for DEFINEs only. Use it in the ALTER ATTACHMENT-SET command to specify DEFINEs whose attributes match those of DEFINEs specified earlier in the command. You also can specify DEFINEs whose attributes match those of existing DEFINEs in the target attachment set.
- To delete ASSIGNs, DEFINEs (except =_DEFAULTS), and PARAMs from an attachment set, use the DELETE ATTACHMENT-SET command. You also can use this command to delete an attachment set from a scheduler.
- The ALTER ATTACHMENT-SET command changes the attachment set specified by the #CURRENT variable to the altered attachment set. This example shows the state of the variable after various ALTER ATTACHMENT-SET commands. BATCHCOM displays an “undefined substitution” message when the variable has a null value.

  ```
  65} RESET ATTACHMENT-SET
  66} INFO ATTACHMENT-SET #CURRENT
  -^-0345-
  E Undefined substitution
  67} ALTER ATTACHMENT-SET (NB.USER)A1, …
  Attachment-set (NB.USER)A1 altered
  68} INFO ATTACHMENT-SET #CURRENT
  ATTACHMENT-SET ATTRIBUTES for (NB.USER)A1 .
  .
  69} ALTER ATTACHMENT-SET (NB.USER)B2, …
  Attachment-set (NB.USER)B2 altered
  70} INFO ATTACHMENT-SET #CURRENT
  ATTACHMENT-SET ATTRIBUTES for (NB.USER)B2 .
  .
  71} RESET ATTACHMENT-SET
  72} INFO ATTACHMENT-SET #CURRENT
  -^-0345-
  E Undefined substitution
  ```

  When you use the ALTER ATTACHMENT-SET command to change the #CURRENT variable, you change the actual attachment set the variable specifies.
- ADP (an abbreviation of ASSIGNs, DEFINEs, and PARAMs) is an alias of the object keyword ATTACHMENT-SET.
- You can omit the object keyword ATTACHMENT-SET from the ALTER ATTACHMENT-SET command only when ATTACHMENT-SET is the current assumed object. For more information, see ASSUME ATTACHMENT-SET Command on page 6-69.

Examples

- This example shows the ALTER ATTACHMENT-SET command altering a sort DEFINE by changing its attributes in the #CURRENT variable:

```
> LOGON NB.USER, psswrd
> INFO DEFINE =SORT, DETAIL
Define Name =SORT
CLASS SORT
SCRATCH \MELBDEV.$TRASH.NB.PURGEME
MODE AUTOMATIC
PRI 149
CPUS ALL
> BATCHCOM $ZBAT
1} SHOW ATTACHMENT-SET DEFINE =SORT
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =SORT, CLASS SORT, SCRATCH
\MELBDEV.$TRASH.NB.PURGEME, MODE
AUTOMATIC, PRI 149, CPUS ALL
2} INFO ATTACHMENT-SET #CURRENT, DEFINE =SORT
-^-0345-
E Undefined substitution
3} ADD ATTACHMENT-SET #CURRENT
Attachment-set (NB.USER)19 added
4} INFO ATTACHMENT-SET 19, DEFINE =SORT
ATTACHMENT-SET ATTRIBUTES for (NB.USER)19
attachments: DEFINE =SORT, CLASS SORT, SCRATCH
\MELBDEV.$TRASH.NB.PURGEME, MODE
AUTOMATIC, PRI 149, CPUS ALL
5} INFO ATTACHMENT-SET #CURRENT, DEFINE =SORT
ATTACHMENT-SET ATTRIBUTES for (NB.USER)19
attachments: DEFINE =SORT, CLASS SORT, SCRATCH
\MELBDEV.$TRASH.NB.PURGEME, MODE
AUTOMATIC, PRI 149, CPUS ALL
6} ALTER ATTACHMENT-SET #CURRENT, (DEFINE =SORT, MODE
MINTIME, PRI 110, NOTCPUS (2,3))
Attachment-set (NB.USER)19 altered
7} INFO ATTACHMENT-SET #CURRENT, DEFINE =SORT
ATTACHMENT-SET ATTRIBUTES for (NB.USER)19
attachments: DEFINE =SORT, CLASS SORT, SCRATCH
\MELBDEV.$TRASH.NB.PURGEME, MODE
MINTIME, PRI 110, CPUS ALL, NOTCPUS
(2,3)
8} INFO ATTACHMENT-SET 19, DEFINE =SORT
ATTACHMENT-SET ATTRIBUTES for (NB.USER)19
attachments: DEFINE =SORT, CLASS SORT, SCRATCH
\MELBDEV.$TRASH.NB.PURGEME, MODE
MINTIME, PRI 110, CPUS ALL, NOTCPUS
(2,3)
```

- This example shows the ALTER ATTACHMENT-SET command changing the value of PARAM DATE in all attachment sets containing that PARAM. The command also adds PARAM TIME to those sets.

```
51} CHANGEUSER 255,255 psswrd
51} INFO ATTACHMENT-SET (*.*)DAY, PARAM DATE, PARAM TIME
ATTACHMENT-SET ATTRIBUTES for (NB.USER)DAY
attachments: PARAM DATE 22JUN93
ATTACHMENT-SET ATTRIBUTES for (NB.MANAGER)DAY
attachments: PARAM DATE 22JUN93
ATTACHMENT-SET ATTRIBUTES for (SUPER.NB)DAY
attachments: PARAM DATE 22JUN93
52} ALTER ATTACHMENT-SET (*.*)DAY, (PARAM DATE 30AUG93),
(PARAM TIME 11:45)
Attachment-set (NB.USER)DAY altered
Attachment-set (NB.MANAGER)DAY altered
Attachment-set (SUPER.NB)DAY altered
53} INFO ATTACHMENT-SET (*.*)DAY, PARAM DATE, PARAM TIME
ATTACHMENT-SET ATTRIBUTES for (NB.USER)DAY
attachments: PARAM DATE 30AUG93
PARAM TIME 11:45
ATTACHMENT-SET ATTRIBUTES for (NB.MANAGER)DAY
attachments: PARAM DATE 30AUG93
PARAM TIME 11:45
ATTACHMENT-SET ATTRIBUTES for (SUPER.NB)DAY
attachments: PARAM DATE 30AUG93
PARAM TIME 11:45
```

- This example shows the LIKE qualifier adding a DEFINE to an attachment set:

```
12} CHANGEUSER SUPER.SUPER psswrd
12} ADD ATTACHMENT-SET OUT-SPEC, (DEFINE =OUT1, CLASS SPOOL,
BATCHNAME DAILY-CLEANUPS, HOLDAFTER ON, LOC
\MELBDEV.$S.#CLEANUP, MAXPRINTPAGES 150, OWNER "133,2",
SELPRI 6)
Attachment-set (SUPER.SUPER)OUT-SPEC added
13} INFO ATTACHMENT-SET OUT-SPEC, DEFINE =OUT1
ATTACHMENT-SET ATTRIBUTES for (SUPER.SUPER)OUT-SPEC
attachments: DEFINE =OUT1, CLASS SPOOL, LOC
\MELBDEV.$S.#CLEANUP, HOLDAFTER ON,
OWNER "133,2", SELPRI 6, BATCHNAME
DAILY-CLEANUPS, MAXPRINTPAGES 150
14} ALTER ATTACHMENT-SET OUT-SPEC, (DEFINE =OUT2, LIKE =OUT1,
BATCHNAME WEEKLY-CLEANUPS, LOC \MELBQAT.$S.#SUPER, OWNER
"255,133")
Attachment-set (SUPER.SUPER)OUT-SPEC altered
15} INFO ATTACHMENT-SET OUT-SPEC, DEFINE =OUT2
ATTACHMENT-SET ATTRIBUTES for (SUPER.SUPER)OUT-SPEC
attachments: DEFINE =OUT2, CLASS SPOOL, LOC
\MELBQAT.$S.#SUPER, HOLDAFTER ON,
OWNER "255,133", SELPRI 6, BATCHNAME
WEEKLY-CLEANUPS, MAXPRINTPAGES 150
```

- This example shows the execution of ADD ATTACHMENT-SET and ALTER ATTACHMENT-SET commands and lists scheduler log file events relating to those commands:

```
> LOGON 205,70, password
> BATCHCOM $SCHD; ADD ADP (205,100) QA-1, SECURITY "NNNN",
TEMPORARY OFF, (ASSIGN A, $A.NB.A), (DEFINE =_DEFAULTS,
VOLUME $DATA7.SCHD), (PARAM TIME 5:00)
Attachment-set (FPP.QA)QA-1 added
> BATCHCOM $SCHD; ALTER ATTACHMENT-SET (205,100) QA-1,
(PARAM DATE 21JUL93)
Attachment-set (FPP.QA)QA-1 altered
> FUP COPY LOGAAG,,SHARE .
.
ADD ATT-SET (205,100)QA-1 SECURITY "NNNN" TEMPORARY OFF
U_205,70 H_\MELBDEV.$ZTN0.#PTY4
UPDATE ATT-SET (205,100)QA-1 U_205,70 H_\MELBDEV.$ZTN0.#PTY4
```

ALTER CLASS Command

Use the ALTER CLASS command to change class attributes.

```
ALT[ER] [ CLASS ] [ / OU[T] [ file-name ] / ]
class-name , attribute
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

class-name

is the name of a class.

attribute

specifies this class attribute (see Section 7, Attributes):

```
INI[TIATION]BBC{( AALVS1(OF[F],ON) )
```

Considerations

- The ALTER CLASS command is available to NetBatch supervisors only.
- Changing the value of a class's INITIATION attribute from ON to OFF does not affect jobs from the class that are currently executing or over limit.
- JOBCLASS is an alias of the object keyword CLASS.

- You can omit the object keyword CLASS from the ALTER CLASS command only when CLASS is the current assumed object. For more information, see ASSUME CLASS Command later in this section.

Examples

- This example shows the ALTER CLASS command changing the value of the INITIATION attribute of class STORES:

```
3} INFO CLASS STORES
CLASS ATTRIBUTE for STORES
initiation: Off
4} ALTER CLASS STORES, INITIATION ON
Class STORES altered
5} INFO CLASS STORES
CLASS ATTRIBUTE for STORES
initiation: On
```

  The scheduler log file events resulting from the ALTER CLASS command from this example are:

```
ALTER CLASS STORES U_255,255 H_\MELBDEV.$ZTN0.#PTY4
UPDATE CLASS STORES S_ON U_255,255 H_\MELBDEV.$ZTN0.#PTY4
```

ALTER EXECUTOR Command

Use the ALTER EXECUTOR command to change executor attributes.

```
ALT[ER] [ EXECUTOR ] [ / OU[T] [ file-name ] / ]
executor-name-1 ,
{ LIK[E] executor-name-2 [ , attribute ]… }
{ attribute [ , attribute ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

executor-name-1

is the name of the executor you want to alter.

LIKE

specifies that all attributes are to match those of executor *executor-name-2*. *attribute* overrides *executor-name-2* attributes of the same type.

executor-name-2

is the name of an executor whose attributes you want executor executor-name-1 to match.

attribute

is one of these executor attributes (see Section 7, Attributes):

```
CL[ASS] BBC{( AALVS1( class-name,( class-name [ , class-name
]…),*) )
CP[U] cpu-number
```

attribute, when specified with LIKE, overrides *executor-name-2* attributes of the same type.

Considerations

- The ALTER EXECUTOR command is available to NetBatch supervisors only.
- The CLASS executor attribute lets you assign up to eight existing classes to an executor. (An existing class is one added to the scheduler by the ADD CLASS command.) You also can use the attribute to dissociate classes from an executor.
  - To assign a class to an executor, specify the name of the class and the names of the executor's pre-existing classes. If you specify only the name of the new class, the ALTER EXECUTOR command dissociates all other classes from the executor.
  - To dissociate a class from an executor, specify the names of the classes that are to remain assigned to the executor.
- The order in which you assign classes to an executor determines the order in which the scheduler scans the classes for jobs. For example, assigning classes B and A to an executor (in that order) makes the scheduler scan class B before it scans class A. Jobs in class B therefore run before jobs in class A.
- You can specify only one CPU per executor.
- When you alter an executor's CPU attribute, the change takes effect immediately unless the executor is in use by a job. In this case, the change does not take effect until the job finishes.
- You can omit the object keyword EXECUTOR from the ALTER EXECUTOR command only when EXECUTOR is the current assumed object. For more information, see ASSUME EXECUTOR Command on page 6-71.

Examples

- This example shows the ALTER EXECUTOR command altering the attributes of an executor. The command alters the CPU attribute to 0 and the CLASS attribute by dissociating class DEFAULT and reordering the remaining classes.

```
22} INFO EXECUTOR MANAGER
EXECUTOR ATTRIBUTES for MANAGER
cpu: 1
classes: X
Y
Z
DEFAULT
23} ALTER EXECUTOR MANAGER, CPU 0, CLASS (Z, Y, X)
Executor MANAGER altered
24} INFO EXECUTOR MANAGER
EXECUTOR ATTRIBUTES for MANAGER
cpu: 0
classes: Z
Y
X
```

  The scheduler log file event recording the ALTER EXECUTOR command from this example is:

```
ALTER EXECUTOR MANAGER U_255,255 H_\MELBDEV.$ZTN0.#PTY4
```

- This example shows the result of an ALTER EXECUTOR command that specifies a nonexistent class:

```
28} ALTER EXECUTOR LONG-JOBS, CLASS COMPILES
2105-E CLASS COMPILES does not exist
```

- This example shows the use of the ALTER EXECUTOR command's LIKE qualifier:

```
46} INFO EXECUTOR *
EXECUTOR ATTRIBUTES for EXEC00
cpu: 0
classes: URGENT-JOBS
STANDARD-CLASS
DEFAULT
EXECUTOR ATTRIBUTES for EXEC03
cpu: 3
classes: *
47} ALTER EXECUTOR EXEC03, LIKE EXEC00, CPU 3
Executor EXEC03 altered
48} INFO EXECUTOR EXEC03
EXECUTOR ATTRIBUTES for EXEC03
cpu: 3
classes: URGENT-JOBS
STANDARD-CLASS
DEFAULT
```

ALTER JOB Command

Use the ALTER JOB command to change job attributes.

```
ALT[ER] [ JOB ] [ / OU[T] [ file-name ] / ]
{ job-ID }
{ ( [ [NOT] job-ID ,]… [NOT] filter [ , [NOT] filter ]… ) }
{ , LIK[E] job-ID-2 [ , attribute ]… }
{ , attribute [ , attribute ]… }
[ { IN } ]
[ , REM[OVE] { J[OB]-L[OG] } ] …
[ { LIB } ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

job-ID-1

specifies the name, number, or range of names of the job or jobs you want to alter. To specify a range of job names, use the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all jobs and is the default in a command that includes *filter* but not *job-ID-1*.

NOT

excludes jobs that satisfy the specified job-selection criterion.

filter

specifies one of these job-selection criteria (see STATUS JOB Command on page 6-165):

```
A[TTACHMENT]-S[ET]BBC[( AALVS1([ (user-ID) ] attachment-set-
ID,#CURRENT) )
CL[ASS] class-ID
IN file-ID
STATE state
U[SER] user-ID
WAITO[N] master-job
```

LIKE

specifies that all attributes are to match those of job *job-ID-2*. *attribute* overrides *job-ID-2* attributes of the same type.

job-ID-2

is the name of a job whose attributes you want job *job-ID-1* or all jobs (*) to match.

attribute

is one of these job attributes (see Section 7, Attributes):

```
AF[TER] [ date ] [ time ]
AT [ date ] [ time ]
A[TTACHMENT]-S[ET]
[ attachment-set | ( attachment-set [ , attachment-set ]… )]
CA[LENDAR] [ file-name ]
CL[ASS] class-name
DES[CRIPTION] " [ string ] "
EV[ERY] [ ( weeks WEEK[S]
days D[AYS]
hours [ : mins ] [HOURS]
hours H[OURS] [ mins MIN[UTES] ]
crontab-entry) ]
E[XECUTOR]-P[ROGRAM] file-name
EXT[SWAP] { $volume-name | file-name }
HIG[HPIN] {OF[F] | ON }
HOLDBBC {OF[F] | ON }
HOLDA[FTER] {OF[F] | ON }
IF[FAILS]BBC {OF[F] | ON }
IN [ file-name ]
J[OB]-L[OG] [ log-file ]
J[OBID]-Z[ERO] {OF[F] | ON }
LIB [ file-name ]
LIM[IT] hours [ : mins ]
MAXPRINTL[INES] { number | NON[E] }
MAXPRINTP[AGES] { number | NON[E] }
ME[M] number
NA[ME] $process-name
OU[T] [ file-name ]
PF[S] number
PR[I] number
P[URGE]-[IN]-[FILE {OF[F] | ON }
REST[ART] {OF[F] | ON }
RUND {OF[F] | ON }
SA[VEABEND] {OF[F] | ON }
SEL[PRI] number
STAL[L] {OF[F] | ON }
STARTU[P] " param-set "
S[TOP]-[ON]-[ABEND] {OF[F] | ON }
SWA[P] {$volume-name | file-name }
TA[PEDRIVES] number
TERM $process-name
V[OLUME]{ \node. [ volume ] [ , " security " ]
[ \node. ] volume [ , " security " ]
[ \node. ] [ volume ] , " security " }
WAIT hours [ : mins ]
WAITO[N] [ job-name [ case ]
( job-name [ case ] [ , job-name [ case ] ]… ) ]
```

attribute, when specified with LIKE, overrides *job-ID-2* attributes of the same type.

`REMOVE`

removes the job's IN attribute, JOB-LOG attribute, or the library file from the job's executor program.

`IN`

removes the job's IN attribute, which makes the scheduler become the job's input file. If the job reads the file, it encounters file-system error 2 (operation not allowed). If the job writes the file, it writes to the scheduler's log file.

`JOB-LOG`

removes the job's JOB-LOG attribute, which makes the scheduler create a spooler-job log file if the job's OUT attribute specifies a spooler collector process. If the OUT attribute specifies a device, a disk file, or a nonspooler collector process, the scheduler does not create a log file.

`LIB`

removes the library file from the job's executor program. As a consequence, the program runs without a library. (Specifying LIB without a file name in a TACL RUN command has the same effect.)

Considerations

- The ALTER JOB command is available to all users, but these conditions apply:
  - You can alter all attributes of a job if the job has a disk input file to which you have write access. You can also alter all attributes of a job if the job's input file does not exist or is a device or a process, but you can do this only if you are the job's owner.
  - NetBatch supervisors can alter all attributes of any job except:

ATTACHMENT-SET	JOB-LOG	PURGE-IN-FILE
DESCRIPTION	JOBID-ZERO	RUND
EXECUTOR-PROGRAM	LIB	STARTUP
HIGHPIN	NAME	STOP-ON-ABEND
IN	OUT	VOLUME

  - The ALTER JOB command's * option lets you alter your own jobs only. To alter other users' jobs to which you have access, use the command's *job-ID-1* option.
- The ALTER JOB command updates a job's description in the scheduler database. Whether the updated description applies to the job when it runs depends on the job's processing state:
  - If the job's state is EVENT, READY, RUNNEXT, RUNNOW, SPECIAL-*n*, TAPE, or TIME, the updated description applies when the job runs.

- If the job's state is EXECUTING, OVER LIMIT, or SUSPENDED, the only attributes that can apply are DESCRIPTION, HOLDAFTER, IFFAILS, LIMIT, PURGE-IN-FILE, RESTART, SAVEABEND, STALL, and STOP-ON-ABEND. If the job is also recurrent, the updated description applies in full the next time the job runs. (A recurrent job has the CALENDAR or EVERY attribute.)

  When you alter an executing, over-limit, or suspended job, BATCHOM displays a warning message similar to:

  ```
  0517-W Job job-name is currently executing or
  suspended; altered attribute(s) might not affect
  current execution
  ```

- You can omit the object keyword JOB from the ALTER JOB command only when JOB is the current assumed object. For more information, see ASSUME JOB Command on page 6-72.

Examples

- This example shows the ALTER JOB command altering the EXECUTOR-PROGRAM and EVERY attributes of a job:

  ```
  2} INFO JOB BACKUPS, EXECUTOR-PROGRAM, EVERY
  JOB ATTRIBUTES for BACKUPS
  jobnumber: 94
  executor-program: \DEV.$SYSTEM.SYSTEM.BACKUP
  every: 1 DAYS
  next-runtime: 11OCT94 15:07:32
  3} ALTER JOB BACKUPS, EXECUTOR-PROGRAM TACL, EVERY 7 DAYS
  Job BACKUPS Jobnumber 94 altered
  4} INFO JOB BACKUPS, EXECUTOR-PROGRAM, EVERY
  JOB ATTRIBUTES for BACKUPS
  jobnumber: 94
  executor-program: \DEV.$SYSTEM.SYSTEM.TACL
  every: 7 DAYS
  ```

- This example shows the effects of various ALTER JOB commands executed by different users:

  ```
  34} STATUS JOB *
  JOB JOBNAME USERID LOG STATE CLASSNAME
  ---- ---------------- ------- ---- ----------- --------------
  347 T 205,70 21:30:00 STANDARD-CLASS
  348 U 205,70 2154 SUSPENDED STANDARD-CLASS
  349 V 205,70 1:Hold STANDARD-CLASS
  350 W 255,205 2156 EXECUTING STANDARD-CLASS
  351 X 255,205 2158 READY STANDARD-CLASS
  352 Y 205,100 EVENT STANDARD-CLASS
  353 Z 133,2 EVENT STANDARD-CLASS
  35} CHANGEUSER 205,70 password
  35} ALTER JOB *, SELPRI 7, PRI 190
  Job T job number 347 altered
  Job U job number 348 altered
  0517-W Job U is currently executing or suspended; altered
  ```

```
attribute(s) might not affect current execution
Job V job number 349 altered
36} ALTER JOB 353, SELPRI 1, PRI 119
2132-E Your user code does not give you access to Z
37} CHANGEUSER 255,205 password
37} ALTER JOB *, OUT $S.#ZBAT
Job W job number 350 altered
0517-W Job W is currently executing or suspended; altered
attribute(s) might not affect current execution
Job X job number 351 altered
38} ALTER JOB 352, AFTER 23:59
Job Y job number 352 altered
39} CHANGEUSER 133,2 password
39} ALTER JOB Z, ATTACHMENT-SET #CURRENT
Job Z job number 353 altered
```

- This example shows execution of SUBMIT JOB and ALTER JOB commands and the resulting events recorded in the scheduler log file:

```
> BATCHCOM $ZBAT; SUBMIT JOB, EXECUTOR-PROGRAM DELAY, STARTUP
"1 MINS", AFTER 11OCT94 19:00
Job ZBAT-0097 Jobnumber 97 submitted
> BATCHCOM $ZBAT; ALTER JOB 97, AT 0:00
Job ZBAT-0097 Jobnumber 97 altered
> FUP COPY LOGABL,,SHARE .
.
ADD JOB ZBAT-0097 C_DEFAULT:3 J_97 U_255,255
H_\MELBDEV.$ZTN0.#PTY4
UPDATE JOB ZBAT-0097 C_DEFAULT:3 J_97 U_255,255
H_\MELBDEV.$ZTN0.#PTY4
LIST JOB ZBAT-0097 RUNNOW J_97 U_255,255
H_\MELBDEV.$ZTN0.#PTY4
ADD EXECUTOR __TEMP_EXEC_97 CPU 2
BEGIN JOB (SUPER.SUPER)ZBAT-0097:1 E___TEMP_EXEC_97 L_693
J_97 P_DELAY \MELBDEV.$Z723:17397037 U_255,255
UPDATE EXECUTOR __TEMP_EXEC_97
LIST JOB ZBAT-0097 EXECUTING J_97
START EXECUTOR-PROGRAM U_255,255 J_97 P_DELAY
\MELBDEV.$Z723:17397037
STOP CC_0 EXECUTOR-PROGRAM J_97 \MELBDEV.$Z723:17397037
DELETE EXECUTOR __TEMP_EXEC_97
FINISH JOB ZBAT-0097 T_0:0:0:12 J_97 P_DELAY
DELETE JOB ZBAT-0097 J_97
```

ALTER SCHEDULER Command

Use the ALTER SCHEDULER command to change scheduler attributes.

```
ALT[ER] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
, attribute [ , attribute ]…
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

attribute

is one of these scheduler attributes (see Section 7, Attributes):

```
AT-A[LLOWED] { OF[F] | ON }
B[ACKUPCPU] { cpu-number-1 [ , cpu-number-2 ] | * }
CAT[CHUP] { OF[F] | ON }
D[EFAULT]-C[LASS] class-name
D[EFAULT]-E[XECUTOR]-[PROGRAM] file-name
D[EFAULT]-H[IGHPIN] { OF[F] | ON }
D[EFAULT]-MAXPRINTL[INES] { number | NON[E] }
D[EFAULT]-MAXPRINTP[AGES] { number | NON[E] }
D[EFAULT]-O[UT] file-name
D[EFAULT]-P[RI] number
D[EFAULT]-SE[LPRI] number
D[EFAULT]-ST[ALL] { OF[F] | ON }
D[EFAULT]-[STOP]-[ON]-[ABEND] { OF[F] | ON }
EM[S] { OF[F] | ON }
INI[TIATION] { OF[F] | ON }
LO[CALNAMES] [\remote-node
( \remote-node [ , \remote-node ]… ) ]
M[AX]-[CONCURRENT]-[JOBS] max-concurrent-jobs
[ , max-temporary-executors ]
M[AX]-[PRI] number
S[UBMIT]-A[LLOWED] { OF[F] | ON }
TA[PEDRIVES] number
```

Considerations

- The ALTER SCHEDULER command is available to NetBatch supervisors only.
- Except for BACKUPCPU, attributes changed by ALTER SCHEDULER take effect on command execution. The command changes this attribute's value in the scheduler database, but does not force an actual change of backup CPUs. (Such a change occurs only when the scheduler's primary or backup process stops.)
- You can omit the object keyword SCHEDULER from the ALTER SCHEDULER command only when SCHEDULER is the current assumed object. For more information, see ASSUME SCHEDULER Command on page 6-73.

Examples

- This example shows the effect of an ALTER SCHEDULER command:

```
10} CHANGEUSER 255,255 password
10} INFO SCHEDULER, AT-ALLOWED, DEFAULT-PRI, DEFAULT-SELPRI,
DEFAULT-OUT, DEFAULT-CLASS
SCHEDULER ATTRIBUTES
at-allowed: Off
default-pri: 119
default-selpri: 2
default-out: \MELBDEV.$S.#ZBATJOB
default-class: DEFAULT
11} ALTER SCHEDULER, AT-ALLOWED ON, DEFAULT-PRI 100, DEFAULT-
SELPRI 2, DEFAULT-OUT $S.#NBJOB, DEFAULT-CLASS STANDARD-
CLASS
Scheduler altered
12} INFO SCHEDULER, AT-ALLOWED, DEFAULT-PRI, DEFAULT-SELPRI,
DEFAULT-OUT, DEFAULT-CLASS
SCHEDULER ATTRIBUTES
at-allowed: On
default-pri: 100
default-selpri: 2
default-out: \MELBDEV.$S.#NBJOB
default-class: STANDARD-CLASS
```

  The scheduler log file event resulting from the ALTER SCHEDULER command from the previous example is:

```
UPDATE SCHEDULER \MELBDEV.$:0:61:17098028 U_255,255
H_\MELBDEV.$ZTN0.#PTY4
```

- This example shows the ALTER SCHEDULER command changing a scheduler's BACKUPCPU attribute. The example shows the changed attribute and the effect of the change when the scheduler's backup process restarts after stopping.

```
> BATCHCOM $ZBAT; INFO SCHEDULER, BACKUPCPU
SCHEDULER ATTRIBUTES
backupcpu: *
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 2,36 0,93 $C2
> BATCHCOM $ZBAT; ALTER SCHEDULER, BACKUPCPU 3
Scheduler altered
18> BATCHCOM $ZBAT; INFO SCHEDULER, BACKUPCPU
SCHEDULER ATTRIBUTES
backupcpu: 3
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 2,36 0,93 $C2
> STOP 0,93
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 2,36 3,23 $C2
```

ASSUME ATTACHMENT-SET Command

Use the ASSUME ATTACHMENT-SET command to set the default object keyword for later commands to ATTACHMENT-SET. BATCHCOM substitutes this object keyword in all commands you enter that do not include an object keyword.

```
ASSU[ME] A[TTACHMENT]-S[ET]
```

Considerations

- The ASSUME ATTACHMENT-SET command is available to all users.
- Attachment-set commands you enter after the ASSUME ATTACHMENT-SET command do not have to include the object keyword ATTACHMENT-SET.
- Class, executor, job, and scheduler commands you enter after the ASSUME ATTACHMENT-SET command must specify the object to which they apply.
- The default object keyword when you start a BATCHCOM session is JOB.
- ADP (an abbreviation of ASSIGNs, DEFINEs, and PARAMs) is an alias of the object keyword ATTACHMENT-SET.

Example

The ASSUME command in this example specifies ATTACHMENT-SET as the default object keyword for later commands. This specification enables the omission of the keyword ATTACHMENT-SET from the SET and SHOW commands operating on that object. The SUBMIT command fails because it does not specify an object (that is, JOB), so BATCHCOM interprets it as an invalid command.

```
13} ASSUME ATTACHMENT-SET
14} SET SECURITY "NONO"
15} SHOW SECURITY
ATTACHMENT-SET ATTRIBUTES
security: "NONO"
16} SUBMIT TODAY, IN NBFILES.TODAY, OUT $S.#TODAY
-^-0290-
E Invalid command SUBMIT ATTACHMENT-SET
```

ASSUME CLASS Command

Use the ASSUME CLASS command to set the default object keyword for later commands to CLASS. BATCHCOM substitutes this object keyword in all commands you enter that do not include an object keyword.

```
ASSU[ME] CL[ASS]
```

Considerations

- The ASSUME CLASS command is available to all users.
- Class commands you enter after the ASSUME CLASS command do not have to include the object keyword CLASS.
- Attachment-set, executor, job, and scheduler commands you enter after the ASSUME CLASS command must specify the object to which they apply.
- The default object keyword when you start a BATCHCOM session is JOB.
- JOBCLASS is an alias of the object keyword CLASS.

Example

This example shows the effect of the ASSUME CLASS command. Because the command sets the default keyword to CLASS, BATCHCOM interprets the ADD command as ADD CLASS and executes it as such. The STATUS command fails because it does not specify an object keyword. BATCHCOM interprets it as an invalid command.

```
13} ASSUME CLASS
14} ADD TEMP-1, INITIATION OFF
Class TEMP-1 added
15} STATUS
0290-E Invalid command STATUS CLASS
```

ASSUME EXECUTOR Command

Use the ASSUME EXECUTOR command to set the default object keyword for later commands to EXECUTOR. BATCHCOM substitutes this object keyword in all commands you enter that do not include an object keyword.

```
ASSU[ME] EXE[CUTOR]
```

Considerations

- The ASSUME EXECUTOR command is available to all users.
- Executor commands you enter after the ASSUME EXECUTOR command do not have to include the object keyword EXECUTOR.
- Attachment-set, class, job, and scheduler commands you enter after the ASSUME EXECUTOR command must specify the object to which they apply.
- The default object keyword when you start a BATCHCOM session is JOB.

Example

This example shows the failure of a START command executed at the beginning of a session. The command fails because BATCHCOM interprets it as the invalid command START JOB (JOB being the default keyword when a session begins). The example then shows the ASSUME EXECUTOR command enabling execution of the START command by specifying the object keyword EXECUTOR.

```
1} START *
-^-0290-
E Invalid command START JOB
2} ASSUME EXECUTOR
3} START *
Executor EXEC-0 started
Executor EXEC-1 started
Executor EXEC-2 started
Executor EXEC-3 started
```

ASSUME JOB Command

Use the ASSUME JOB command to set the default object keyword for later commands to JOB. BATCHCOM substitutes this object keyword in all commands you enter that do not include an object keyword.

```
ASSU[ME] JOB
```

Considerations

- The ASSUME JOB command is available to all users.
- Job commands you enter after the ASSUME JOB command do not have to include the object keyword JOB.
- Attachment-set, class, executor, and scheduler commands you enter after the ASSUME JOB command must specify the object to which they apply.
- The default object keyword when you start a BATCHCOM session is JOB.

Examples

- The object keyword assumed by BATCHCOM in this example is JOB (the default at the start of a session):

```
34> BATCHCOM
1} SHOW
JOB ATTRIBUTES
volume: $DATA7.NB, "AAAO"
jobid-zero: Off
user: 205,70
```

- This example shows the ASSUME JOB command specifying the object keyword for a series of SET JOB and SUBMIT JOB commands. The last ALTER command fails because it does not specify the object to which it applies (CLASS). BATCHCOM therefore interprets it as an invalid ALTER JOB command instead of the intended ALTER CLASS command.

```
29} ALTER CLASS DEFAULT, INITIATION OFF
Class DEFAULT altered
30} ASSUME JOB
31} SET CLASS DEFAULT
32} SUBMIT J04, IN D4; SUBMIT J05, IN D5; SUBMIT J06, IN D6
Job J04 job number 24 submitted
Job J05 job number 25 submitted
Job J06 job number 26 submitted
33} ALTER DEFAULT, INITIATION ON
-^-0290-
E Invalid command ALTER JOB
```

ASSUME SCHEDULER Command

Use the ASSUME SCHEDULER command to set the default object keyword for later commands to SCHEDULER. BATCHCOM substitutes this object keyword in all commands you enter that do not include an object keyword.

```
ASSU[ME] SC[HEDULER]
```

Considerations

- The ASSUME SCHEDULER command is available to all users.
- Scheduler commands you enter after the ASSUME SCHEDULER command do not have to include the object keyword SCHEDULER.
- Attachment-set, class, executor, and job commands you enter after the ASSUME SCHEDULER command must specify the object to which they apply.
- The default object keyword when you start a BATCHCOM session is JOB.

Example

This example shows the effect of the ASSUME SCHEDULER command. Because the command sets the default keyword to SCHEDULER, BATCHCOM interprets the STATUS command as STATUS SCHEDULER and executes it as such. The SUBMIT command fails because it does not specify the object keyword JOB. BATCHCOM interprets it as an invalid command.

```
1} ASSUME SCHEDULER
2} STATUS
SCHEDULER STATUS
Process : \MELRISK.$ZBAT Primary : 0,65 Backup : 1,45
Database: \MELRISK.$QA.ZBAT
Logfile : \MELRISK.$QA.ZBAT.LOGAAB
Time : 02SEP94 16:13:09 .
.
3} SUBMIT CLEANUP-JOB, IN CLEANUP, OUT $S.#CLEANUP
-^-0290-
E Invalid command SUBMIT SCHEDULER
```

CHANGEUSER Command

Use the CHANGEUSER command to log on as a different user during a BATCHCOM session. The logon takes place on the system where the BATCHCOM process is running.

```
CH[ANGEUSER] user-ID [ password ]
```

user-ID

is a user ID in either of these forms:

group-name.user-name

group-name

is the name of the group to which the user belongs.

user-name

is the name of the user assigned to the group.

An example of a user ID in *group-name.user-name* form is NB.USER.

group-ID,user-ID

group-ID

is a number between 1 and 255 identifying the group to which the user belongs.

user-ID

is a number between 1 and 255 identifying the user assigned to the group.

An example of a user ID in *group-ID,user-ID* form is 205, 70.

password

is the password of the user specified by user-ID.

Considerations

- The CHANGEUSER command is available to all users.
- BATCHCOM treats all text following *user-ID* as *password*. CHANGEUSER must therefore be the last command you enter in a command line.
- The logged-on user when a BATCHCOM session starts and after the session ends is the user who started the session.
- The CHANGEUSER command does not change the node, volume, subvolume, and file security defaults that are current when you execute the command.

- If you specify an incorrect user ID or password, this message appears:

  ```
  0300-E USER not found or incorrect password for LOGON
  ```

  BATCHCOM delays the return of its prompt for one minute if you enter an incorrect user ID or password three times in succession.

- BATCHCOM does not store the CHANGEUSER command in its history buffer.

- You can disable the CHANGEUSER command with the NBFLAGS procedure. For details, see on page 2-3.

Examples

- This example shows use of the CHANGEUSER command to log on the super ID (255,255) and enable execution of the SHUTDOWN SCHEDULER command. It also shows how the CHANGEUSER command does not change the user logged on to the TACL environment.

  ```
  37> WHO .
  .
  Userid: 133,2 Username: NB.USER Security: "NCNC"
  38> BATCHCOM $SCHD
  1} SHUTDOWN SCHEDULER
  2132-E Your user code does not give you access to SHUTDOWN
  SCHEDULER
  2} CHANGEUSER 255,255 psswrd
  2} SHUTDOWN SCHEDULER
  Scheduler shutting down
  3} EXIT
  39> WHO .
  .
  Userid: 133,2 Username: NB.USER Security: "NCNC"
  ```

- This example shows the result of a CHANGEUSER command that is the first of two semicolon-separated commands. The command fails because BATCHCOM interprets all text following the specified user ID as the user's password. The example then shows BATCHCOM successfully executing the commands after their entry on separate lines. The example also shows the successful execution of a CHANGEUSER command entered last in a command series.

  ```
  8} CHANGEUSER 255,205 psswrd1; SUBMIT JOB X
  -^-0300-
  E USER not found or incorrect password for LOGON
  8} CHANGEUSER 255,205 psswrd1
  8} SUBMIT JOB X
  Job X job number 175 submitted
  9} SUBMIT JOB Y; CHANGEUSER 205,70 psswrd2
  Job Y job number 176 submitted
  ```

COMMENT Command

Use the COMMENT command at the beginning of a line in a BATCHCOM input file to cause BATCHCOM to ignore the line.

```
COM[MENT] [ comment-text ]
```

comment-text

is any text, including braces and square brackets.

Considerations

- The COMMENT command is available to all users.
- You can enter the COMMENT command only at the beginning of a line in a BATCHCOM input file.
- Comment text finishes at the end of a line. To continue the comment on the next line, do one of:
  - Type the line continuation character (&) at the end of the line you want to continue, and then press RETURN.
  - Enter the COMMENT command at the beginning of the new line.

Note. You also can use the line continuation character to continue a line during an interactive session. After you enter the character and press RETURN, a double-braces prompt (}}) appears indicating the continued line.

- Double equals (==) is an alias of the command keyword COMMENT.

Examples

- This example shows the effect of the COMMENT command and line continuation character in the input file ABC. In the example, BATCHCOM executes the first and fourth SUBMIT JOB commands but ignores the second and third.

```
> FUP COPY ABC
SUBMIT JOB J1, IN INFILE1
COMMENT SUBMIT JOB J2, IN INFILE2 &
*** Run jobs J2 and J3 on Mondays and Tuesdays only *** &
SUBMIT JOB J3, IN INFILE3
SUBMIT JOB J4, IN INFILE4
> BATCHCOM /IN ABC/ $SCHD
SUBMIT JOB J1, IN INFILE1
Job J1 job number 10 submitted
COMMENT SUBMIT JOB J2, IN INFILE2 &
*** Run jobs J2 and J3 on Mondays and Tuesdays only *** &
SUBMIT JOB J3, IN INFILE3
SUBMIT JOB J4, IN INFILE4
Job J4 job number 11 submitted
```

- This example shows the line continuation character used during an interactive session and the resulting double-braces prompts indicating the continued lines:

```
1} SUBMIT JOB FRIDAY, &
}} IN $DATA7.NB.WEEKEND, &
}} OUT \MELBDEV.$S.#FRIDAY, &
}} EXECUTOR-PROGRAM $SYSTEM.SYSTEM.FUP, &
}} AFTER 23:00, &
}} WAITON THURSDAY
Job FRIDAY job number 127 submitted
```

DELETE ATTACHMENT-SET Command

Use the DELETE ATTACHMENT-SET command to delete attachment sets from a scheduler. You also can use the command to delete specified ASSIGNs, DEFINEs (except =_DEFAULTS), and PARAMs from attachment sets.

```
D[ELETE] [ ATTACHMENT-SET ] [ / OU[T] [ file-name ] / ]
[ ( user-ID ) ] attachment-set-IDBBC[( AALVS1([ , attribute
]…,, DET[AIL]) )
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

user-ID

specifies a user ID or a range of user IDs specified with the asterisk (*) and question mark (?) wild-card characters. (*user-ID* must be in *group-name.user-name* or *group-ID*,*user-ID* form.) The default is the user ID of the current user.

attachment-set-ID

is one of:

attachment-set-name

specifies an attachment-set name or a range of names when specified with the asterisk (*) and question mark (?) wild-card characters.

attachment-set-number

is a scheduler-generated number identifying an attachment set created by means of the #CURRENT variable.

*

specifies all attachment sets.

attribute

is one of these attachment-set attributes:

```
ASSI[GN] [ ASSIGN-name ]
DEFI[NE] [ DEFINE-name ]
PA[RAM] [ PARAM-name ]
```

attribute causes BATCHCOM to delete only the specified attribute from the attachment sets specified by attachment-set-ID. If you omit *attribute* and do not specify DETAIL, BATCHCOM deletes all attachment sets specified by *attachment-set-ID.*

You can use *ASSIGN-name*, *DEFINE-name*, or *PARAM-name* to specify an ASSIGN, DEFINE, or PARAM name or a range of names. To specify a range of names, use the asterisk (*) and question mark (?) wild-card characters. If you specify ASSIGN, DEFINE, or PARAM, but omit *ASSIGN-name*, *DEFINE-name*, or *PARAM-name*, BATCHCOM deletes all ASSIGNs, DEFINEs (except =_DEFAULTS), or PARAMs.

attribute cannot specify the SECURITY or TEMPORARY attributes.

DETAIL

causes BATCHCOM to delete all ASSIGNs, DEFINEs (except =_DEFAULTS), and PARAMs from the attachment sets specified by *attachment-set-ID*. If you omit DETAIL and do not specify *attribute*, BATCHCOM deletes all attachment sets specified by *attachment-set-ID*.

DETAIL does not delete the SECURITY and TEMPORARY attributes.

Considerations

- The DELETE ATTACHMENT-SET command is available to all users, but:
  - To delete an attachment set, you must have purge access to it. For example, NB.USER could delete attachment set (SUPER.SUPER)X secured “NNNN” but not attachment set (NB.MANAGER)Y secured “GGGO.”
  - To delete ASSIGNs, DEFINEs, and PARAMs from an attachment set, you must have write access to it. For example, only super-group users (255, n) could delete ASSIGNs, DEFINEs, or PARAMs from attachment set (SUPER.NB)Z secured “AGAO.”
- ADP (an abbreviation of ASSIGNs, DEFINEs, and PARAMs) is an alias of the object keyword ATTACHMENT-SET.
- You can omit the object keyword ATTACHMENT-SET from the DELETE ATTACHMENT-SET command only when ATTACHMENT-SET is the current assumed object. For more information, see ASSUME ATTACHMENT-SET Command on page 6-69.

Examples

- This example shows the results of various DELETE ATTACHMENT-SET commands executed by different users:

```
42} CHANGEUSER SUPER.NB psswrd
42} INFO ATTACHMENT-SET A1, DETAIL
ATTACHMENT-SET ATTRIBUTES for (SUPER.NB)A1
security: "GGGO"
temporary: Off
attachments: DEFINE =MAP, CLASS MAP, FILE \A.$D6.NB.MYFILE
DEFINE =TAPE, CLASS TAPE, LABELS OMITTED,
DEVICE \A.$TAPE
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
ASSIGN FILE-A, $NB.MYFILES.A
ASSIGN FILE-B, $NB.MYFILES.B
PARAM COUNTRY USA
43} CHANGEUSER NB.USER psswrd
43} DELETE ATTACHMENT-SET (SUPER.NB)A1
2132-E Your user code does not give you access to
(SUPER.NB)A1
44} DELETE ATTACHMENT-SET (SUPER.NB)A1, DEFINE =?AP*, ASSIGN
2132-E Your user code does not give you access to
(SUPER.NB)A1
45} CHANGEUSER SUPER.FPP psswrd
45} DELETE ATTACHMENT-SET (SUPER.NB)A1, DEFINE =?AP*, ASSIGN
Attachment-set (SUPER.NB)A1 altered
46} INFO ATTACHMENT-SET (SUPER.NB)A1, DETAIL
ATTACHMENT-SET ATTRIBUTES for (SUPER.NB)A1
security: "GGGO"
temporary: Off
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
PARAM COUNTRY USA
47} DELETE ATTACHMENT-SET (SUPER.NB)A1, DETAIL
Attachment-set (SUPER.NB)A1 altered
0536-I =_DEFAULTS cannot be deleted
48} INFO ATTACHMENT-SET (SUPER.NB)A1, DETAIL
ATTACHMENT-SET ATTRIBUTES for (SUPER.NB)A1
security: "GGGO"
temporary: Off
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
49} CHANGEUSER SUPER.NB psswrd
49} DELETE ATTACHMENT-SET A1; INFO ATTACHMENT-SET A1
Attachment-set (SUPER.NB)A1 deleted
2172-E (SUPER.NB)A1 does not exist
```

- This example shows a DELETE ATTACHMENT-SET command deleting ASSIGNs FRIDAY, MONDAY, and OUTPUT-FILE from NB.USER's attachment sets SLS-FRIDAY-JOBS and SLS-MONDAY-JOBS. The command does not delete ASSIGNs

from attachment set SLS-WEDNESDAY-JOBS because the set's name is outside the specified range.

```
25} CHANGEUSER NB.USER psswrd
25} INFO ATTACHMENT-SET SL?-*, ASSIGN
ATTACHMENT-SET ATTRIBUTES for (NB.USER)SLS-FRIDAY-JOBS
attachments: ASSIGN AFTERNOON, $A.NB.FRDYPM
ASSIGN FRIDAY, $A.NB.FRDY
ASSIGN MORNING, $A.NB.FRDYAM
ASSIGN OUTPUT-FILE, $TRASH.PURGEME
ATTACHMENT-SET ATTRIBUTES for (NB.USER)SLS-MONDAY-JOBS
attachments: ASSIGN AFTERNOON, $A.NB.MNDYPM
ASSIGN MONDAY, $A.NB.MNDY
ASSIGN MORNING, $A.NB.MNDYAM
ASSIGN OUTPUT-FILE, $TRASH.PURGEME
ATTACHMENT-SET ATTRIBUTES for (NB.USER)SLS-WEDNESDAY-JOBS
attachments: ASSIGN AFTERNOON, $A.NB.WDNSDYPM
ASSIGN MORNING, $A.NB.WDNSDYAM
ASSIGN OUTPUT-FILE, $TRASH.PURGEME
ASSIGN WEDNESDAY, $A.NB.WDNSDY
ATTACHMENT-SET ATTRIBUTES for (NB.USER)SLT-WEEKEND-JOBS
attachments: ASSIGN AFTERNOON, $A.NB.WKNDPM
ASSIGN MORNING, $A.NB.WKNDAM
ASSIGN WEEKEND, $A.NB.WKND
26} DELETE ATTACHMENT-SET SLS-???DAY*, ASSIGN *DAY, ASSIGN O*
Attachment-set (NB.USER)SLS-FRIDAY-JOBS altered
Attachment-set (NB.USER)SLS-MONDAY-JOBS altered
27} INFO ATTACHMENT-SET SLS*, ASSIGN
ATTACHMENT-SET ATTRIBUTES for (NB.USER)SLS-FRIDAY-JOBS
attachments: ASSIGN AFTERNOON, $A.NB.FRDYPM
ASSIGN MORNING, $A.NB.FRDYAM
ATTACHMENT-SET ATTRIBUTES for (NB.USER)SLS-MONDAY-JOBS
attachments: ASSIGN AFTERNOON, $A.NB.MNDYPM
ASSIGN MORNING, $A.NB.MNDYAM
ATTACHMENT-SET ATTRIBUTES for (NB.USER)SLS-WEDNESDAY-JOBS
attachments: ASSIGN AFTERNOON, $A.NB.WDNSDYPM
ASSIGN MORNING, $A.NB.WDNSDYAM
ASSIGN OUTPUT-FILE, $TRASH.PURGEME
ASSIGN WEDNESDAY, $A.NB.WDNSDY
```

- This example shows execution of a DELETE ATTACHMENT-SET command and the resulting events recorded in the scheduler log file:

```
> BATCHCOM $ZBAT; DELETE ATTACHMENT-SET (133,2)*
Attachment-set (NB.USER)ADP1 deleted
Attachment-set (NB.USER)ADP2 deleted
Attachment-set (NB.USER)ADP3 deleted
> FUP COPY LOGABL,,SHARE .
.
DELETE ATT-SET (133,2)ADP1 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
DELETE ATT-SET (133,2)ADP2 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
DELETE ATT-SET (133,2)ADP3 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
```

DELETE CLASS Command

Use the DELETE CLASS command to delete classes from a scheduler.

```
D[ELETE] [ CLASS ] [ / OU[T] [ file-name ] / ] class-name
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

class-name

is the name of a class.

Considerations

- The DELETE CLASS command is available to NetBatch supervisors only.
- Before deleting a class, you must dissociate it from all executors to which it belongs by using the ALTER EXECUTOR command.
- Deleting a class does not delete the class's jobs. The jobs are never eligible for execution unless you do one of:
  - Reassign them to existing classes
  - Add a class with the same name as the deleted class
- JOBCLASS is an alias of the object keyword CLASS.
- You can omit the object keyword CLASS from the DELETE CLASS command only when CLASS is the current assumed object. For more information, see ASSUME CLASS Command on page 6-70.

Examples

- This example shows the deletion of a class that does not belong to any executors:

```
90} DELETE CLASS ACT
Class ACT deleted
```

  The scheduler log file event resulting from the DELETE CLASS command from the previous example is:

```
DELETE CLASS ACT U_255,255 H_\MELBDEV.$ZTN0.#PTY4
```

- This example shows the message BATCHCOM displays when you try to delete a class belonging to one or more executors:

```
10} INFO EXECUTOR *
EXECUTOR ATTRIBUTES for EX0
```

```
cpu: 0
classes: DEFAULT
STANDARD-CLASS
EXECUTOR ATTRIBUTES for EX1
cpu: 1
classes: SPARE-PARTS-DEPT
11} DELETE CLASS SPARE-PARTS-DEPT
2104-E CLASS SPARE-PARTS-DEPT is in use by one or more
executors. To delete the CLASS, you must remove it
from the executors first.
```

- This example shows events leading up to and including the deletion of class DEVELOPMENT. These events include dissociation of the class from the two executors to which it belongs and from a job.

```
50} INFO EXECUTOR *
EXECUTOR ATTRIBUTES for EX1
cpu: 1
classes: DEVELOPMENT
PRODUCTION
EXECUTOR ATTRIBUTES for EX2
cpu: 2
classes: DEVELOPMENT
EXECUTOR ATTRIBUTES for EX3
cpu: 3
classes: DEFAULT
51} STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- -----------
90 BILL-OF-MATERIALS 255,255 19:45:00 DEVELOPMENT
52} ALTER EXECUTOR EX1, CLASS PRODUCTION
Executor EX1 altered
53} DELETE EXECUTOR EX2
Executor EX2 deleted
54} ALTER JOB 90, CLASS DEFAULT
Job BILL-OF-MATERIALS job number 90 altered
55} DELETE CLASS DEVELOPMENT
Class DEVELOPMENT deleted
```

DELETE EXECUTOR Command

Use the DELETE EXECUTOR command to delete executors from a scheduler.

```
D[ELETE] [ EXECUTOR ] [ / OU[T] [ file-name ] / ]
executor-name
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

executor-name

is the name of an executor.

Considerations

- The DELETE EXECUTOR command is available to NetBatch supervisors only.
- A job using an executor that is the subject of a DELETE EXECUTOR command finishes before the scheduler deletes the executor. The executor's state pending deletion in this circumstance is DELETE.
- After deleting an executor, you must reassign classes that were unique to it to other executors. If you do not do this, the scheduler never scans those classes for jobs. To reassign classes, use the ADD EXECUTOR or ALTER EXECUTOR command.
- You can omit the object keyword EXECUTOR from the DELETE EXECUTOR command only when EXECUTOR is the current assumed object. For more information, see ASSUME EXECUTOR Command on page 6-71.

Examples

- This example shows BATCHCOM's response to a DELETE EXECUTOR command:

```
25} DELETE EXECUTOR CPU2-EXEC
Executor CPU2-EXEC deleted
```

- The scheduler log file event resulting from the DELETE EXECUTOR command from the previous example is:

```
DELETE EXECUTOR CPU2-EXEC U_255,255 H_\MELBDEV.$ZTN0.#PTY4
```

- This example shows the deletion of an executor (CPU0-EXEC) following the reassignment of the executor's unique class:

```
40} INFO EXECUTOR *, CLASS
EXECUTOR ATTRIBUTES for CPU0-EXEC
classes: A
DEFAULT
EXECUTOR ATTRIBUTES for CPU1-EXEC
classes: DEFAULT
41} ALTER EXECUTOR CPU1-EXEC, CLASS (A, DEFAULT)
Executor CPU1-EXEC altered
42} DELETE EXECUTOR CPU0-EXEC
Executor CPU0-EXEC deleted
43} INFO EXECUTOR *, CLASS
EXECUTOR ATTRIBUTES for CPU1-EXEC
classes: A
DEFAULT
```

- This example shows the deletion of an executor in use by a job. The scheduler allows the job to finish before deleting the executor. Note the change in executor state from ACTIVE to DELETE.

```
7} STATUS EXECUTOR FAST-JOBS
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
------------------------ --- ------ ---- --------------------
FAST-JOBS 0 ACTIVE 105 INQUIRIES
8} DELETE EXECUTOR FAST-JOBS
Executor FAST-JOBS deleted
9} STATUS EXECUTOR FAST-JOBS
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
------------------------ --- ------ ---- --------------------
FAST-JOBS 0 DELETE 105 INQUIRIES
10} STATUS EXECUTOR FAST-JOBS
2087-E EXECUTOR FAST-JOBS does not exist
```

DELETE JOB Command

Use the DELETE JOB command to delete jobs that are not executing, over limit, or suspended.

```
D[ELETE] [ JOB ] [ / OU[T] [ file-name ] / ]
{ job-ID }
{ ( [ [NOT] job-ID ,]… [NOT] filter [ , [NOT] filter ]…)}
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

job-ID

specifies a job name or number, or a range of job names. To specify a range of job names, use the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all jobs.

NOT

excludes jobs that satisfy the specified job-selection criterion.

filter

specifies one of these job-selection criteria (see STATUS JOB Command on page 6-165):

```
A[TTACHMENT]-S[ET]
[ [ ( user-ID ) ] attachment-set-ID
#CURRENT ]
```

```
CL[ASS] class-ID
IN file-ID
STATE state
U[SER] user-ID
WAITO[N] master-job
```

Considerations

- The DELETE JOB command is available to all users, but these conditions apply:
  - NetBatch supervisors can delete jobs belonging to any user.
  - Users who are not NetBatch supervisors can delete any job whose input file is a disk file to which they have write access. If the input file does not exist or is a device or process, only the owner and NetBatch supervisors can delete the job.
  - If you are not a NetBatch supervisor, a DELETE JOB command that specifies *job-ID* with wild-card characters deletes only your own jobs.

    To delete another user's job if you have write access to its input file, specify the job's full name or number in *job-ID*.
- The DELETE JOB command deletes jobs whose state is EVENT, READY, RUNNEXT, RUNNOW, SPECIAL- n, TAPE, or TIME. The command does not delete jobs whose state is EXECUTING, OVER LIMIT, or SUSPENDED. (For information about stopping and deleting an executing, over-limit, or suspended job, see STOP JOB Command on page 6-180.)
- You can omit the object keyword JOB from the DELETE JOB command only when JOB is the current assumed object. For more information, see ASSUME JOB Command on page 6-72.

Examples

- This example shows the effect of a DELETE JOB * command entered by user 205,70:

```
1} CHANGEUSER 205,70 password
1} STATUS JOB *
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ---------- -------------
181 DEVELOPMENT 255,255 1398 4:Start err DVLPMNT
184 TOTALS 255,205 EVENT STANDARD-CLASS
186 TEST 205,100 1401 9:STALL STANDARD-CLASS
187 CLEANUP 205,100 1411 3:NEWP. err STANDARD-CLASS
188 CAL-JOB-1 205,70 7:CAL error STANDARD-CLASS
189 USERS 205,70 READY DEFAULT
190 BACKUP 8,255 TAPE STANDARD-CLASS
192 ACCOUNTS 8,1 1405 5:Fail RsOn STANDARD-CLASS
193 ADMIN 133,2 1407 6:Fail RsOffSTANDARD-CLASS
194 CAL-JOB-2 205,70 8:CAL exprd STANDARD-CLASS
195 STATISTICS 255,8 1409 2:NB failed STANDARD-CLASS
```

```
196 URGENT 255,8 RUN-NOW URGENT-JOBS
197 COMPILE 133,2 1412 SUSPENDED STANDARD-CLASS
198 Q3 205,70 1414 EXECUTING STANDARD-CLASS
2} DELETE JOB *
2132-E Your user code does not give you access to DEVELOPMENT
2132-E Your user code does not give you access to TOTALS
2132-E Your user code does not give you access to TEST
2132-E Your user code does not give you access to CLEANUP
Job CAL-JOB-1 Jobnumber 188 deleted
Job USERS Jobnumber 189 deleted
2132-E Your user code does not give you access to BACKUP
2132-E Your user code does not give you access to ACCOUNTS
2132-E Your user code does not give you access to ADMIN
Job CAL-JOB-2 Jobnumber 194 deleted
2132-E Your user code does not give you access to STATISTICS
2132-E Your user code does not give you access to URGENT
2132-E Your user code does not give you access to COMPILE
2077-E Job is executing or suspended; DELETE command ignored
```

- This example shows execution of a DELETE JOB command and the resulting events recorded in the scheduler log file:

```
> BATCHCOM $ZBAT; DELETE JOB *
Job P Jobnumber 39 deleted
Job Q Jobnumber 40 deleted
Job R Jobnumber 41 deleted
> FUP COPY $DATA7.ZBAT.LOGABL,,SHARE .
.
DELETE JOB P J_39 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
DELETE JOB Q J_40 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
DELETE JOB R J_41 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
```

DISPLAY-SPI Command

Use the DISPLAY-SPI command to display the contents of SPI-format command and response buffers that BATCHCOM sends to and receives from the scheduler. Displaying the buffers is useful when programming management applications use the NetBatch programmatic interface. For information on management-application programming, see the *SPI Programming Manual.* For information on the NetBatch programmatic interface, see the *NetBatch Management Programming Manual.*

```
D[ISPLAY]-SP[I] { OF[F] | ON }
```

`OFF`

disables the display of command and response buffers.

`ON`

enables the display of command and response buffers. The buffers appear in this format:

```
-------------------------------------------------------------
|_SPI_BUFFER_ { BEING_SENT_TO | RETURNED_FROM } _SCHEDULER_ |
-------------------------------------------------------------
Checksum: zspi-tkn-checksum
Header Type: zspi-tkn-hdrtype
Last error: zspi-tkn-lasterr
Last error code: zspi-tkn-lasterrcode
Last Position: zspi-tkn-lastposition
Max Field Version: zspi-tkn-max-field-version
Maxresp: zspi-tkn-maxresp
Position: zspi-tkn-position
Server version: zspi-tkn-server-version
Subsystem ID: zspi-tkn-ssid
Used length: zspi-tkn-usedlen
command-name ( command-name-value ) object-type ( object-
type-value )
[ TDT: token-data-type: { VAR | size }; token-code x count [
size: ] token-value ]…
```

zspi-tkn-checksum

is the value of the SPI header token ZSPI-TKN-CHECKSUM. The token contains the buffer checksum flag.

zspi-tkn-hdrtype

is the value of the SPI header token ZSPI-TKN-HDRTYPE. The token identifies the type of header the message contains.

zspi-tkn-lasterr

is the value of the SPI header token ZSPI-TKN-LASTERR. The token records the last nonzero status code returned by an SPI procedure while processing the buffer.

zspi-tkn-lasterrcode

is the value of the SPI header token ZSPI-TKN-LASTERRCODE. The token records the token code of the token involved in the last nonzero status code returned by an SPI procedure while processing the buffer.

zspi-tkn-lastposition

is the value of the SPI header token ZSPI-TKN-LASTPOSITION. The token contains the position of the last token SPI procedure SSPUT added to the buffer.

zspi-tkn-max-field-version

is the value of the SPI header token ZSPI-TKN-MAX-FIELD-VERSION. The token contains the highest version of any non-null field in any extensible structured token added to the buffer.

zspi-tkn-maxresp

is the value of the SPI header token ZSPI-TKN-MAXRESP. The token contains a value indicating how many response records BATCHCOM accepts in a response message.

zspi-tkn-position

is the value of the SPI header token ZSPI-TKN-POSITION. The token contains the current-token pointer.

zspi-tkn-server-version

is the value of the SPI header token ZSPI-TKN-SERVER-VERSION. The token contains the version of the scheduler, as set by the scheduler when it prepares its response to a command.

zspi-tkn-ssid

is the value of the SPI header token ZSPI-TKN-SSID. The token contains the subsystem ID of the scheduler that is to process the command.

zspi-tkn-usedlen

is the value of the SPI header token ZSPI-TKN-USEDLEN. The token contains the length of the used portion of the buffer.

command-name

is a command keyword. The keyword is the value of name in ZBAT-CMD-*name* or ZSPI-CMD- name (enumerated values of NetBatch private token ZBAT-DDL-RETCODE).

command-name-value

is a command number. The number is the numeric value of *command-name* as defined in the NetBatch DDL.

object-type

is an object keyword. The keyword is the value of name in ZBAT-OBJ-*name* (an enumerated value of NetBatch private token ZBAT-DDL-OBJECT).

object-type-value

is an object number. The number is the numeric value of *object-type* as defined in the NetBatch DDL.

token-data-type

is the data type of the token. The data type is the value of type in ZSPI-TDT-*type* (an SPI token).

VAR

indicates the token is a variable-length token. The size of the token's value in bytes is given in *size*: *token-value*.

size

is the token size in bytes (when size appears in token-data-type: *size*) or the size of the token's value in bytes (when size appears in *size*: *token-value*). *size* appears with *token-data-type* when the token is not a variable-length token. *size* appears with *token-value* when the token is a variable-length token.

token-code

is the token code. The token code is the value of code in ZSPI-TNM-*code* (an SPI token number) or ZBAT-TNM-*code* (a NetBatch token number).

count

is the number of times *token-code* occurs in the message.

token-value

is the token value. How BATCHCOM displays the value depends on the token data type:

- If the type is BYTE, CHAR, DEVICE, FNAME, STRUCT, or SUBVOL and the value is printable, that value appears. For example, the name TWENTY-FOUR-CHARACTERS-X in ZBAT-TKN-NETBATCH-NAME appears as:

  ```
  TDT: CHAR:24; NETBATCH-NAMEx1 TWENTY-FOUR-CHARACTERS-X
  ```

- If the value is printable and includes spaces, BATCHCOM appends the value with information in the format:

  ```
  ? [ count x ] binary-value
  ```

  count

  specifies the number of occurrences of binary-value. *count* appears only when *binary-value* occurs more than once.

  binary-value

  is the binary value of a space (32).

  For example, the 19-character name NINETEEN-CHARACTERS in the 24-character token ZBAT-TKN-NETBATCH-NAME appears as:

  ```
  TDT: CHAR:24; NETBATCH-NAMEx1 NINETEEN-CHARACTERS ?5x32
  ```

- If the value includes printable and unprintable characters, or is unprintable, the display format is:

  bytes: ? [*count x*] *binary-value*
  [? [*count x*] *binary-value*]… [*printable-chars*]…

  bytes

  is the number of bytes in the value.

  count

  specifies the number of occurrences of *binary-value*. *count* appears only when *binary-value* occurs more than once.

  binary-value

  is the binary value of the character (not necessarily a space).

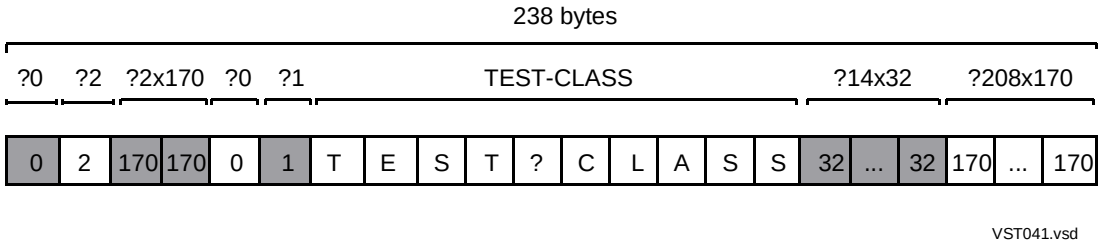
  printable-chars

  represents printable characters.

For example, the value of extensible structured token ZBAT-MAP-DEF-EXECUTOR in the command ADD EXECUTOR TEST-CLASS appears as:

```
TDT: STRUCT:VAR; DEF-EXECUTORx1
238: ?0 ?1 ?2x170 ?0 ?1 TEST-CLASS ?14x32 ?208x170
```

The interpretation of 238: ?0 ?2 ?2x170 ?0 ?1 TEST-CLASS ?14x32 ?208x170 from the previous example is:



- If the type is BOOLEAN, ENUM, INT, INT2, or UINT, the actual value appears. For example, a zero value for ZSPI-TKN-RETCODE appears as:

  ```
  TDT: ENUM:2; RETCODEx1 0000
  ```

- If the type is DATALIST, ENDLIST, or ERRLIST, then DATALIST, END LIST, or ERROR LIST appears as appropriate with details of the tokens in the list. For example, tokens in the data list returned by the scheduler in response to the command START EXECUTOR NINETEEN-CHARACTERS appear as:

  ```
  START DATA LIST
  TDT: CHAR:VAR; SEL-EXECUTORNAMEx1
  19:NINETEEN-CHARACTERS
  TDT: ENUM:2; RETCODEx1 0000
  END LIST
  ```

Considerations

- The DISPLAY-SPI command is available to all users.
- The RUN NETBATCH command parameter DISPLAY-SPI is an alternative to the DISPLAY-SPI command. The parameter makes the scheduler write to its log file command and response buffers received from and sent to requesters such as BATCHCOM. The buffers appear in the log file in the same format as in BATCHCOM. For more information, see Running NETBATCH on page 3-10.

Example

This example shows BATCHCOM displaying the command and response buffers for the command DELETE EXECUTOR EXEC-01:

```
17} DISPLAY-SPI ON
18} DELETE EXECUTOR EXEC-01
--------------------------------------
|_SPI_BUFFER_BEING_SENT_TO_SCHEDULER_|
--------------------------------------
Checksum: 0
Header Type: 0
Last error: 0
Last error code: tkn 29/4/-506
Last Position: SSCTL 8 -439
Max Field Version: 0
Maxresp: -1
Position: SSCTL 8 -442
Server version: 0 (0)00
Subsystem ID: TANDEM.9.D30
Used length: 69
DELETE (260)EXECUTOR (512)
TDT: CHAR:VAR; SEL-EXECUTORNAMEx1
7:EXEC-01
--------------------------------------
|_SPI_BUFFER_RETURNED_FROM_SCHEDULER_|
--------------------------------------
Checksum: 0
Header Type: 0
Last error: 0
Last error code: tkn 29/4/-506
Last Position: SSCTL 8 -439
Max Field Version: 0
Maxresp: -1
Position: SSCTL 8 -442
Server version: 17438 D30
Subsystem ID: TANDEM.9.D30
Used length: 86
DELETE (260)EXECUTOR (512)
START DATA LIST
TDT: CHAR:VAR; SEL-EXECUTORNAMEx1
7:EXEC-01
TDT: ENUM:2; RETCODEx1 0000
```

```
END LIST
Executor EXEC-01 deleted
```

EXIT Command

Use the EXIT command to end an interactive BATCHCOM session. The command stops the BATCHCOM process and returns you to the process from which you started the session.

```
E[XIT]
```

Considerations

- The EXIT command is available to all users.
- Creating an end-of-file (EOF) condition by pressing CTRL/Y has the same effect as the EXIT command.

Examples

This example shows the EXIT command ending a BATCHCOM session started from a TACL process:

```
> BATCHCOM .
.
1} EXIT
>
```

This example shows the effect of pressing CTRL/Y during a BATCHCOM session:

```
> BATCHCOM .
.
1} EOF!
>
E[XIT]
```

FC Command

Use the FC command to retrieve, edit, and reexecute command lines from BATCHCOM's history buffer. (FC is an abbreviation of fix command.)

```
F[C] [ num | - num | text ]
```

num

is a positive integer identifying a line in the history buffer.

- num

is a negative integer identifying a line in the history buffer relative to the current line.

text

is a character string identifying the latest line in the history buffer beginning with the string.

Considerations

- The FC command is available to all users.
- If no line in the history buffer matches your specification, this message appears:

  ```
  0541-I No such line
  ```

- Entering FC with no parameter retrieves the latest line from the history buffer (FC is the same as FC -1).
- The FC command displays the specified line with a period prompt (.) below it. The blank line at the prompt is an editing template in which you enter subcommands to edit the line. (The template and subcommands are the same as those from the TACL FC command.) The syntax of the template is:

```
subcommand [ // subcommand ]…
```

subcommand

is any of these:

```
{R replacement-text
I insertion-text
D replacement-text }
```

For information about using the editing template and its subcommands, see the description of the FC command in the *TACL Reference Manual*.

- When the FC command retrieves a multiline command, it initially displays only the first line of that command. Press RETURN to display the second line, then again for the third line, and so on. (BATCHCOM prefixes the second and later lines with double-braces prompts.) To escape from FC command mode without displaying all retrieved lines, press CTRL/Y.
- BATCHCOM stores all commands in its history buffer except CHANGEUSER, COMMENT, and HISTORY.

Examples

- These four examples show the FC command retrieving various lines from the history buffer. They also show editing of the lines and execution of the resulting commands. The examples relate to these commands in the buffer:

  ```
  3} INFO CLASS *
  4} ADD CLASS DEFAULT, INITIATION OFF
  5} STATUS EXECUTOR *
  6} ALTER EXECUTOR EXEC-01, CLASS (DEFAULT, ADMIN, SALES)
  ```

```
7} INFO JOB /OUT $S.#INFO/ *
8} SUBMIT JOB J3, IN INFILE3, AT 15:00
```

- This example shows retrieval, editing, and execution of the latest line in the buffer:

```
9} FC
9} SUBMIT JOB J3, IN INFILE3, AT 15:00
9}. 4// 4, WAITON J3
9} SUBMIT JOB J4, IN INFILE4, WAITON J3
9}.
Job J4 job number 4 submitted
10}
```

- This example shows retrieval, editing, and execution of command line 4 in the buffer:

```
9} FC 4
9} ADD CLASS DEFAULT, INITIATION OFF
9}.DDDIALTER// DDIN
9} ALTER CLASS DEFAULT, INITIATION ON
9}.
Class DEFAULT altered
10}
```

- This example shows retrieval, editing, and execution of the third command line preceding the current command line:

```
9} FC -3
9} ALTER EXECUTOR EXEC-01, CLASS (DEFAULT, ADMIN, SALES)
9}. ADMIN, SALES, DEFAULT
9} ALTER EXECUTOR EXEC-01, CLASS (ADMIN, SALES, DEFAULT)
9}.
Executor EXEC-01 altered
10}
```

- This example shows retrieval, editing, and execution of the latest command line beginning with INFO J:

```
9} FC INFO J
9} INFO JOB /OUT $S.#INFO/ *
9}. DDDICLASS// RADMIN
9} INFO CLASS /OUT $S.#INFO/ ADMIN
9}.
10}
```

- This example shows the retrieval, editing, and execution of a multiline command:

```
15} SUBMIT JOB STOCK-LEVELS, IN $A.NBFILES.STOCK, OUT $S,
MAXPRINTPAGES 100, CLASS STANDARD-CLASS, PRI 149, AFTER
17:30, HOLDAFTER ON
Job STOCK-LEVELS job number 5 submitted
16} FC 15
16} SUBMIT JOB STOCK-LEVELS, IN $A.NBFILES.STOCK, OUT $S,
MAXPRINTPAGES 100, &
16}.DDDDDDIALTER// B
16} ALTER JOB STOCK-LEVELS, IN $B.NBFILES.STOCK, OUT $S,
MAXPRINTPAGES 100, &
```

```
16}.
16}}CLASS STANDARD-CLASS, PRI 149, AFTER 17:30, HOLDAFTER ON
16}. DDDDDDDDDDDDDDIDEFAULT// FF
16}}CLASS DEFAULT, PRI 149, AFTER 17:30, HOLDAFTER OFF
16}.
Job STOCK-LEVELS job number 5 altered
```

HELP Command

Use the HELP command to display information about objects, commands, and attributes and about other general NetBatch topics such as keyword abbreviation.

```
H[ELP] [ / OU[T] [ file-name ] / ] [ topic | ALL ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

topic

is one of:

keyword

is a BATCHCOM keyword or its abbreviation or alias. For lists of keywords and keyword abbreviations and aliases, see Keywords on page 6-7.

special-topic

is one of:

ABBREVIATIONS	FILE-NAME	PERIOD
ASATTR	FREQUENCY	PHANDLE
ASSIGN-NAME	JBATTR	RUN OPT
ASSIGN-SPEC	JOB-FILTER-SELECTION	RUNATTR
ASTERISK	KEYWORDS	RWUP
AT-AFTER-TIME	LIST	SCATTR
ATT-NAME	MASTER	SCHEDULER-ID
ATT-NAME-SPEC	NAME	SECURITY
ATTRIBUTES	NB-NAME	SEMICOLON
CLATTR	NB-SECURITY	SHOW-A
COMMANDS	NBATTR	SHOW-E
CRONTAB	NETBATCH-ID	SHOW-J
DATE	NODENAME	SHOW-S

DATE-TIME	NUM	SPECIAL
DEFINE ?	NUMBER	STRING
DEFINE DEFINE-*class*	NUMCONC	SUPERVISOR
DEFINE-ATTRIBUTE	NUMNOW	TIME
DEFINE-CLASS	OBJECTS	USER-ID
DEFINE-NAME	ON-OFF	WAIT-FOR

ALL

causes BATCHCOM to display information about attachment sets, classes, executors, jobs, and schedulers, and to list the commands for manipulating them.

Considerations

- The HELP command is available to all users.
- Entering HELP without topic lists all topics for which HELP is available. Entering HELP HELP lists the same information.
- Some help text includes items enclosed in brackets (<>) for which further help is available. (The special-topic list includes these items.) Do not enter these brackets when you specify the items in help commands.

Examples

- This example shows a sequence of HELP commands that display information about ASSIGN attributes:

```
274} HELP ATTRIBUTES
Object Attribute
------ ---------
ATTACHMENT-SET <asattr>
JOB <jbattr>
SCHEDULER <scattr>
EXECUTOR <exattr>
CLASS <clattr>
275} HELP ASATTR
<asattr> SEC[URITY] <rwup>
TEM[PORARY] { ON | OFF }
( ASSI[GN] <assign-name> {, <assign-spec> } … )
( DEFI[NE] <define-name> {, <define-spec> } … )
( PA[RAM] { <param-name> <param-value> }, … )
<asattr> represents an attribute of an ATTACHMENT-SET.
276} HELP ASSIGN-SPEC
<assign-spec> is [ <filename> ] , [ <assign-attributes> ] , …
<assign-attributes> are:
EXT <primary-extent-size>
EXT ( [<primary-extent-size>] [, <secondary-extent-size> ] )
C[ODE] <filecode>
<exclusion-spec> | EXC[LUSIVE] | <access-spec> | I-O |
| S[HARED] | | IN[PUT] |
```

```
| P[ROTECTED] | | O[UTPUT] |
R[EC] <recordsize>
B[LOCK] <blocksize>All numeric values must be in the range 0-
65535.
DEFINE-SPEC PARAM-NAME WAIT-TIME
DEFINE-VALUE PARAM-VALUE *
EXATTR PASSWORD-STRING ;
```

- This example shows HELP DEFINE-class commands listing all possible attributes of catalog and defaults DEFINEs:

```
103} HELP DEFINE CATALOG
ATTRIBUTES ATTRIBUTE-VALUE
---------- ---------------
CLASS CATALOG
SUBVOL Required
104} HELP DEFINE DEFAULTS
ATTRIBUTES ATTRIBUTE-VALUE
---------- ---------------
CLASS DEFAULTS
VOLUME Required
SWAP Optional
CATALOG Optional
```

- This example shows the result of a HELP DEFINE ? command executed immediately after a SET ATTACHMENT-SET (DEFINE …) command. The display lists all possible tape DEFINE attributes and the values set by the SET command.

```
279} SET ATTACHMENT-SET (DEFINE =TAPE, CLASS TAPE, DEVICE
$TAPE, LABELS ANSI, VOLUME SCRATCH, OWNER NB.USER, EXPIRATION
OCT301994, MOUNTMSG "Load the tape!")
280} HELP DEFINE ?
ATTRIBUTES ATTRIBUTE-VALUE
---------- ---------------
CLASS TAPE
VOLUME SCRATCH Optional
LABELS ANSI Defaulted
REELS Optional
OWNER NB.USER Optional
FILESECT Optional
FILESEQ Optional
FILEID Optional
RETENTION Optional
EXPIRATION 30 OCT 1994 Optional
GEN Optional
VERSION Optional
RECFORM Optional
BLOCKLEN Optional
RECLEN Optional
DENSITY Optional
USE Optional
DEVICE \MELBDEV.$TAPE Optional
EBCDIC Optional
MOUNTMSG Load the tape! Optional
SYSTEM Optional
TAPEMODE Optional
```

HISTORY Command

Use the HISTORY command to display a specified number of the most recently executed command lines in BATCHCOM's history buffer.

```
HIS[TORY] [ / OU[T] [ file-name ] / ] [ num ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

num

is an integer greater than zero specifying the number of command lines to display. The default is 10 lines.

Considerations

- The HISTORY command is available to all users.
- The size of the history buffer is 2048 bytes. The number of command lines the buffer can store depends on how many characters are in the commands. As a guide, the buffer could store approximately eighty 25-character commands or forty 50-character commands.
- BATCHCOM stores all commands in its history buffer except CHANGEUSER, COMMENT, and the HISTORY command itself.
- To manipulate command lines in the history buffer, use the FC, exclamation point (!), and question mark (?) commands:
  - Use the FC command to retrieve, edit, and reexecute a command line from the buffer. For more information, see FC Command on page 6-92.
  - Use the ! command to reexecute immediately a command line from the buffer. For more information, see ! Command on page 6-198.
  - Use the ? command to display a command line from the buffer. For more information, see ? Command on page 6-200.

Each command line you reexecute with the FC, !, or ? command becomes a new line in the history buffer.

Examples

- This HISTORY command in this example displays the two latest command lines:

```
13} HISTORY 2
11} SUBMIT JOB TODAY, IN $A.DATA.TODAY, OUT $S, WAIT 4:00
```

```
12} ALTER JOB TODAY, PRI 165, SELPRI 7
13}
```

- This example shows the HISTORY command displaying all commands except CHANGEUSER:

```
1} SUBMIT JOB X, AFTER 23:00
Job X job number 7 submitted
2} CHANGEUSER 255,205 password
2} ALTER SCHEDULER, AT-ALLOWED OFF
Scheduler altered
3} CHANGEUSER 205,70 password
3} SUBMIT JOB Y, AFTER 23:15
Job Y job number 8 submitted
4} HISTORY
1} SUBMIT JOB X, AFTER 23:00
2} ALTER SCHEDULER, AT-ALLOWED OFF
3} SUBMIT JOB Y, AFTER 23:15
4}
```

INFO ATTACHMENT-SET Command

Use the INFO ATTACHMENT-SET command to list the attributes of attachment sets.

```
I[NFO] [ ATTACHMENT-SET ] [ / OU[T] [ file-name ] / ]
{ [ ( user-ID ) ] attachment-set-ID | #CURRENT }
[ [ , attribute ]… | , DET[AIL] ] [ , O[BEY]-[FORM] ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

user-ID

specifies a user ID or a range of user IDs specified with the asterisk (*) and question mark (?) wild-card characters. (*user-ID* must be in *group-name.user-name* or *group-ID*,*user-ID* form.) The default is the user ID of the current user.

attachment-set-ID

is one of:

attachment-set-name

specifies an attachment-set name or a range of names when specified with the asterisk (*) and question mark (?) wild-card characters.

attachment-set-number

is a scheduler-generated number identifying an attachment set created by means of the #CURRENT variable.

*

specifies all attachment sets.

#CURRENT

causes BATCHCOM to list attributes of the attachment set specified by the #CURRENT variable. If #CURRENT has a null value, BATCHCOM displays this message:

```
0345-E Undefined substitution
```

attribute

is one of these attachment-set attributes:

```
ASSI[GN] [ ASSIGN-name ]
DEFI[NE] [ DEFINE-name ]
PA[RAM] [ PARAM-name ]
SEC[URITY]
TEM[PORARY]
```

attribute causes BATCHCOM to list only the specified attribute. If you omit attribute and do not specify DETAIL, BATCHCOM lists the SECURITY and TEMPORARY attributes.

You can use *ASSIGN-name*, *DEFINE-name*, or *PARAM-name* to specify an ASSIGN, DEFINE, or PARAM name or a range of names. To specify a range of names, use the asterisk (*) and question mark (?) wild-card characters.

DETAIL

causes BATCHCOM to list all attachment-set attributes. If you omit DETAIL and do not specify attribute, BATCHCOM lists the SECURITY and TEMPORARY attributes.

OBEY-FORM

causes BATCHCOM to list attributes in SET ATTACHMENT-SET command format. BATCHCOM prefixes the SET commands with ASSUME ATTACHMENT-SET and RESET ATTACHMENT-SET commands, and suffixes them with an ADD ATTACHMENT-SET command.

OBEY-FORM lists the ADD ATTACHMENT-SET command in one of two forms: uncommented or commented. The command appears in uncommented form if it specifies a named attachment set; for example:

```
ADD (SUPER.FPP)DAILY-PARAMS
```

The command appears in commented form if it specifies a numbered attachment set. For example:

```
==ADD (FPP.MANAGER)8
```

Considerations

- The INFO ATTACHMENT-SET command is available to all users. However, you must have read access to an attachment set for BATCHCOM to display more than the set's SECURITY attribute. For example, NB.USER could list all attributes of set (SUPER.FPP)A secured "AGAG," but only the SECURITY attribute of set (SUPER.FPP)B secured "GOGO."
- The OBEY-FORM qualifier is available for all INFO commands. It enables the creation of BATCHCOM command files that you can use to set up a scheduler. For more information, see Section 3, Scheduler Planning, Configuration, and Management.
- ADP (an abbreviation of ASSIGNs, DEFINEs, and PARAMs) is an alias of the object keyword ATTACHMENT-SET.
- You can omit the object keyword ATTACHMENT-SET from the INFO ATTACHMENT-SET command only when ATTACHMENT-SET is the current assumed object. For more information, see ASSUME ATTACHMENT-SET Command on page 6-69.

Examples

- This example shows the effect of the user-ID, attachment-set-ID, and attribute options in an INFO ATTACHMENT-SET command. The (*.*)* user-ID and attachment-set-ID options cause BATCHCOM to list all attachment sets to which the current user (NB.USER) has read access. The SECURITY, PARAM SHIFT, and DEFINE =OUT attribute options further limit the listing.

```
15} CHANGEUSER NB.USER psswrd

15} INFO ATTACHMENT-SET (*.*)*, SECURITY, PARAM SHIFT,
DEFINE =OUT
ATTACHMENT-SET ATTRIBUTES for (NB.USER)EARLY
security: "NNNN"
attachments: DEFINE =OUT, CLASS SPOOL, LOC \A.$S.#EARLY
PARAM SHIFT 0700-1500
ATTACHMENT-SET ATTRIBUTES for (NB.MANAGER)LATE
security: "AOAO"
attachments: DEFINE =OUT, CLASS SPOOL, LOC \A.$S.#LATE
PARAM SHIFT 1500-2300
ATTACHMENT-SET ATTRIBUTES for (SUPER.NB)NIGHT
security: "NCNO"
attachments: DEFINE =OUT, CLASS SPOOL, LOC \A.$S.#NIGHT
PARAM SHIFT 2300-0700
ATTACHMENT-SET ATTRIBUTES for (SUPER.SUPER)MANAGERS-ONLY
security: "OOOO"
0535-I Your user code does not give you access to
```

```
(SUPER.SUPER)MANAGERS-ONLY data
ATTACHMENT-SET ATTRIBUTES for (SUPER.SUPER)SWING
security: "AOOO"
attachments: DEFINE =OUT, CLASS SPOOL, LOC \A.$S.#SWING
PARAM SHIFT 0900-1700
```

- This example shows an INFO ATTACHMENT-SET command writing specified attachment-set attributes to an EDIT file:

```
6} ADD ATTACHMENT-SET (FPP.MANAGER)MNGR-ADPS, (ASSIGN
BAPREWK,
\MELBORN.$FPP.MSMNTHLY.BAPREWK), (PARAM REP-DATE .06-OCT-
94.),
(PARAM MERGE-DATE .941006.), SECURITY "NONO", TEMPORARY OFF
Attachment-set (FPP.MANAGER)MNGR-ADPS added
7} INFO ATTACHMENT-SET /OUT X/ (FPP.MANAGER)MNGR-ADPS,
SECURITY, TEMPORARY, PARAM *DATE*
8} RUN FUP COPY X
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)MNGR-ADPS
security: "NONO"
temporary: Off
attachments: PARAM REP-DATE .06-OCT-94.
PARAM MERGE-DATE .941006.
```

- This example shows the result of an INFO ATTACHMENT-SET command that includes the OBEY-FORM qualifier. In the example, BATCHCOM sends the command's output to an EDIT file.

```
9} ADD ATTACHMENT-SET (NB.USER)GROUP-NB, SECURITY "GOGO",
TEMPORARY OFF, (DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB), (ASSIGN A, $DATA6.NB.A), (ASSIGN B, $DATA6.NB.B),
(PARAM DAY MONDAY), (PARAM MONTH OCTOBER), (PARAM YEAR 1994)
Attachment-set (NB.USER)GROUP-NB added
10} INFO ATTACHMENT-SET /OUT ADPFILE/ (NB.USER)GROUP-NB,
DETAIL, OBEY-FORM
11} RUN FUP COPY ADPFILE
ASSUME ATTACHMENT-SET
RESET
SET SECURITY "GOGO"
SET TEMPORARY OFF
SET (DEFINE =_DEFAULTS,CLASS DEFAULTS,VOLUME $A.NB)
SET (ASSIGN A,$DATA6.NB.A)
SET (ASSIGN B,$DATA6.NB.B)
SET (PARAM DAY MONDAY)
SET (PARAM MONTH OCTOBER)
SET (PARAM YEAR 1994)
ADD (NB.USER)GROUP-NB
```

INFO CLASS Command

Use the INFO CLASS command to list the attributes of classes.

```
I[NFO] [ CLASS ] [ / OU[T] [ file-name ] / ] class-ID
[ , O[BEY]-[FORM] ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

class-ID

is a class name, or a range of class names when specified with the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all classes.

OBEY-FORM

causes BATCHCOM to list attributes in ADD CLASS command format.

Considerations

- The INFO CLASS command is available to all users.
- The INITIATION attribute is the only attribute a class can have. Possible values of this attribute are:

Attribute Value	Effect
OFF	Denies jobs in the class access to the class's executors
ON	Gives jobs in the class access to the class's executors

- The OBEY-FORM qualifier is available for all INFO commands. It enables the creation of BATCHCOM command files that you can use to set up a scheduler. For more information, see Section 3, Scheduler Planning, Configuration, and Management.
- JOBCLASS is an alias of the object keyword CLASS.
- You can omit the object keyword CLASS from the INFO CLASS command only when CLASS is the current assumed object. For more information, see ASSUME CLASS Command on page 6-70.

Examples

- This example shows the result of an INFO CLASS command:

```
15} INFO CLASS *
CLASS ATTRIBUTE for DEFAULT
```

```
initiation: On
CLASS ATTRIBUTE for MIS
initiation: Off
CLASS ATTRIBUTE for SALES
initiation: Off
```

- This example shows the result of an INFO CLASS command that writes class details to an EDIT file:

```
24} INFO CLASS /OUT OPSINFO/ *
25} RUN FUP COPY OPSINFO
CLASS ATTRIBUTE for DEFAULT
initiation: On
CLASS ATTRIBUTE for MIS
initiation: Off
CLASS ATTRIBUTE for SALES
initiation: Off
```

- This example shows the result of an INFO CLASS command that includes the OBEY-FORM qualifier. In the example, BATCHCOM sends the command's output to an EDIT file.

```
16} INFO CLASS /OUT ADDCLASS/ *, OBEY-FORM
17} RUN FUP COPY ADDCLASS
ADD CLASS DEFAULT, INITIATION ON
ADD CLASS MIS, INITIATION OFF
ADD CLASS SALES, INITIATION OFF
```

INFO EXECUTOR Command

Use the INFO EXECUTOR command to list the attributes of executors.

```
I[NFO] [ EXECUTOR ] [ / OU[T] [ file-name ] / ]
executor-ID [ [ , attribute ]… | , O[BEY]-[FORM] ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

executor-ID

is an executor name, or a range of executor names when specified with the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all executors.

attribute

is one of these executor attributes:

```
CL[ASS]
CP[U]
```

attribute causes BATCHCOM to list only the specified attribute. If you omit attribute, BATCHCOM lists all attributes.

OBEY-FORM

causes BATCHCOM to list attributes in ADD EXECUTOR command format.

Considerations

- The INFO EXECUTOR command is available to all users.
- The OBEY-FORM qualifier is available for all INFO commands. It enables the creation of BATCHCOM command files that you can use to set up a scheduler. For more information, see Section 3, Scheduler Planning, Configuration, and Management.
- You can omit the object keyword EXECUTOR from the INFO EXECUTOR command only when EXECUTOR is the current assumed object. For more information, see ASSUME EXECUTOR Command on page 6-71.

Examples

- This example shows the result of an INFO EXECUTOR command executed during an interactive BATCHCOM session:

```
30} INFO EXECUTOR *
EXECUTOR ATTRIBUTES for RISC-0
cpu: 0
classes: INQUIRIES
SALES
SERVICE
EXECUTOR ATTRIBUTES for RISC-1
cpu: 1
classes: SALES
SERVICE
INQUIRIES
EXECUTOR ATTRIBUTES for RISC-2
cpu: 2
classes: SERVICE
INQUIRIES
SALES
EXECUTOR ATTRIBUTES for RISC-3
cpu: 3
classes: ACCOUNTS
ADMINISTRATION
```

- This example shows an INFO EXECUTOR command writing the CPU attributes of a scheduler's executors to an EDIT file:

```
31} INFO EXECUTOR /OUT CPUINFO/ *, CPU
32} RUN FUP COPY CPUINFO
EXECUTOR ATTRIBUTES for RISC-0
cpu: 0
EXECUTOR ATTRIBUTES for RISC-1
```

```
cpu: 1
EXECUTOR ATTRIBUTES for RISC-2
cpu: 2
EXECUTOR ATTRIBUTES for RISC-3
cpu: 3
```

- This example shows the result of an INFO EXECUTOR command that includes the OBEY-FORM qualifier. In the example, BATCHCOM sends the command's output to an EDIT file.

```
33} INFO EXECUTOR /OUT NEWEXECS/ *, OBEY-FORM
34} RUN FUP COPY NEWEXECS
ADD EXECUTOR RISC-0, CPU 0, CLASS(INQUIRIES, SALES, SERVICE)
ADD EXECUTOR RISC-1, CPU 1, CLASS(SALES, SERVICE, INQUIRIES)
ADD EXECUTOR RISC-2, CPU 2, CLASS(SERVICE, INQUIRIES, SALES)
ADD EXECUTOR RISC-3, CPU 3, CLASS(ACCOUNTS, ADMINISTRATION)
```

INFO JOB Command

Use the INFO JOB command to list the attributes of jobs.

```
I[NFO] [ JOB ] [ / OU[T] [ file-name ] / ]
{ job-ID }
{ ( [ [NOT] job-ID ,]… [NOT] filter [ , [NOT] filter ]…)}
[ , attribute ]… [ , OW[NER] ] [ , O[BEY]-[FORM] ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

job-ID

specifies a job name or number, or a range of job names. To specify a range of job names, use the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all jobs.

NOT

excludes jobs that satisfy the specified job-selection criterion.

filter

specifies one of these job-selection criteria (see STATUS JOB Command on page 6-165):

```
A[TTACHMENT]-S[ET
[ ( user-ID ) ] attachment-set-ID
#CURRENT ]
CL[ASS] class-ID
IN file-ID
```

```
STATE state
U[SER] user-ID
WAITO[N] master-job
```

attribute

is one of these job attributes:

```
AF[TER]        J[OB]-L[OG]            RUND
AT             J[OBID]-Z[ERO]         SA[VEABEND]
CA[LENDAR]     LIB                    SEL[PRI]
CL[ASS]        LIM[IT]                STAL[L]
DES[CRIPTION]  MAXPRINTL[INES]        STARTU[P]
EV[ERY]        MAXPRINTP[AGES]        S[TOP]-[ON]-[ABEND]
E[XECUTOR]     ME[M]                  SWA[P]
-[PROGRAM
EXT[SWAP]      NA[ME]                 TA[PEDRIVES]
HIG[HPIN]      OU[T]                  V[OLUME]
HOLD           PF[S]                  WAIT
HOLDA[FTER]    PR[I]                  WAITO[N]
F[AILS]        P[URGE]-[IN]-[FILE]
IN              REST[ART]
```

attribute causes BATCHCOM to list only the specified attribute. If you omit *attribute* and do not specify OWNER, BATCHCOM lists all attributes along with the date and time the job was submitted, the date and time the job was last altered and the ID of the user who altered it, and the owner's user ID.

OWNER

causes BATCHCOM to list the user ID of the specified job's owner as well as the attributes specified by *attribute*. If you specify OWNER but omit *attribute*, BATCHCOM lists the user ID only. If you omit both OWNER and *attribute*, BATCHCOM lists all attributes along with the date and time the job was submitted, the date and time the job was last altered and the ID of the user who altered it, and the owner's user ID.

OBEY-FORM

causes BATCHCOM to list attributes in SET JOB command format. BATCHCOM prefixes the SET commands with ASSUME JOB and RESET JOB commands, and suffixes them with SUBMIT JOB and commented CHANGEUSER commands. The CHANGEUSER command (for example, ==CHANGEUSER 133,2) specifies the user ID of the job's owner, but not the password. For the command to work, you must edit it to remove the double equal signs and add the owner's password.

OBEY-FORM lists the SET JOB ATTACHMENT-SET command in one of two forms: uncommented or commented. The command appears in uncommented form if it specifies a named attachment set. For example:

```
SET JOB ATTACHMENT-SET ((SUPER.FPP)ADMIN)
```

The command appears in commented form if it specifies a numbered attachment set. For example:

```
==SET JOB ATTACHMENT-SET ((FPP.MANAGER)4)
```

Considerations

- The INFO JOB command is available to all users.
- The OBEY-FORM qualifier is available for all INFO commands. It enables the creation of BATCHCOM command files that you can use to set up a scheduler. For more information, see Section 3, Scheduler Planning, Configuration, and Management.
- You can omit the object keyword JOB from the INFO JOB command only when JOB is the current assumed object. For more information, see ASSUME JOB Command on page 6-72.

Examples

- The INFO JOB command in this example lists all attributes of a specified job:

```
50} SUBMIT JOB, EXECUTOR-PROGRAM DELAY, STARTUP "1 MINS",
EVERY
0:05 HOURS, DESCRIPTION "This is a test job. It starts a
DELAY process that runs for one minute every five minutes.
The job has a scheduler-assigned name."
Job ZBAT-0029 Jobnumber 29 submitted
51} INFO JOB 29
JOB ATTRIBUTES for ZBAT-0029
jobnumber: 29
volume: \DEV.$DATA7.NB, "NCNU"
out: \DEV.$S.#BATCH
executor-program: \DEV.$SYSTEM.SYSTEM.DELAY
startup: 1 MINS
jobid-zero: Off
pri: 120
selpri: 3
maxprintlines: None
maxprintpages: None
class: DEFAULT
stall: Off
stop-on-abend: Off
every: 0:05 HOURS
highpin: Off
submit: 07OCT94 10:07:34
alter: 07OCT94 10:07:34
user: 255,255
next-runtime: 07OCT94 10:07:34
description:
This is a test job. It starts a DELAY process that runs for
one minute every five minutes. The job has a scheduler
assigned name.
```

- This example shows the result of an INFO JOB command that includes the OBEY-FORM qualifier. In the example, BATCHCOM sends the command's output to an EDIT file.

```
59} INFO JOB /OUT JOB29/ 29, EXECUTOR-PROGRAM, STARTUP,
EVERY,
OBEY-FORM
60} RUN FUP COPY JOB29
ASSUME JOB
RESET
SET EXECUTOR-PROGRAM \MELBDEV.$SYSTEM.SYSTEM.DELAY
SET STARTUP "1 MINS"
SET EVERY 0:05 HOURS
==CHANGEUSER 255,255
SUBMIT ZBAT-0029
```

- This example shows an INFO JOB command appending commands that describe a job to the file containing commands that describe the job's attachment set:

```
32> ASSIGN SAVINGS-FILE, $DATA7.BANK.SAVINGS
33> ASSIGN LOANS-FILE, $DATA7.BANK.LOANS
34> PARAM PERIOD Q3
35> ALTER DEFINE =_DEFAULTS, VOLUME $A.NB
36> BATCHCOM $SCHD; SUBMIT JOB MARGIN, IN PROFIT, ATTACHMENT-
SET #CURRENT, HOLD ON
Attachment-set (SUPER.FPP)5 added
Job MARGIN job number 44 submitted
37> BATCHCOM $SCHD; INFO ATTACHMENT-SET /OUT MYJOB/
(SUPER.FPP)5, OBEY-FORM
38> BATCHCOM $SCHD; INFO JOB /OUT MYJOB/ 44, OBEY-FORM
39> FUP COPY MYJOB
ASSUME ATTACHMENT-SET
RESET
SET SECURITY "UUUU"
SET TEMPORARY On
SET (DEFINE =_DEFAULTS,CLASS DEFAULTS,VOLUME $A.NB)
SET (ASSIGN LOANS-FILE,$DATA7.BANK.LOANS)
SET (ASSIGN SAVINGS-FILE,$DATA7.BANK.SAVINGS)
SET (PARAM PERIOD Q3)
==ADD (SUPER.FPP)5
ASSUME JOB
RESET
SET VOLUME $A.NB, "NNNC"
SET IN $A.NB.PROFIT
SET OUT $S.#BATCH .
.
==SET JOB ATTACHMENT-SET ((SUPER.FPP)5)
==CHANGEUSER 255,205
SUBMIT MARGIN
```

INFO SCHEDULER Command

Use the INFO SCHEDULER command to list the attributes of schedulers.

```
I[NFO] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
[ , attribute ]… [ , O[BEY]-[FORM] ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

attribute

is one of these scheduler attributes:

AT-A[LLOWED]	D[EFAULT]-MAXPRINTP[AGES]	INI[TIATION]
B[ACKUPCPU]	D[EFAULT]-O[UT]	LO[CALNAMES]
CAT[CHUP]	D[EFAULT]-P[RI]	M[AX]-[CONCURRENT]-[JOBS]
D[EFAULT]-C[LASS]	D[EFAULT]-SE[LPRI]	M[AX]-[PRI]
D[EFAULT]-E[XECUTOR]-[PROGRAM]	D[EFAULT]-ST[ALL]	S[UBMIT]-A[LLOWED]
D[EFAULT]-H[IGHPIN]	D[EFAULT]-[STOP]-[ON]-[ABEND]	TA[PEDRIVES]
D[EFAULT]-MAXPRINTL[INES]	EM[S]	

attribute causes BATCHCOM to list only the specified attribute. If you omit attribute, BATCHCOM lists all attributes.

OBEY-FORM

causes BATCHCOM to list attributes in ALTER SCHEDULER command format. BATCHCOM prefixes the ALTER commands with the ASSUME SCHEDULER and RESET SCHEDULER commands.

Considerations

- The INFO SCHEDULER command is available to all users.
- The OBEY-FORM qualifier is available for all INFO commands. It enables the creation of BATCHCOM command files that you can use to set up a scheduler. For more information, see Section 3, Scheduler Planning, Configuration, and Management.

- You can omit the object keyword SCHEDULER from the INFO SCHEDULER command only when SCHEDULER is the current assumed object. For more information, see ASSUME SCHEDULER Command on page 6-73.

Examples

- This example shows the result of an INFO SCHEDULER command:

```
65} INFO SCHEDULER
SCHEDULER ATTRIBUTES
backupcpu: 0
at-allowed: On
submit-allowed: On
initiation: On
default-executor-program: \MELBDEV.$SYSTEM.SYSTEM.TACL
max-pri: 199
default-pri: 120
max-concurrent-jobs: 1,1
default-selpri: 3
default-maxprintlines: None
default-maxprintpages: None
tapedrives: 2
default-out: \MELBDEV.$S.#BATCH
default-class: DEFAULT
default-stall: Off
default-stop-on-abend: Off
default-highpin: Off
catchup: On
ems: Off
```

- This example shows an INFO SCHEDULER command writing specified attributes of a scheduler to an EDIT file:

```
66} INFO SCHEDULER /OUT ZBATINFO/, DEFAULT-EXECUTOR-PROGRAM,
DEFAULT-PRI, DEFAULT-OUT, DEFAULT-CLASS
67} RUN FUP COPY ZBATINFO
SCHEDULER ATTRIBUTES
default-executor-program: \MELBDEV.$SYSTEM.SYSTEM.TACL
default-pri: 120
default-out: \MELBDEV.$S.#BATCH
default-class: DEFAULT
```

- This example shows the result of an INFO SCHEDULER command that includes the OBEY-FORM qualifier. In the example, BATCHCOM sends the command's output to an EDIT file. Note the ASSUME SCHEDULER and RESET SCHEDULER commands at the start of the file.

```
68} INFO SCHEDULER /OUT ZBATATTR/, OBEY-FORM
69} RUN FUP COPY ZBATATTR
ASSUME SCHEDULER
RESET
ALTER, BACKUPCPU 0
ALTER, AT-ALLOWED ON
ALTER, SUBMIT-ALLOWED ON
ALTER, INITIATION ON
```

```
ALTER, DEFAULT-EXECUTOR-PROGRAM \MELBDEV.$SYSTEM.SYSTEM.TACL
ALTER, MAX-PRI 199
ALTER, DEFAULT-PRI 120
ALTER, MAX-CONCURRENT-JOBS 1,1
ALTER, DEFAULT-SELPRI 3
ALTER, DEFAULT-MAXPRINTLINES NONE
ALTER, DEFAULT-MAXPRINTPAGES NONE
ALTER, TAPEDRIVES 2
ALTER, DEFAULT-OUT \MELBDEV.$S.#BATCH
ALTER, DEFAULT-CLASS DEFAULT
ALTER, DEFAULT-STALL OFF
ALTER, DEFAULT-STOP-ON-ABEND OFF
ALTER, DEFAULT-HIGHPIN OFF
ALTER, CATCHUP ON
ALTER, EMS OFF
```

OBEY Command

Use the OBEY command to execute BATCHCOM commands contained in a disk file.

```
O[BEY] file-name
```

file-name

is the name of a disk file containing BATCHCOM commands. The file can include nested OBEY commands specified directly in the file itself or indirectly in files opened by those commands.

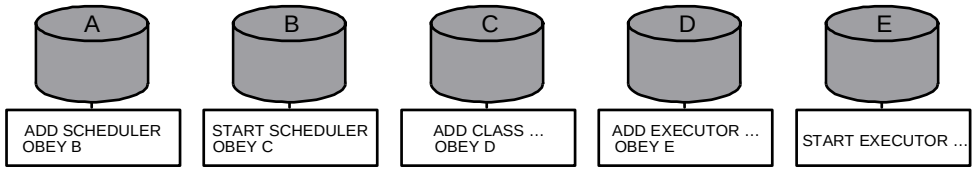
Note. The number of OBEY commands you can nest depends on the number of files those commands cause BATCHCOM to open concurrently. BATCHCOM can handle up to five open input files at a time.

Considerations

- The OBEY command is available to all users.
- To stop BATCHCOM executing the commands contained in *file-name*, press the BREAK key on the terminal used to enter the OBEY command.

Example

This example shows execution of four OBEY commands, three of which occur as nested commands in files A through C (see figure). File D's OBEY command fails because BATCHCOM cannot open any more files. (When execution fails, the open files are the terminal—opened by the RUN BATCHCOM command—and files A through D.)




```
> BATCHCOM $ZBAT; OBEY A
ADD SCHEDULER
Scheduler added
OBEY B
START SCHEDULER
Scheduler started
OBEY C
ADD CLASS DEFAULT, INITIATION OFF
Class DEFAULT added
OBEY D
ADD EXECUTOR EX0, CPU 0, CLASS DEFAULT
Executor EX0 added
OBEY E
0342-E Too many OBEY files
```

OPEN Command

Use the OPEN command during an interactive BATCHCOM session to specify the target scheduler for later commands.

```
OP[EN] [ \ node. ] $schd
```

node

is a node name. If the node is remote, you must have remote access to it (that is, you must have remote passwords set up on both local and remote nodes). The default is the node where the process that creates the BATCHCOM process is running.

schd

is a scheduler process name.

Considerations

- The OPEN command is available to all users.
- An implicit OPEN command applies in either of these circumstances:

- When you specify a scheduler in BATCHCOM's TACL RUN command. For example, the first of these commands opens scheduler $SCHD. The second starts BATCHCOM but does not open a scheduler:

```
> BATCHCOM $SCHD
BATCHCOM - T9190D30 …
NETBATCH SERVER - T9190D30 … Time: 06OCT94 11:47:24
1}
10> BATCHCOM
BATCHCOM - T9190D30 …
1}
```

- When BATCHCOM executes the first scheduler-related command after a session begins and BATCHCOM's TACL RUN command did not specify a scheduler. In this case, BATCHCOM opens scheduler $ZBAT if it exists. For example:

```
> BATCHCOM
BATCHCOM - T9190D30 …
1} STATUS SCHEDULER
NETBATCH SERVER - T9190D30 … Time: 06OCT94 11:49:08
SCHEDULER STATUS
Process : \DEV.$ZBAT Primary : 0,39 Backup : 1,55
Database: \DEV.$DATA7.ZBAT
Logfile : \DEV.$DATA7.ZBAT.LOGABJ
Time : 06OCT94 11:49:08 .
.
```

- If $ZBAT does not exist, the first scheduler-related command fails, and BATCHCOM displays a message similar to:

```
> BATCHCOM
BATCHCOM - T9190D30 …
1} STATUS SCHEDULER
-^-0014-
E Device does not exist, \MELBDEV.$ZBAT
```

- The OPEN command, like the RESET ATTACHMENT-SET and SET ATTACHMENT-SET commands, sets the #CURRENT variable's value to null.

Example

This example shows an OPEN command changing the target scheduler from the D-series scheduler \MELBDEV.$ZBAT to the C-series scheduler \MELBQAT.$SCHD:

```
> BATCHCOM /HIGHPIN OFF/
BATCHCOM - T9190D30 …
1} STATUS SCHEDULER
NETBATCH SERVER - T9190D30 … Time: 06OCT94 12:03:34
SCHEDULER STATUS
Process : \MELBDEV.$ZBAT Primary : 0,267 Backup : 2,270
Database: \MELBDEV.$DATA7.ZBAT
Logfile : \MELBDEV.$DATA7.ZBAT.LOGABK
Time : 06OCT94 12:03:35 .
.
```

```
2} OPEN \MELBQAT.$SCHD
NETBATCH SERVER - T9190C23 … Time: 07OCT94 20:43:31
3} STATUS SCHEDULER
SCHEDULER STATUS
Process : \MELBQAT.$SCHD Primary : 1,43 Backup : 0,72
Database: $DATA7.SCHD
Logfile : $DATA7.SCHD.LOGAAB
Time : 07OCT94 20:43:31 .
.
```

RELEASE-WAITON Command

Use the RELEASE-WAITON command to release dependent jobs from their masters, without running the masters.

```
R[ELEASE]-W[AITON] dependent-job-ID FR[OM] master-job-ID
```

dependent-job-ID

specifies a dependent job's name or, when master-job-ID does not contain wild-card characters, a range of dependent jobs' names. To specify a range of job names, use the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all dependent jobs.

master-job-ID

specifies a master job's name or, when dependent-job-ID does not contain wild-card characters, a range of master jobs' names. To specify a range of job names, use the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all master jobs.

Considerations

- The RELEASE-WAITON command is available to all users, but:
  - NetBatch supervisors can release dependent jobs belonging to any user.
  - Users who are not NetBatch supervisors can release any dependent job whose input file is a disk file to which they have write access. If the input file does not exist or is a device or process, only the owner and NetBatch supervisors can release the job.
- Only one of *dependent-job-ID* and *master-job-ID* can contain wild-card characters, not both.
- BATCHCOM confirms the release of a job by the RELEASE-WAITON command with the message:

  Job *job-name* Jobnumber *job-number* altered
- The scheduler records the release of a job by the RELEASE-WAITON command by writing an UPDATE event to the job and scheduler log files. For example:

```
UPDATE JOB X C_DEFAULT:3 J_24 U_255,255
H_\MELRISK.$ZTN0.#PTY4
```

Example

This example shows the effect of the RELEASE-WAITON command:

```
18} INFO JOB *, WAITON
JOB ATTRIBUTES for X
jobnumber: 24
waiton: Z, Not Released
JOB ATTRIBUTES for Y
jobnumber: 25
waiton: Z, Not Released
19} RELEASE-WAITON * FROM Z
Job Y Jobnumber 25 altered
Job X Jobnumber 24 altered
20} INFO JOB *, WAITON
JOB ATTRIBUTES for X
jobnumber: 24
waiton: Z, Released
JOB ATTRIBUTES for Y
jobnumber: 25
waiton: Z, Released
```

REPORT JOB Command

Use the REPORT JOB command to generate reports of jobs have been submitted and run.

```
REPORT JOB
[ { job-ID } ]
[ { CLASS class-name } ]
, TYPE rptype
[ , START-TIME rptime ]   [ , END-TIME rptime  ]
```

job-ID

specifies a job name or number, or a range of job names. To specify a range of job names, use the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all jobs.

class-name

is the class for which the report is to be generated. It is used only and mandatorily with MISC-REPORTs (see rptype below).

rptype

is one of these report types:

HOLD-REPORT	Job-Submit	Jobs that are put on hold
MISC-REPORT	Job-Submit	Jobs configured under a particular class, or satisfying the *job-filter* attribute
PRED-REPORT	Job-Submit	Master jobs of jobs satisfying the*job-filter* attribute
SUCR-REPORT	Job-Submit	Dependent jobs of jobs satisfying the *job-filter* attribute
TAPE-REPORT	Job-Submit	Jobs that require tapedrives to run
ABEND-REPORT	Job-Run	Job RunJobs that abended
LATE-REPORT	Job-Run	Job RunJobs not run at the scheduled time
RUNNOW-REPORT	Job-Run	Job RunJobs started by the RUNNOW command
SUCCESS-REPORT	Job-Run	Job RunJobs that are complete successfully

rptime

specifies the job submit time if the report is valid after job submit, or the job start time if the report is valid after job run.

Consideration

The REPORT-JOB command is available to all users.

RESET ATTACHMENT-SET Command

Use the RESET ATTACHMENT-SET command to remove attachment-set attributes from BATCHCOM's working-attributes set.

```
RESE[T] [ ATTACHMENT-SET ] [ attribute [ , attribute ]… ]
```

attribute

is one of these attachment-set attributes:

```
ASSI[GN] [ ASSIGN-name ]
DEFI[NE] [ DEFINE-name ]
PA[RAM] [ PARAM-name ]
SEC[URITY]
TEM[PORARY]
```

If you omit *attribute*, BATCHCOM removes all attachment-set attributes except DEFINE =_DEFAULTS from the working-attributes set. If you specify ASSIGN,

DEFINE, or PARAM, but omit *ASSIGN-name*, *DEFINE-name*, or *PARAM-name*, BATCHCOM removes all ASSIGNs, DEFINEs (except =_DEFAULTS), or PARAMs.

Considerations

- The RESET ATTACHMENT-SET command is available to all users.
- The RESET ATTACHMENT-SET command, like the OPEN and SET ATTACHMENT-SET commands, sets the #CURRENT variable's value to null.
- ADP (an abbreviation of ASSIGNs, DEFINEs, and PARAMs) is an alias of the object keyword ATTACHMENT-SET.
- You can omit the object keyword ATTACHMENT-SET from the RESET ATTACHMENT-SET command only when ATTACHMENT-SET is the current assumed object. For more information, see ASSUME ATTACHMENT-SET Command on page 6-69.

Examples

- The RESET ATTACHMENT-SET command in this example removes all attachment-set attributes except the DEFINE =_DEFAULTS from the working-attributes set:

```
3} SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
temporary: On
attachments: DEFINE =INFILE, CLASS MAP, FILE $A.NB.INFILE
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
ASSIGN FILE-X, $A.NB.X
ASSIGN FILE-Y, $A.NB.Y
PARAM DATE 19JUN93
PARAM TIME 2:39:12
4} RESET ATTACHMENT-SET; SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
```

- This example shows the RESET ATTACHMENT-SET command removing two PARAMs from the working-attributes set:

```
36} SHOW ATTACHMENT-SET PARAM
ATTACHMENT-SET ATTRIBUTES
attachments: PARAM ADDR1 19333_VALLCO_PARKWAY
PARAM ADDR2 CUPERTINO
PARAM ADDR3 CA_95014
PARAM ADDR4 USA
PARAM ADDR5 408_725-6000
37} RESET ATTACHMENT-SET PARAM ADDR4, PARAM ADDR5
38} SHOW ATTACHMENT-SET PARAM
ATTACHMENT-SET ATTRIBUTES
attachments: PARAM ADDR1 19333_VALLCO_PARKWAY
```

```
PARAM ADDR2 CUPERTINO
PARAM ADDR3 CA_95014
```

- In this example, the RESET ATTACHMENT-SET command removes the TEMPORARY attribute from the working-attributes set and all ASSIGN attributes. The command also removes the DEFINE =TAPE but leaves all other DEFINEs, the PARAMs, and the SECURITY attribute.

```
33} SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
security: "AOAO"
temporary: On
attachments: DEFINE =SQLCAT, CLASS CATALOG, SUBVOL
\MELBDEV.$SYSTEM.SYSTEM2
DEFINE =TAPE, CLASS TAPE, LABELS ANSI, OWNER
"255,205", FILEID ACCNTBAL, DEVICE
$TAPE1
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
ASSIGN X, $A.NB.X
ASSIGN Y, $A.NB.Y
PARAM DAY WEDNESDAY
PARAM MONTH JUN
PARAM YEAR 1993
34} RESET ATTACHMENT-SET TEMPORARY, ASSIGN, DEFINE =TAPE
35} SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
security: "AOAO"
attachments: DEFINE =SQLCAT, CLASS CATALOG, SUBVOL
\MELBDEV.$SYSTEM.SYSTEM2
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
PARAM DAY WEDNESDAY
PARAM MONTH JUN
PARAM YEAR 1993
```

RESET CLASS Command

Use the RESET CLASS command to restore the value of the INITIATION class attribute in BATCHCOM's working-attributes set to its default (ON).

```
RESE[T] [ CLASS ]
```

Considerations

- The RESET CLASS command is available to all users.
- JOBCLASS is an alias of the object keyword CLASS.
- You can omit the object keyword CLASS from the RESET CLASS command only when CLASS is the current assumed object. For more information, see ASSUME CLASS Command on page 6-70.

Example

This example shows the RESET CLASS command restoring the default value of the INITIATION attribute in the working-attributes set:

```
6} SHOW CLASS
CLASS ATTRIBUTE
initiation: Off
7} RESET CLASS
8} SHOW CLASS
CLASS ATTRIBUTE
initiation: On
```

RESET EXECUTOR Command

Use the RESET EXECUTOR command to remove executor attributes from BATCHCOM's working-attributes set.

```
RESE[T] [ EXECUTOR ] [ attribute [ , attribute ] ]
```

attribute

is one of these executor attributes:

```
CL[ASS]
CP[U]
```

If you omit attribute, BATCHCOM removes all executor attributes from the working-attributes set.

Considerations

- The RESET EXECUTOR command is available to all users.
- You can omit the object keyword EXECUTOR from the RESET EXECUTOR command only when EXECUTOR is the current assumed object. For more information, see ASSUME EXECUTOR Command on page 6-71.

Examples

- This example shows the RESET EXECUTOR command removing executor attributes from the working-attributes set:

```
4} SHOW EXECUTOR
EXECUTOR ATTRIBUTES
cpu: 2
classes: SERVICE
INQUIRIES
SALES
5} RESET EXECUTOR
6} SHOW EXECUTOR
EXECUTOR ATTRIBUTES
7}
```

- To use the RESET EXECUTOR command to remove the CPU executor attribute from the working-attributes set:

```
1} SET EXECUTOR CLASS *, CPU 0
2} SHOW EXECUTOR
EXECUTOR ATTRIBUTES
cpu: 0
classes: *
3} RESET EXECUTOR CPU
4} SHOW EXECUTOR
EXECUTOR ATTRIBUTES
classes: *
```

RESET JOB Command

Use the RESET JOB command to remove job attributes from BATCHCOM's working-attributes set.

```
RESE[T] [ JOB ] [ attribute [ , attribute ]… ]
```

attribute

is one of these job attributes:

AF[TER]	IN	REST[ART]
AT	J[OB]-L[OG]	RUND
A[TTACHMENT]-S[ET]	J[OBID]-Z[ERO]	SA[VEABEND]
CA[LENDAR]	LIB	SEL[PRI]
CL[ASS]	LIM[IT]	STAL[L]
DES[CRIPTION]	MAXPRINTL[INES]	STARTU[P]
EV[ERY]	MAXPRINTP[AGES]	S[TOP]-[ON]-[ABEND]
E[XECUTOR]-P[ROGRAM]	ME[M]	SWA[P]
EXT[SWAP]	NA[ME]	TA[PEDRIVES]
HIG[HPIN]	OU[T]	V[OLUME]
HOLD	PF[S]	WAIT
HOLDA[FTER]	PR[I]	WAITO[N]
IF[FAILS]	P[URGE]-[IN]-[FILE]	

If you omit *attribute*, BATCHCOM removes all job attributes from the working-attributes set except JOBID-ZERO and VOLUME. BATCHCOM restores the values of these attributes to their defaults.

Considerations

- The RESET JOB command is available to all users.

- You can omit the object keyword JOB from the RESET JOB command only when JOB is the current assumed object. For more information, see ASSUME JOB Command on page 6-72.

Examples

- This example shows the RESET JOB command removing the HOLD and AFTER job attributes from the working-attributes set:

```
6} SHOW JOB
JOB ATTRIBUTES
volume: \MELBDEV.$A.NB, "AAAA"
jobid-zero: Off
class: REPORTS
hold: On
after: 22:00:00
user: 205,70
7} RESET JOB HOLD, AFTER
8} SHOW JOB
JOB ATTRIBUTES
volume: \MELBDEV.$A.NB, "AAAA"
jobid-zero: Off
class: REPORTS
user: 205,70
```

- This example shows the RESET JOB command removing all job attributes from the working-attributes set except the JOBID-ZERO and VOLUME attributes. The command restores the default values of the attributes.

```
13} SHOW JOB
JOB ATTRIBUTES
volume: \MELBQAT.$QAT2.TEMP, "AAAA"
in: \DEV.$SYSTEM.FILES.XYZ
out: \DEV.$S.#LPT1
jobid-zero: On
user: 255,133
14} RESET JOB
15} SHOW JOB
JOB ATTRIBUTES
volume: \DEV.$SYSTEM.FILES, "NCNC"
jobid-zero: Off
user: 255,133
```

RESET SCHEDULER Command

Use the RESET SCHEDULER command to remove scheduler attributes from BATCHCOM's working-attributes set.

```
RESE[T] [ SCHEDULER ] [ attribute [ , attribute ]… ]
```

attribute

is one of these scheduler attributes:

AT-A[LLOWED]	D[EFAULT]-MAXPRINTP[AGES]	INI[TIATION]
B[ACKUPCPU]	D[EFAULT]-O[UT]	LO[CALNAMES]
CAT[CHUP]	D[EFAULT]-P[RI]	M[AX]-[CONCURRENT]-[JOBS]
D[EFAULT]-C[LASS]	D[EFAULT]-SE[LPRI]	M[AX]-[PRI]
D[EFAULT]-E[XECUTOR]-[PROGRAM]	D[EFAULT]-ST[ALL]	S[UBMIT]-A[LLOWED]
D[EFAULT]-H[IGHPIN]	D[EFAULT]-[STOP]-[ON]-[ABEND]	TA[PEDRIVES]
D[EFAULT]-MAXPRINTL[INES]	EM[S]	

If you omit *attribute*, BATCHCOM removes all scheduler attributes from the working-attributes set.

Considerations

- The RESET SCHEDULER command is available to all users.
- You can omit the object keyword SCHEDULER from the RESET SCHEDULER command only when SCHEDULER is the current assumed object. For more information, see ASSUME SCHEDULER Command on page 6-73.

Examples

- This example shows the RESET SCHEDULER command removing two specified scheduler attributes from the working-attributes set:

```
4} SHOW SCHEDULER
SCHEDULER ATTRIBUTES
submit-allowed: On
default-executor-program: \MELBDEV.$SYSTEM.SYSTEM.TACL
default-pri: 119
default-selpri: 3
default-out: \MELBDEV.$S.#FPP
default-class: STANDARD-CLASS
default-stop-on-abend: Off
5} RESET SCHEDULER DEFAULT-OUT, DEFAULT-STOP-ON-ABEND
6} SHOW SCHEDULER
SCHEDULER ATTRIBUTES
submit-allowed: On
default-executor-program: \MELBDEV.$SYSTEM.SYSTEM.TACL
default-pri: 119
default-selpri: 3
default-class: STANDARD-CLASS
```

- This example shows the RESET SCHEDULER command removing all scheduler attributes from the working-attributes set:

```
3} SHOW SCHEDULER
SCHEDULER ATTRIBUTES
backupcpu: 1,2
at-allowed: On
submit-allowed: On
4} RESET SCHEDULER
5} SHOW SCHEDULER
SCHEDULER ATTRIBUTES
```

RUN Command

Use the RUN command to run a program during a BATCHCOM session.

```
RUN program-file [ / OU[T] [ list-file ] / ] [ param-set ]
```

program-file

is the name of a program file. BATCHCOM expands a partially qualified file name by using the defaults specified in the last VOLUME command. If there was no such command, the defaults current at the start of the session apply. If the resulting expanded name specifies a nonexistent file, BATCHCOM uses $SYSTEM.SYSTEM for expansion purposes.

list-file

is the output file of the new process. If you omit OUT *list-file*, BATCHCOM's home terminal is the output file. If you include OUT with no *list-file*, spaces are sent as the name of the output file. *list-file* can specify a DEFINE from BATCHCOM's working-attributes set.

param-set

is one or more program parameters sent to the new process in the startup message. BATCHCOM deletes leading and trailing spaces.

Considerations

- The RUN command is available to all users.
- The program you run uses BATCHCOM's defaults for file-name expansion. These defaults are the same as those used by BATCHCOM to expand program-file.
- Stopping program-file returns the BATCHCOM prompt.
- You can disable the RUN command with the NBFLAGS procedure. For details, see on page 2-3.

Example

This example illustrates the use of the RUN command during a BATCHCOM session:

```
2} RUN EDIT; GET $QA.TESTS.FILES; LIST ALL
TEXT EDITOR - T9601D20 - (01JUN93)
CURRENT FILE IS $QA.TESTS.FILES
1 FILES /OUT \MELBDEV.$ZTN0.#PTY4/ $NB.T9190MAN
*ADD
1 FILES /OUT \MELBDEV.$ZTN0.#PTY4/ $NB.T9190MAN
2 FILES /OUT \MELBDEV.$ZTN0.#PTY4/ $SYSTEM.SYSTEM
3 //
*EXIT
3}
```

RUNNEXT JOB Command

Use the RUNNEXT JOB command to make the scheduler run a job immediately when an executor associated with the job's class is available. The command overrides the job's dependencies, timing attributes, and selection priority.

```
RUNNE[XT] [ JOB ] [ / OU[T] [ file-name ] / ]
{ job-ID }
{ ( [ [NOT] job-ID ,]… [NOT] filter [ , [NOT] filter ]… )
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

job-ID

specifies the name, number, or range of names of a job or jobs whose state is EVENT, READY, RUNNOW, TAPE, or TIME. (*job-ID* cannot specify jobs whose states are EXECUTING, OVER LIMIT, RUNNEXT, SPECIAL-*n*, or SUSPENDED.) To specify a range of job names, use the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all jobs.

NOT

excludes jobs that satisfy the specified job-selection criterion.

filter

specifies one of these job-selection criteria (see STATUS JOB Command on page 6-165):

```
A[TTACHMENT]-S[ET] [ ( user-ID ) ] attachment-set-ID
#CURRENT ]
CL[ASS] class-ID
```

```
IN file-ID
STATE state
U[SER] user-ID
WAITO[N] master-job
```

Considerations

- The RUNNEXT JOB command is available to NetBatch supervisors only.
- A job whose state is EVENT has the WAITON attribute and does not run until released by its masters. The RUNNEXT JOB command overrides the WAITON attribute and makes the job eligible to run, regardless of the completion states of its masters.
- If several jobs from a class are in a RUNNEXT state, the scheduler selects them based on their SELPRI attributes. Selection of jobs with the same SELPRI attribute is by submission time on a first-in, first-out basis.
- A job that is subject to a RUNNEXT JOB command does not run if either of these conditions exists:
  - The job has the TAPEDRIVES attribute and requires more drives than are available. For more information about this attribute, see TAPEDRIVES Job Attribute on page 7-109 and TAPEDRIVES Scheduler Attribute on page 7-110.
  - The job's class has the attribute INITIATION OFF. This attribute prevents jobs belonging to the class from running. To make the jobs available for execution, change the value of the INITIATION attribute to ON by using the ALTER CLASS command.
- The RUNNEXT JOB command, unlike RUNNOW JOB, does not make the scheduler create a temporary executor.
- You can omit the object keyword JOB from the RUNNEXT JOB command only when JOB is the current assumed object. For more information, see ASSUME JOB Command on page 6-72.

Examples

- This example shows execution of a RUNNEXT JOB command and the resulting scheduler log-file event:

```
> BATCHCOM $ZBAT; STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- -------------
70 FIRST 255,255 670 EXECUTING DEFAULT
71 SECOND 255,255 READY DEFAULT
72 THIRD 255,255 READY DEFAULT
> BATCHCOM $ZBAT; RUNNEXT JOB THIRD
Job THIRD Jobnumber 72 will run next
> BATCHCOM $ZBAT; STATUS JOB *
JOB STATUS
```

```
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
70 FIRST 255,255 670 EXECUTING DEFAULT
71 SECOND 255,255 READY DEFAULT
72 THIRD 255,255 RUNNEXT DEFAULT
> FUP COPY LOGABR,,SHARE .
.
LIST JOB THIRD RUNNEXT J_72 U_255,255 H_\QA.$ZTN0.#PTY4
```

- This example shows two RUNNEXT JOB commands making jobs SELPRI-0 and SELPRI-1 run immediately after job SELPRI-7 finishes. The commands make job SELPRI-1 the highest priority job in class DEFAULT after SELPRI-7, followed by SELPRI-0, then SELPRI-6, and so on.

```
40} STATUS JOB *, SELPRI
JOB STATUS
JOB JOBNAME USERID SEL STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
73 SELPRI-0 255,205 0 READY DEFAULT
74 SELPRI-1 255,205 1 READY DEFAULT
75 SELPRI-2 255,205 2 READY DEFAULT
76 SELPRI-3 255,205 3 READY DEFAULT
77 SELPRI-4 255,205 4 READY DEFAULT
78 SELPRI-5 255,205 5 READY DEFAULT
79 SELPRI-6 255,205 6 READY DEFAULT
80 SELPRI-7 255,205 7 EXECUTING DEFAULT
41} RUNNEXT JOB 73; RUNNEXT JOB 74
Job SELPRI-0 Jobnumber 73 will run next
Job SELPRI-1 Jobnumber 74 will run next
42} STATUS JOB *, SELPRI
JOB STATUS
JOB JOBNAME USERID SEL STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
73 SELPRI-0 255,205 0 RUNNEXT DEFAULT
74 SELPRI-1 255,205 1 RUNNEXT DEFAULT
75 SELPRI-2 255,205 2 READY DEFAULT
76 SELPRI-3 255,205 3 READY DEFAULT
77 SELPRI-4 255,205 4 READY DEFAULT
78 SELPRI-5 255,205 5 READY DEFAULT
79 SELPRI-6 255,205 6 READY DEFAULT
80 SELPRI-7 255,205 7 EXECUTING DEFAULT
43} STATUS JOB *, SELPRI
JOB STATUS
JOB JOBNAME USERID SEL STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
73 SELPRI-0 255,205 0 RUNNEXT DEFAULT
74 SELPRI-1 255,205 1 EXECUTING DEFAULT
75 SELPRI-2 255,205 2 READY DEFAULT
76 SELPRI-3 255,205 3 READY DEFAULT
77 SELPRI-4 255,205 4 READY DEFAULT
78 SELPRI-5 255,205 5 READY DEFAULT
79 SELPRI-6 255,205 6 READY DEFAULT
```

RUNNOW JOB Command

Use the RUNNOW JOB command to make the scheduler run a job immediately. The command overrides job dependencies, timing attributes, and selection priority and makes the scheduler create a temporary executor for the job.

```
RUNNO[W] [ JOB ] [ / OU[T] [ file-name ] / ]
{ job-ID }
{ ( [ [NOT] job-ID ,]… [NOT] filter [ , [NOT] filter ]… )
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

job-ID

specifies the name, number, or range of names of a job or jobs whose state is EVENT, READY, RUNNEXT, TAPE, or TIME. (*job-ID* cannot specify jobs whose states are EXECUTING, OVER LIMIT, RUNNOW, SPECIAL-*n*, or SUSPENDED.) To specify a range of job names, use the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all jobs.

NOT

excludes jobs that satisfy the specified job-selection criterion.

filter

specifies one of these job-selection criteria (see STATUS JOB Command on page 6-165):

```
A[TTACHMENT]-S[ET] [ ( user-ID ) ] attachment-set-ID
#CURRENT ]
CL[ASS] class-ID
IN file-ID
STATE state
U[SER] user-ID
WAITO[N] master-job
```

Considerations

- The RUNNOW JOB command is available to NetBatch supervisors only.
- A job whose state is EVENT has the WAITON attribute and does not run until released by its masters. The RUNNOW JOB command overrides the attribute and forces the job to run, regardless of the completion states of its masters.

- The scheduler creates temporary executors for all jobs with the AT attribute and for all jobs that are operated on by the RUNNOW JOB command. The scheduler deletes the executors when the jobs finish.
- A temporary executor has a scheduler-assigned name of the form __TEMP_EXEC_ *job-number*, where *job-number* is the number of the job using the executor; for example, __TEMP_EXEC_496.
- The scheduler selects the CPU of a temporary executor from available CPUs on the scheduler's node.
- A job that is subject to a RUNNOW JOB command does not run if any of these conditions exists:
  - The job has the TAPEDRIVES attribute and requires more drives than are available. For more information about this attribute, see TAPEDRIVES Job Attribute on page 7-109 and TAPEDRIVES Scheduler Attribute on page 7-110.
  - The job's class has the attribute INITIATION OFF. This attribute prevents jobs belonging to the class from running. To make the jobs available for execution, change the value of the INITIATION attribute to ON by using the ALTER CLASS command.
  - The scheduler would exceed its temporary-executors limit by running the job. In this case, the job runs when the scheduler can create a temporary executor without exceeding the limit. For more information, see MAX-CONCURRENT-JOBS Scheduler Attribute on page 7-79.
- You can omit the object keyword JOB from the RUNNOW JOB command only when JOB is the current assumed object. For more information, see ASSUME JOB Command on page 6-72.
- If the scheduler's INITIATION attribute is OFF, the RUNNOW command does not run the jobs submitted to the scheduler. After RUNNOW is issued, the jobs are in the RUNNOW state until the scheduler INITIATION attribute is set to ON using the ALTER SCHEDULER command.

Example

This example shows the submission of a job to a scheduler that has no executors. The RUNNOW JOB command makes the scheduler run the job in a temporary executor, which the scheduler deletes when the job finishes. The example lists the scheduler log-file events that record job submission, temporary-executor creation, job execution, and so on.

```
53> BATCHCOM $ZBAT; STATUS EXECUTOR *
2117-I No EXECUTOR selected
54> BATCHCOM $ZBAT; SUBMIT JOB, EXECUTOR-PROGRAM DELAY,
STARTUP
"1 MINS"
Job ZBAT-0082 Jobnumber 82 submitted
55> BATCHCOM $ZBAT; STATUS JOB *
```

```
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ---------- ---------
82 ZBAT-0082 255,205 READY DEFAULT
56> BATCHCOM $ZBAT; RUNNOW JOB 82
Job ZBAT-0082 Jobnumber 82 runnow completed
57> BATCHCOM $ZBAT; STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ---------- ---------
82 ZBAT-0082 255,205 681 EXECUTING DEFAULT
58> BATCHCOM $ZBAT; STATUS EXECUTOR *
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
---------------------- --- ------ ---- -------
__TEMP_EXEC_82 3 DELETE 82 DEFAULT
59> FUP COPY LOGABU,,SHARE .
.
ADD JOB ZBAT-0082 C_DEFAULT:2 J_82 U_255,205
H_\QA.$ZTN0.#PTY4
LIST JOB ZBAT-0082 RUNNOW J_82 U_255,205 H_\QA.$ZTN0.#PTY4
ADD EXECUTOR __TEMP_EXEC_82 CPU 3
BEGIN JOB (SUPER.FPP)ZBAT-0082:1 E___TEMP_EXEC_82 L_681 J_82
P_DELAY \QA.$Z703:17393453 U_255,205
UPDATE EXECUTOR __TEMP_EXEC_82
LIST JOB ZBAT-0082 EXECUTING J_82
START EXECUTOR-PROGRAM U_255,205 J_82 P_DELAY
\QA.$Z703:17393453
STOP CC_0 EXECUTOR-PROGRAM J_82 \QA.$Z703:17393453
DELETE EXECUTOR __TEMP_EXEC_82
FINISH JOB ZBAT-0082 T_0:0:0:12 J_82 P_DELAY
DELETE JOB ZBAT-0082 J_82
```

SET ATTACHMENT-SET Command

Use the SET ATTACHMENT-SET command to specify attachment-set attributes in BATCHCOM's working-attributes set. Attachment sets added later by the ADD ATTACHMENT-SET command adopt these attributes (which override attachment-set defaults) unless that command specifies otherwise.

```
SET [ ATTACHMENT-SET ] attribute [ , attribute ]…
```

attribute

is one of these attachment-set attributes (see Section 7, Attributes):

```
( ASSI[GN] ASSIGN-name , ASSIGN-attributes )
( DEFI[NE] DEFINE-name-1 , [ LIK[E] DEFINE-name-2 , ]
[ CLASS DEFINE-class , ]
DEFINE-attribute [ , DEFINE-attribute ]… )
( PA[RAM] PARAM-name PARAM-value )
SEC[URITY] " security "
TEM[PORARY] { OF[F] | ON }
```

Considerations

- The SET ATTACHMENT-SET command is available to all users.
- You can list attachment-set attributes in the working-attributes set by using the SHOW ATTACHMENT-SET command.
- Two types of default attachment-set attributes are displayed in the working-attributes set: temporary and permanent.
- Temporary defaults are the ASSIGNs, DEFINEs, and PARAMs that BATCHCOM inherits from the TACL environment. They are defaults because they are automatically available when a session begins. They are temporary defaults, however, because you can delete them during the session by using the RESET ATTACHMENT-SET command. (You also can use this command to delete attributes set by the SET ATTACHMENT-SET command.)
- The only permanent default attribute is defaults DEFINE =_DEFAULTS. Like temporary default attributes, this DEFINE comes from the TACL environment. Unlike those attributes, you cannot delete it.
- DEFINE =_DEFAULTS specifies the node and volume set by the last SET ATTACHMENT-SET (DEFINE =_DEFAULTS, VOLUME …), SYSTEM, or VOLUME command. If there was no such command, the DEFINE specifies the current node and volume from the TACL environment.
- The LIKE qualifier is available for DEFINEs only. Use it in the SET ATTACHMENT-SET command to specify DEFINEs whose attributes match those of DEFINEs specified earlier in the command. You also can specify DEFINEs whose attributes match those of existing DEFINEs in the working-attributes set.
- The SET ATTACHMENT-SET command, like the OPEN and RESET ATTACHMENT-SET commands, sets the #CURRENT variable's value to null.

  ADP (an abbreviation of ASSIGNs, DEFINEs, and PARAMs) is an alias of the object keyword ATTACHMENT-SET.
- You can omit the object keyword ATTACHMENT-SET from the SET ATTACHMENT-SET command only when ATTACHMENT-SET is the current assumed object. For more information, see ASSUME ATTACHMENT-SET Command on page 6-69.

Examples

- This example first shows attachment-set attributes in the working set. It then shows a SET ATTACHMENT-SET command adding attributes to the set and altering an existing attribute. Finally, the example displays the updated set before showing the effect of the RESET ATTACHMENT-SET command.

```
1} SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
```

```
$A.NB
ASSIGN ABC, $A.NB.ABC
PARAM X 24
2} SET ATTACHMENT-SET (ASSIGN DEF, $A.NB.DEF), (DEFINE =GHI,
CLASS MAP, FILE $A.NB.GHI), (PARAM Y 25), (ASSIGN ABC,
$DATA7.NB.ABC); SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =GHI, CLASS MAP, FILE \DEV.$A.NB.GHI
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
ASSIGN ABC, $DATA7.NB.ABC
ASSIGN DEF, $A.NB.DEF
PARAM X 24
PARAM Y 25
3} RESET ATTACHMENT-SET; SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
```

- This example shows the LIKE qualifier used in a SET ATTACHMENT-SET command:

```
6} SET ATTACHMENT-SET (DEFINE =A1, CLASS MAP, FILE $A.NB.X),
(DEFINE =B1, LIKE =A1), (DEFINE =C1, LIKE =B1)
7} SHOW ATTACHMENT-SET DEFINE =*1
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =A1, CLASS MAP, FILE \DEV.$A.NB.X
DEFINE =B1, CLASS MAP, FILE \DEV.$A.NB.X
DEFINE =C1, CLASS MAP, FILE \DEV.$A.NB.X
```

- This example shows the SET ATTACHMENT-SET command specifying various attachment-set attributes in the working-attributes set. The set added after execution of the command adopts the attributes. Note the use of the LIKE qualifier.

```
10} SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
11} SET ATTACHMENT-SET (ASSIGN ACCOUNT-REC, $D6.FIN.ACCNT), &
}} (DEFINE =S, CLASS SPOOL, OWNER "255,1", LOC $S), &
}} (DEFINE =I, LIKE =S, LOC $I), (PARAM SHIFT NIGHT), &
}} SECURITY "AAAA", TEMPORARY OFF
12} SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
security: "AAAA"
temporary: Off
attachments: DEFINE =I, CLASS SPOOL, LOC \DEV.$I, OWNER
"255,1"
DEFINE =S, CLASS SPOOL, LOC \DEV.$S, OWNER
"255,1"
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
ASSIGN ACCOUNT-REC, $D6.FIN.ACCNT
PARAM SHIFT NIGHT
13} ADD ATTACHMENT-SET #CURRENT
Attachment-set (SUPER.DRAT)2 added
```

```
14} INFO ATTACHMENT-SET (SUPER.DRAT)2, DETAIL
ATTACHMENT-SET ATTRIBUTES for (SUPER.DRAT)2
security: "AAAA"
temporary: Off
attachments: DEFINE =I, CLASS SPOOL, LOC \DEV.$I, OWNER
"255,1"
DEFINE =S, CLASS SPOOL, LOC \DEV.$S, OWNER
"255,1"
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
ASSIGN ACCOUNT-REC, $D6.FIN.ACCNT
PARAM SHIFT NIGHT
```

SET CLASS Command

Use the SET CLASS command to specify the value of the INITIATION class attribute in BATCHCOM's working-attributes set. Classes added later by the ADD CLASS command adopt this attribute unless that command specifies otherwise.

```
SET [ CLASS ] attribute
```

attribute

specifies this class attribute (see Section 7, Attributes):

```
INI[TIATION]BBC{( AALVS1(OF[F],ON) )
```

Considerations

- The SET CLASS command is available to all users. However, you must be a NetBatch supervisor to use the ADD CLASS command.
- You can display the INITIATION attribute in the working-attributes set by using the SHOW CLASS command.
- The default value of the INITIATION attribute in the working-attributes set is ON. You can use the RESET CLASS command to restore this value during a BATCHCOM session.
- JOBCLASS is an alias of the object keyword CLASS.
- You can omit the object keyword CLASS from the SET CLASS command only when CLASS is the current assumed object. For more information, see ASSUME CLASS Command on page 6-70.

Example

This example shows the SET CLASS command setting the value of the INITIATION attribute in the working-attributes set. The class added after execution of the command adopts the attribute.

```
1} SHOW CLASS
CLASS ATTRIBUTE
initiation: On
2} SET CLASS INITIATION OFF
3} SHOW CLASS
CLASS ATTRIBUTE
initiation: Off
4} ADD CLASS OPS-DAYSHIFT
Class OPS-DAYSHIFT added
5} INFO CLASS OPS-DAYSHIFT
CLASS ATTRIBUTE for OPS-DAYSHIFT
initiation: Off
```

SET EXECUTOR Command

Use the SET EXECUTOR command to specify executor attributes in BATCHCOM's working-attributes set. Executors added later by the ADD EXECUTOR command adopt these attributes (which override executor defaults) unless that command specifies otherwise.

```
SET [ EXECUTOR ] [ LIK[E] executor-name , ]
attribute [ , attribute ]
```

`LIKE`

specifies that all attributes are to match those of executor *`executor-name`*.

`executor-name`

is the name of an executor.

`attribute`

is one of these executor attributes (see Section 7, Attributes):

```
CL[ASS] { class-name | ( class-name [ , class-name ]… ) | * }
CP[U] cpu-number
```

Considerations

- The SET EXECUTOR command is available to all users. However, you must be a NetBatch supervisor to use the ADD EXECUTOR command.
- You can list executor attributes in the working-attributes set by using the SHOW EXECUTOR command.

- No default executor attributes are displayed in the working-attributes set. To remove attributes added to the set by the SET EXECUTOR command, use the RESET EXECUTOR command.
- You can omit the object keyword EXECUTOR from the SET EXECUTOR command only when EXECUTOR is the current assumed object. For more information, see ASSUME EXECUTOR Command on page 6-71.

Examples

- This example shows the SET EXECUTOR command specifying executor attributes in the working-attributes set. The executor added after execution of the command adopts the attributes.

```
14} SHOW EXECUTOR
EXECUTOR ATTRIBUTES
15} SET EXECUTOR CLASS *, CPU 0
16} SHOW EXECUTOR
EXECUTOR ATTRIBUTES
cpu: 0
classes: *
17} ADD EXECUTOR DEFAULT-EXEC-0
Executor DEFAULT-EXEC-0 added
18} INFO EXECUTOR DEFAULT-EXEC-0
EXECUTOR ATTRIBUTES for DEFAULT-EXEC-0
cpu: 0
classes: *
```

- This example shows the effect of the SET EXECUTOR command's LIKE qualifier:

```
46} SHOW EXECUTOR
EXECUTOR ATTRIBUTES
47} INFO EXECUTOR RISC-3
EXECUTOR ATTRIBUTES for RISC-3
cpu: 3
classes: ACCOUNTS
ADMINISTRATION
48} SET EXECUTOR LIKE RISC-3
49} SHOW EXECUTOR
EXECUTOR ATTRIBUTES
cpu: 3
classes: ACCOUNTS
ADMINISTRATION
```

SET JOB Command

Use the SET JOB command to specify job attributes in BATCHCOM's working-attributes set. Jobs submitted later by the SUBMIT JOB command adopt these attributes (which override job defaults) unless that command specifies otherwise.

```
SET [ JOB ] [ LIK[E] job-ID , ] attribute [ , attribute ]…
```

`LIKE`

specifies that all attributes are to match those of job *job-ID*.

job-ID

specifies a job name or job number.

attribute

is one of these job attributes (see Section 7, Attributes):

```
AF[TER] [ date ] [ time ]
AT [ date ] [ time ]
A[TTACHMENT]-S[ET]
[ attachment-set | ( attachment-set [ , attachment-set ]… ) ]
CA[LENDAR] [ file-name ]
CL[ASS] class-name
DES[CRIPTION] " [ string ] "
EV[ERY] [ weeks WEEK[S]
days D[AYS]
hours [ : mins ] [HOURS]
hours H[OURS] [ mins MIN[UTES] ]
crontab-entry) ]
E[XECUTOR]-P[ROGRAM] file-name
EXT[SWAP]{ $volume-name | file-name }
HIG[HPIN] { OF[F] | ON }
HOLD { OF[F] | ON }
HOLDA[FTER] { OF[F] | ON }
IF[FAILS] { OF[F] | ON }
IN [ file-name ]
J[OB]-L[OG] [ log-file ]
J[OBID]-Z[ERO] { OF[F] | ON }
LIB [ file-name ]
LIM[IT] hours [ : mins ]
MAXPRINTL[INES] { number | NON[E] }
MAXPRINTP[AGES] { number | NON[E] }
ME[M] number
NA[ME] $process-name
OU[T] [ file-name ]
PF[S] number
PR[I] number
P[URGE]-[IN]-[FILE] { OF[F] | ON }
REST[ART] { OF[F] | ON }
RUNDBBC { OF[F] | ON }
SA[VEABEND] { OF[F] | ON }
```

```
SEL[PRI] number
STAL[L] { OF[F] | ON }
STARTU[P] " param-set "
S[TOP]-[ON]-[ABEND] { OF[F] | ON }
SWA[P] {$volume-name | file-name }
TA[PEDRIVES] number
TERM $process-name
V[OLUME] { \node. [ volume ] [ , " security " ]
[ \node. ] volume [ , " security " ]
[ \node. ] [ volume ] , " security " }
WAIT hours [ : mins ]|
WAITO[N] [ job-name [ case ]
( job-name [ case ] [ , job-name [ case ] ]… ) ]
```

Considerations

- The SET JOB command is available to all users.
- To list job attributes in the working-attributes set, use the SHOW JOB command.
- The default job attributes in the working-attributes set are JOBID-ZERO and VOLUME. The default value of JOBID-ZERO is OFF. The default value of VOLUME is the node, volume, subvolume, and security defaults set by the last SET JOB VOLUME command. If there was no such command, the attribute specifies the defaults current when the session began.
- To remove all job attributes from the working-attributes set except JOBID-ZERO and VOLUME and to set the default values of JOBID-ZERO and VOLUME, use the RESET JOB command.
- ATTACHMENT-SET cannot be the first or only attribute specified by *attribute* if you omit the object keyword JOB from the SET JOB command. BATCHCOM interprets SET ATTACHMENT-SET as a command instead of a SET JOB command that specifies the ATTACHMENT-SET job attribute. For example:

  ```
  19} ASSUME JOB; SET ATTACHMENT-SET MY-PARAMS, OUT $S.#OUT
                                          -^-

  0290-E Invalid command SET ATTACHMENT-SET
  ```

  To prevent the SET command from failing in these circumstances, do not specify ATTACHMENT-SET as the first attribute in a list, or include the object keyword JOB in the command. For example:

  ```
  SET OUT $S.#OUT, ATTACHMENT-SET MY-PARAMS
  SET JOB ATTACHMENT-SET MY-PARAMS, OUT $S.#OUT
  ```

- You can omit the object keyword JOB from the SET JOB command only when JOB is the current assumed object. For more information, see ASSUME JOB Command on page 6-72.

Examples

- This example shows the SET JOB command specifying a series of job attributes in the working-attributes set. Note the effect of the RESET JOB command.

```
59} SHOW JOB
JOB ATTRIBUTES
volume: \DEV.$A.NB, "AAAO"
in: \DEV.$A.NB.INFILE1
out: \DEV.$S.#TEMPOUT
jobid-zero: Off
user: 205,70
60} SET JOB VOLUME $DATA6.NBFILES, "NU-U", IN
$DATA3.NBPBAT.OMSBATCH, OUT $S.#BPROC, EXECUTOR-PROGRAM
NBEXEC,
STARTUP "B-1, LI 2:00, XPRI 110"; SHOW JOB
JOB ATTRIBUTES
volume: \DEV.$DATA6.NBFILES, "NU-U"
in: \DEV.$DATA3.NBPBAT.OMSBATCH
out: \DEV.$S.#BPROC
executor-program: \DEV.$A.T9190MAN.NBEXEC
startup: B-1, LI 2:00, XPRI 110
jobid-zero: Off
user: 205,70
61} RESET JOB; SHOW JOB
JOB ATTRIBUTES
volume: \DEV.$A.NB, "AAAO"
jobid-zero: Off
user: 205,70
The SET JOB command in the next example specifies the
MAXPRINTPAGES and OUT attributes. MAXPRINTPAGES applies to
the job submitted later in the example; OUT is overridden by
the SUBMIT JOB command. The job's other attributes are
defaults from the scheduler, except for IN and SELPRI, which
are specified by SUBMIT JOB.
2} SHOW JOB
JOB ATTRIBUTES
volume: \DEV.$A.NB, "AAAA"
jobid-zero: Off
user: 133,2
3} SET JOB MAXPRINTPAGES 100, OUT $S.#MYOUT; SHOW JOB
JOB ATTRIBUTES
volume: \DEV.$A.NB, "AAAA"
out: \DEV.$S.#MYOUT
jobid-zero: Off
maxprintpages: 100
user: 133,2
4} SUBMIT JOB STATS, IN JOB1, OUT $S.#STATS, SELPRI 6; INFO
JOB
STATS; INFO SCHEDULER
Job STATS Jobnumber 23 submitted
JOB ATTRIBUTES for STATS
jobnumber: 23
volume: \DEV.$A.NB, "AAAA"
in: \DEV.$A.NB.JOB1
```

```
out: \DEV.$S.#STATS
executor-program: \DEV.$SYSTEM.SYSTEM.TACL
jobid-zero: Off
pri: 120
selpri: 6
maxprintlines: None
maxprintpages: 100
class: DEFAULT
stall: Off
stop-on-abend: Off
highpin: Off
submit: 06OCT94 09:42:31
alter: 06OCT94 09:42:31
user: 133,2
next-runtime: 06OCT94 09:42:32
SCHEDULER ATTRIBUTES .
default-executor-program: \DEV.$SYSTEM.SYSTEM.TACL .
default-pri: 120 .
default-selpri: 3
default-maxprintlines: None
default-maxprintpages: None .
default-out: \DEV.$S.#BATCH
default-class: DEFAULT
default-stall: Off
default-stop-on-abend: Off
default-highpin: Off .
```

- This example shows the effect of the SET JOB command's LIKE qualifier:

```
17} INFO JOB DAILY-BACKUP; SHOW JOB
JOB ATTRIBUTES for DAILY-BACKUP
jobnumber: 25
volume: \DEV.$A.NB, "GOGO"
in: \DEV.$A.NB.DLYBKP
out: \DEV.$S.#BACKUPS
executor-program: \DEV.$SYSTEM.SYSTEM.BACKUP
jobid-zero: Off
pri: 149
selpri: 5
maxprintlines: None
maxprintpages: None
class: DEFAULT
stall: Off
stop-on-abend: Off
every: 1 DAYS
after: 06OCT94 18:00:00
highpin: Off
submit: 06OCT94 10:09:06
alter: 06OCT94 10:09:06
user: 255,205
next-runtime: 06OCT94 18:00:00
JOB ATTRIBUTES
volume: \DEV.$A.NB, "AAAA"
jobid-zero: Off
user: 255,205
18} SET JOB LIKE DAILY-BACKUP, IN WKLYBKP, EVERY 7 DAYS; SHOW
```

```
JOB
JOB ATTRIBUTES
volume: \DEV.$A.NB, "GOGO"
in: \DEV.$A.NB.WKLYBKP
out: \DEV.$S.#BACKUPS
executor-program: \DEV.$SYSTEM.SYSTEM.BACKUP
jobid-zero: Off
pri: 149
selpri: 5
maxprintlines: None
maxprintpages: None
class: DEFAULT
stall: Off
stop-on-abend: Off
every: 7 DAYS
after: 06OCT94 18:00:00
highpin: Off
user: 255,205
```

SET SCHEDULER Command

Use the SET SCHEDULER command to specify scheduler attributes in BATCHCOM's working-attributes set. Schedulers added later by the ADD SCHEDULER command adopt these attributes (which override scheduler defaults) unless that command specifies otherwise.

```
SET [ SCHEDULER ] attribute [ , attribute ]…
```

attribute

is one of these scheduler attributes (see Section 7, Attributes):

```
AT-A[LLOWED] { OF[F] | ON) }
B[ACKUPCPU] { cpu-number-1 [ , cpu-number-2 ] | * }
CAT[CHUP] { OF[F] | ON) }
D[EFAULT]-C[LASS] class-name
D[EFAULT]-E[XECUTOR]-[PROGRAM] file-name
D[EFAULT]-H[IGHPIN] { OF[F] | ON) }
D[EFAULT]-MAXPRINTL[INES] { number | NON[E] }
D[EFAULT]-MAXPRINTP[AGES] { number | NON[E] }
D[EFAULT]-O[UT] file-name
D[EFAULT]-P[RI] number
D[EFAULT]-SE[LPRI] number
D[EFAULT]-ST[ALL] { OF[F] | ON }
D[EFAULT]-[STOP]-[ON]-[ABEND] { OF[F] | ON }
EM[S] { ER[RORS] | OF[F] | ON }
INI[TIATION] { OF[F] | ON }
LO[CALNAMES] [ \remote-node
( \remote-node [ , \remote-node ]… ) ]
M[AX]-[CONCURRENT]-[JOBS] max-concurrent-jobs
[ , max-temporary-executors ]
M[AX]-[PRI] number
S[UBMIT]-A[LLOWED] { OF[F] | ON }
TA[PEDRIVES] number
```

Considerations

- The SET SCHEDULER command is available to all users. However, you must be a NetBatch supervisor to use the ADD SCHEDULER command.
- To list scheduler attributes in the working-attributes set, use the SHOW SCHEDULER command.
- No default scheduler attributes are displayed in the working-attributes set. To remove attributes added to the set by the SET SCHEDULER command, use the RESET SCHEDULER command.
- You can omit the object keyword SCHEDULER from the SET SCHEDULER command only when SCHEDULER is the current assumed object. For more information, see ASSUME SCHEDULER Command on page 6-73.

Example

This example shows the SET SCHEDULER command specifying the AT-ALLOWED, DEFAULT-OUT, and DEFAULT-PRI attributes. These attributes apply to the scheduler added later, except DEFAULT-PRI, which the ADD SCHEDULER command overrides. The scheduler's other attributes are defaults, aside from BACKUPCPU and DEFAULT-EXECUTOR-PROGRAM whose values the ADD SCHEDULER command specifies.

```
11> NETBATCH /NAME $SCHD, NOWAIT/ $DATA7.SCHD !
12> BATCHCOM $SCHD
1} SET SCHEDULER AT-ALLOWED ON, DEFAULT-OUT $S.#SCHD,
 DEFAULT-PRI
149
2} ADD SCHEDULER, BACKUPCPU 2,3, DEFAULT-EXECUTOR-PROGRAM
FUP,
DEFAULT-PRI 157; INFO SCHEDULER
Scheduler added
SCHEDULER ATTRIBUTES
backupcpu: 2,3
at-allowed: On
submit-allowed: On
initiation: On
default-executor-program: \MELBDEV.$SYSTEM.SYSTEM.FUP
max-pri: 199
default-pri: 157
max-concurrent-jobs: 500,500
default-selpri: 3
default-maxprintlines: None
default-maxprintpages: None
tapedrives: 2
default-out: \MELBDEV.$S.#SCHD
default-class: DEFAULT
default-stall: Off
default-stop-on-abend: Off
default-highpin: Off
catchup: On
ems: Off
```

SHOW ATTACHMENT-SET Command

Use the SHOW ATTACHMENT-SET command to list attachment-set attributes from BATCHCOM's working-attributes set.

```
SHO[W] [ ATTACHMENT-SET ] [ / OU[T] [ file-name ] / ]
[ attribute [ , attribute ]… ] [ , O[BEY]-[FORM] ]
[ O[BEY]-[FORM] ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

attribute

is one of these attachment-set attributes:

```
ASSI[GN] [ ASSIGN-name ]
DEFI[NE] [ DEFINE-name ]
PA[RAM] [ PARAM-name ]
SEC[URITY]
TEM[PORARY]
```

attribute causes BATCHCOM to list only the specified attribute. If you omit *attribute*, BATCHCOM lists all attributes. If you specify ASSIGN, DEFINE, or PARAM, but omit *ASSIGN-name*, *DEFINE-name*, or *PARAM-name*, BATCHCOM lists all ASSIGNs, DEFINEs, or PARAMs.

You can include the asterisk (*) and question mark (?) wild-card characters in *ASSIGN-name*, *DEFINE-name*, or *PARAM-name* to specify a range of ASSIGNs, DEFINEs, or PARAMs.

DEFINE-name must include the equals sign (=) prefix even when you specify *DEFINE-name* with wild-card characters; for example, … DEFINE =OUT*, not … DEFINE OUT*.

OBEY-FORM

causes BATCHCOM to list attributes in SET ATTACHMENT-SET command format. BATCHCOM prefixes the SET commands with the ASSUME ATTACHMENT-SET and RESET ATTACHMENT-SET commands.

Considerations

- The SHOW ATTACHMENT-SET command is available to all users.
- To specify the values of attachment-set attributes in the working-attributes, use the SET ATTACHMENT-SET command. To remove the attributes, use the RESET ATTACHMENT-SET command.

- The command SHOW ATTACHMENT-SET DEFINE =? displays the last DEFINE added to, altered in, or listed from the working-attributes set; for example:

```
5} SET ATTACHMENT-SET (DEFINE=BACKUP, CLASS TAPE, DEVICE
$TAPE1, EXPIRATION 22OCT1994, OWNER "205,70", USE OPENFLAG,
DENSITY 6250, LABELS ANSI)
6} SHOW ATTACHMENT-SET DEFINE =?
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =_Work set, CLASS TAPE, LABELS
ANSI, OWNER "205,70", EXPIRATION
"22 OCT 1994", DENSITY 6250, USE
OPENFLAG, DEVICE \MELBDEV.$TAPE1
```

  BATCHCOM identifies the DEFINE as “=_Work set” instead of by name.

- The OBEY-FORM qualifier is available for all SHOW commands. It enables the creation of BATCHCOM command files that you can use to set up a working-attributes set.
- ADP (an abbreviation of ASSIGNs, DEFINEs, and PARAMs) is an alias of the object keyword ATTACHMENT-SET.
- You can omit the object keyword ATTACHMENT-SET from the SHOW ATTACHMENT-SET command only when ATTACHMENT-SET is the current assumed object. For more information, see ASSUME ATTACHMENT-SET Command on page 6-69.

Examples

- The SHOW ATTACHMENT-SET command in this example lists attachment-set attributes in a sample working-attributes set. The set inherits all attributes from the TACL working set, except DEFINEs whose names begin with =_ZBAT. (The =_ZBAT DEFINEs in the example specify the scheduler’s help text file and log file.)

```
> ASSIGN FILE-A, $BIG2.FILES.A
> ASSIGN FILE-B, $BIG2.FILES.B
> ASSIGN FILE-C, $BIG2.FILES.C
> ALTER DEFINE =_DEFAULTS, VOLUME $A.NB
> ADD DEFINE =LP, CLASS SPOOL, COPIES 1, LOC $S
> ADD DEFINE =_ZBAT_IMMU_FILE, CLASS MAP, FILE $A.NB.HELP
> ADD DEFINE =_ZBAT_LOG_FILE, CLASS MAP, FILE $A.NB.DAYLOG
> PARAM 1 ABC
> PARAM 2 DEF
> PARAM 3 GHI
> BATCHCOM
1} SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =LP, CLASS SPOOL, LOC $S, COPIES 00001
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
ASSIGN FILE-A, $BIG2.FILES.A
ASSIGN FILE-B, $BIG2.FILES.B
ASSIGN FILE-C, $BIG2.FILES.C
PARAM 1 ABC
```

```
PARAM 2 DEF
PARAM 3 GHI
```

- This example shows the SHOW ATTACHMENT-SET command listing the attributes of a specified ASSIGN and PARAM from the working-attributes set:

```
> BATCHCOM; SHOW ATTACHMENT-SET ASSIGN FILE-C, PARAM 3
ATTACHMENT-SET ATTRIBUTES
attachments: ASSIGN FILE-C, $BIG2.FILES.C
PARAM 3 GHI
```

- This example shows a SHOW ATTACHMENT-SET command writing the ASSIGN attachment-set attributes from the working-attributes set to an EDIT file:

```
> BATCHCOM; SHOW ATTACHMENT-SET /OUT ASSIGNS/ ASSIGN
> FUP COPY ASSIGNS
ATTACHMENT-SET ATTRIBUTES
attachments: ASSIGN FILE-A, $BIG2.FILES.A
ASSIGN FILE-B, $BIG2.FILES.B
ASSIGN FILE-C, $BIG2.FILES.C
```

- This example shows the result of a SHOW ATTACHMENT-SET command that includes the OBEY-FORM qualifier. In the example, BATCHCOM sends the command's output to an EDIT file. Note the use of the keyword alias ADP.

```
> BATCHCOM; SHOW ADP /OUT A1/ DEFINE, PARAM, OBEY-FORM
> FUP COPY A1
ASSUME ATTACHMENT-SET
RESET
SET (DEFINE =LASER,CLASS SPOOL,LOC \MELBQAT.$S,COPIES 00001)
SET (DEFINE =_DEFAULTS,CLASS DEFAULTS,VOLUME $A.NB)
SET (PARAM 1 ABC)
SET (PARAM 2 DEF)
SET (PARAM 3 GHI)
```

- This example shows a SHOW ATTACHMENT-SET command listing the SECURITY and TEMPORARY attributes specified by a SET ATTACHMENT-SET command:

```
1} SET ATTACHMENT-SET SECURITY "AOGO", TEMPORARY ON
2} SHOW ATTACHMENT-SET SECURITY, TEMPORARY
ATTACHMENT-SET ATTRIBUTES
security: "AOGO"
temporary: On
```

- This example shows the effect of the question mark wild-card character in a SHOW ATTACHMENT-SET command. Note the use of the keyword alias ADP.

```
49} SET ADP (PARAM AX 1), (PARAM BY 2), (PARAM CZ 3)
50} SET ADP (PARAM DX 1), (PARAM EY 2), (PARAM FZ 3)
51} SET ADP (PARAM GX 1), (PARAM HY 2), (PARAM IZ 3)
52} SHOW ADP PARAM ?Y
ATTACHMENT-SET ATTRIBUTES
attachments: PARAM BY 2
PARAM EY 2
PARAM HY 2
```

SHOW CLASS Command

Use the SHOW CLASS command to display the INITIATION class attribute from BATCHCOM's working-attributes set.

```
SHO[W] [ CLASS ] [ / OU[T] [file-name ] / ] [ O[BEY]-[FORM] ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

OBEY-FORM

causes BATCHCOM to display the INITIATION class attribute from the working-attributes set in SET CLASS command format.

Considerations

- The SHOW CLASS command is available to all users.
- You can specify the value of the class attribute in the working-attributes set by using the SET CLASS command.
- The OBEY-FORM qualifier is available for all SHOW commands. It enables the creation of BATCHCOM command files that you can use to set up a working-attributes set.
- JOBCLASS is an alias of the object keyword CLASS.
- You can omit the object keyword CLASS from the SHOW CLASS command only when CLASS is the current assumed object. For more information, see ASSUME CLASS Command on page 6-70.

Examples

- This example shows a SHOW CLASS command displaying the class attribute from the working-attributes set:

```
1} SHOW CLASS
CLASS ATTRIBUTE
initiation: On
```

- This example shows the result of a SHOW CLASS command that writes class attribute details to an EDIT file:

```
22> BATCHCOM; SHOW CLASS /OUT SHOWCL1/
23> FUP COPY SHOWCL1
CLASS ATTRIBUTE
initiation: On
```

- This example shows the result of a SHOW CLASS command that includes the OBEY-FORM qualifier. In the example, BATCHCOM sends the command's output to an EDIT file.

```
26> BATCHCOM; SHOW CLASS /OUT SHOWCL2/ OBEY-FORM
27> FUP COPY SHOWCL2
SET CLASS INITIATION ON
```

SHOW EXECUTOR Command

Use the SHOW EXECUTOR command to list executor attributes from BATCHCOM's working-attributes set.

```
SHO[W] [ EXECUTOR ] [ / OU[T] [ file-name ] / ]
[ attribute [ , attribute ] [ , O[BEY]-[FORM] ]
[ O[BEY]-[FORM] ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

attribute

is one of these executor attributes:

```
CL[ASS]
CP[U]
```

attribute causes BATCHCOM to list only the specified attribute. If you omit attribute, BATCHCOM lists all attributes.

OBEY-FORM

causes BATCHCOM to list attributes in SET EXECUTOR command format. BATCHCOM prefixes the SET commands with the ASSUME EXECUTOR and RESET EXECUTOR commands.

Considerations

- The SHOW EXECUTOR command is available to all users.
- To specify the values of executor attributes in the working-attributes set, use the SET EXECUTOR command. To remove the attributes, use the RESET EXECUTOR command.
- The OBEY-FORM qualifier is available for all SHOW commands. It enables the creation of BATCHCOM command files that you can use to set up a working-attributes set.

- You can omit the object keyword EXECUTOR from the SHOW EXECUTOR command only when EXECUTOR is the current assumed object. For more information, see ASSUME EXECUTOR Command on page 6-71.

Examples

- This example shows a SHOW EXECUTOR command listing executor attributes from the working-attributes set:

```
25} SHOW EXECUTOR
EXECUTOR ATTRIBUTES
cpu: 3
classes: BANKING
INSURANCE
```

- This example shows a SHOW EXECUTOR command writing the CLASS executor attribute from the working-attributes set to an EDIT file:

```
8} SHOW EXECUTOR /OUT ZBATEXEC/ CLASS
9} RUN FUP COPY ZBATEXEC
EXECUTOR ATTRIBUTES
classes: BANKING
INSURANCE
```

- This example shows the result of a SHOW EXECUTOR command that includes the OBEY-FORM qualifier. In the example, BATCHCOM sends the command's output to an EDIT file.

```
13} SHOW EXECUTOR /OUT SETEXECS/ OBEY-FORM
14} RUN FUP COPY SETEXECS
ASSUME EXECUTOR
RESET;SET CPU 3, CLASS(BANKING, INSURANCE)
```

SHOW JOB Command

Use the SHOW JOB command to list job attributes from BATCHCOM's working-attributes set. The command also displays the ID of the current user.

```
SHO[W] [ JOB ] [ / OU[T] [ file-name ] / ]
[ attribute [ , attribute ]… [ , O[BEY]-[FORM] ]
[ O[BEY]-[FORM] ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

attribute

is one of these job attributes:

AF[TER}	IN	REST[ART]
AT	J[OB]-L[OG]	RUND
A[TTACHMENT]-S[ET]	J[OBID]-Z[ERO]	SA[VEABEND]
CA[LENDAR]	LIB	SEL[PRI]
CL[ASS]	LIM[IT]	STAL[L]
DES[CRIPTION]	MAXPRINTL[INES]	STARTU[P]
EV[ERY]	MAXPRINTP[AGES]	S[TOP]-[ON]-[ABEND]
E[XECUTOR]-P[ROGRAM]	ME[M]	SWA[P]
EXT[SWAP]	NA[ME]	TA[PEDRIVES]
HIG[HPIN]	OU[T]	V[OLUME]
HOLD	PF[S]	WAIT
HOLDA[FTER]	PR[I]	WAITO[N]
IF[FAILS]	P[URGE]-[IN]-[FILE]	

attribute causes BATCHCOM to list only the specified attribute. If you omit *attribute*, BATCHCOM lists all attributes.

OBEY-FORM

causes BATCHCOM to list attributes in SET JOB command format. BATCHCOM prefixes the SET commands with the ASSUME JOB and RESET JOB commands, and suffixes them with a commented CHANGEUSER command.

The CHANGEUSER command (for example, ==CHANGEUSER 255,205) specifies the user ID of the current user, but not the user's password. For the command to work, you must edit it to remove the double equal signs and add the user's password.

OBEY-FORM lists the SET JOB ATTACHMENT-SET command in one of two forms: uncommented or commented. The command appears in uncommented form if it specifies a named attachment set. For example:

```
SET JOB ATTACHMENT-SET ((SUPER.FPP)COMPILES)
```

The command appears in commented form if it specifies a numbered attachment set. For example:

```
==SET JOB ATTACHMENT-SET ((NB.USER)50)
```

Considerations

- The SHOW JOB command is available to all users.
- To specify the values of job attributes in the working-attributes set, use the SET JOB command. To remove the attributes, use the RESET JOB command.

- The OBEY-FORM qualifier is available for all SHOW commands. It enables the creation of BATCHCOM command files that you can use to set up a working-attributes set.
- The SHOW JOB command displays the current user's ID in *group-ID*,*user-ID* form. For example:

```
27} SHOW JOB
JOB ATTRIBUTES
volume: \MELBDEV.$A.NB, "NCNU"
jobid-zero: Off
user: 255,255
```

  User is not a job attribute though BATCHCOM displays it as if it were.

- You can omit the object keyword JOB from the SHOW JOB command only when JOB is the current assumed object. For more information, see ASSUME JOB Command on page 6-72.

Examples

- The SHOW JOB command in this example displays the state of the working-attributes set at the start of a session:

```
23> WHO .
Current volume: $SYSTEM.NETBATCH .
Userid: 255,205 Username: SUPER.FPP Security: "NNNC"
24> BATCHCOM
1} SHOW JOB
JOB ATTRIBUTES
volume: \QA.$SYSTEM.NETBATCH, "NNNC"
jobid-zero: Off
user: 255,205
```

- This example shows a SHOW JOB command listing job attributes from the working-attributes set during a session:

```
4} SHOW JOB
JOB ATTRIBUTES
volume: \MELBDEV.$DATA7.NB, "AAAA"
executor-program: \MELBDEV.$SYSTEM.SYSTEM.FUP
jobid-zero: Off
purge-in-file: On
attachment-set: #CURRENT
user: 133,2
```

- This example shows a SHOW JOB command writing specified job attributes from the working-attributes set to an EDIT file:

```
8} SHOW JOB /OUT JOBDTLS/ EXECUTOR-PROGRAM, MAXPRINTPAGES,
CLASS, ATTACHMENT-SET
9} RUN FUP COPY JOBDTLS
JOB ATTRIBUTES
executor-program: \DEV.$SYSTEM.SYSTEM.ENFORM
maxprintpages: 100
```

```
class: ACCOUNTS-RPRTS
attachment-set: LASER-PRINTER
```

- This example shows the result of a SHOW JOB command that includes the OBEY-FORM qualifier. In the example, BATCHCOM sends command output to an EDIT file. Note the commented CHANGEUSER command after the last SET command.

```
12} SHOW JOB
JOB ATTRIBUTES
volume: \DEV.$QAT2.HPLJ, "NNNN"
out: \DEV.$S.#LP2
executor-program: \DEV.$SYSTEM.SYSTEM.COBOL85
jobid-zero: Off
pri: 119
attachment-set: (SUPER.FPP)COMPILES
user: 133,2
13} SHOW JOB /OUT JOBATTR/ OBEY-FORM
14} RUN FUP COPY JOBATTR
ASSUME JOB
RESET
SET VOLUME \DEV.$QAT2.HPLJ, "NNNN"
SET OUT \DEV.$S.#LP2
SET EXECUTOR-PROGRAM \DEV.$SYSTEM.SYSTEM.COBOL85
SET JOBID-ZERO OFF
SET PRI 119
SET JOB ATTACHMENT-SET ((SUPER.FPP)COMPILES)
==CHANGEUSER 133,2
```

SHOW SCHEDULER Command

Use the SHOW SCHEDULER command to list scheduler attributes from BATCHCOM's working-attributes set.

```
SHO[W] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
[ attribute [ , attribute ]… [ , O[BEY]-[FORM] ]
[ O[BEY]-[FORM] ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

attribute

is one of these scheduler attributes:

AT-A[LLOWED]	D[EFAULT]-MAXPRINTP[AGES]	INI[TIATION]
B[ACKUPCPU]	D[EFAULT]-O[UT]	LO[CALNAMES]
CAT[CHUP]	D[EFAULT]-P[RI]	M[AX]-[CONCURRENT]-[JOBS]

```
D[EFAULT]-C[LASS]   D[EFAULT]-SE[LPRI]    M[AX]-[PRI]
D[EFAULT]-          D[EFAULT]-ST[ALL]     S[UBMIT]-A[LLOWED]
E[XECUTOR]-
[PROGRAM]
D[EFAULT]-          D[EFAULT]-[STOP]-     TA[PEDRIVES]
H[IGHPIN]           [ON]-[ABEND]
D[EFAULT]-          EM[S]
MAXPRINTL[INES]
```

attribute causes BATCHCOM to list only the specified attribute. If you omit *attribute*, BATCHCOM lists all attributes.

OBEY-FORM

causes BATCHCOM to list attributes in SET SCHEDULER command format. BATCHCOM prefixes the SET commands with the ASSUME SCHEDULER and RESET SCHEDULER commands.

Considerations

- The SHOW SCHEDULER command is available to all users.
- To specify the values of scheduler attributes in the working-attributes set, use the SET SCHEDULER command. To remove the attributes, use the RESET SCHEDULER command.
- The OBEY-FORM qualifier is available for all SHOW commands. It enables the creation of BATCHCOM command files for use when setting up a working-attributes set.
- You can omit the object keyword SCHEDULER from the SHOW SCHEDULER command only when SCHEDULER is the current assumed object. For more information, see ASSUME SCHEDULER Command on page 6-73.

Examples

- This example shows a SHOW SCHEDULER command listing scheduler attributes from the working-attributes set:

```
20} SHOW SCHEDULER
SCHEDULER ATTRIBUTES
21} SET SCHEDULER AT-ALLOWED OFF, SUBMIT-ALLOWED ON, DEFAULT
EXECUTOR-
PROGRAM $SYSTEM.SYSTEM.TACL, DEFAULT-CLASS DEFAULT,
DEFAULT-OUT $S.#SCHD, DEFAULT-PRI 149, DEFAULT-SELPRI 4,
DEFAULT-STOP-ON-ABEND ON, TAPEDRIVES 1
22} SHOW SCHEDULER
SCHEDULER ATTRIBUTES
at-allowed: Off
submit-allowed: On
default-executor-program: \MELBDEV.$SYSTEM.SYSTEM.TACL
```

```
default-pri: 149
default-selpri: 4
tapedrives: 1
default-out: \MELBDEV.$S.#SCHD
default-class: DEFAULT
default-stop-on-abend: On
```

- This example shows a SHOW SCHEDULER command writing specified scheduler attributes from the working-attributes set to an EDIT file:

```
24} SHOW SCHEDULER /OUT COBCOMP/ DEFAULT-EXECUTOR-PROGRAM,
DEFAULT-OUT, DEFAULT-PRI
25} RUN FUP COPY COBCOMP
SCHEDULER ATTRIBUTES
default-executor-program: \DEV.$SYSTEM.SYSTEM.COBOL85
default-pri: 110
default-out: \DEV.$S.#COMPILE
```

- This example shows the result of a SHOW SCHEDULER command that includes the OBEY-FORM qualifier. In the example, BATCHCOM sends the command's output to an EDIT file.

```
27} SET SCHEDULER AT-ALLOWED OFF, DEFAULT-CLASS STANDARD-
CLASS,
DEFAULT-EXECUTOR-PROGRAM $SYSTEM.SYSTEM.BATCHCOM, DEFAULT-OUT
$S.#NBJOBS
28} SHOW SCHEDULER /OUT SCHDATTR/ OBEY-FORM
29} RUN FUP COPY SCHDATTR
ASSUME SCHEDULER
RESET
SET AT-ALLOWED OFF
SET DEFAULT-EXECUTOR-PROGRAM \MELBDEV.$SYSTEM.SYSTEM.BATCHCOM
SET DEFAULT-OUT \MELBDEV.$S.#NBJOBS
SET DEFAULT-CLASS STANDARD-CLASS
```

SHUTDOWN SCHEDULER Command

Use the SHUTDOWN SCHEDULER command to shut down a scheduler. The command allows executing and over-limit jobs to finish before shutdown, but it immediately stops suspended jobs and their processes.

The scheduler deletes from its database jobs whose processes it stops. The only jobs not deleted are recurrent jobs (jobs with the CALENDAR or EVERY attribute) and jobs with the HOLDAFTER ON attribute.

```
SHU[TDOWN] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

Considerations

- The SHUTDOWN SCHEDULER command is available to NetBatch supervisors only.
- SHUTDOWN SCHEDULER is one of two commands you can use to shut down a scheduler. The other command, ABORT SCHEDULER, does not allow executing and over-limit jobs to finish before shutdown. Like with SHUTDOWN SCHEDULER, it stops suspended jobs immediately. For more information, see ABORT SCHEDULER Command on page 6-30.
- No scheduler or job control commands are effective after the SHUTDOWN SCHEDULER command shuts down the scheduler.
- Shutting down a scheduler results in the closure of its database files and log file. The scheduler keeps details of its configuration at shutdown in its BATCHCTL file for use during a warm start.
- To restart a shutdown scheduler, do either of:
  - Warm start the scheduler:
    1. Run the scheduler program NETBATCH.
    2. Start the scheduler by using the START SCHEDULER command.
  - Cold start the scheduler:
    1. Run the scheduler program NETBATCH.
    2. Initialize the scheduler database with the ADD SCHEDULER command.
    3. Start the scheduler by using the START SCHEDULER command.

  For more information about scheduler cold-start and warm-start procedures, see Section 3, Scheduler Planning, Configuration, and Management.
- You can omit the object keyword SCHEDULER from the SHUTDOWN SCHEDULER command only when SCHEDULER is the current assumed object. For more information, see ASSUME SCHEDULER Command on page 6-73.

Examples

- This example shows the SHUTDOWN SCHEDULER command shutting down a scheduler after stopping a suspended job. The scheduler allows an executing job to finish before shutdown. The example also shows the results of a job command entered after shutdown.

```
96} STATUS JOB *
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ----------- -------------
35 ZBAT-0035 205,70 644 EXECUTING DEFAULT
36 ZBAT-0036 205,70 645 SUSPENDED DEFAULT
37 ZBAT-0037 205,70 READY DEFAULT
```

```
38 ZBAT-0038 205,70 14:45:00 DEFAULT
97} SHUTDOWN SCHEDULER
Scheduler shutting down
98} STATUS JOB *
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- --------------
35 ZBAT-0035 205,70 644 EXECUTING DEFAULT
37 ZBAT-0037 205,70 READY DEFAULT
38 ZBAT-0038 205,70 14:45:00 DEFAULT
16} STATUS JOB *
0201-E Process gone or path down; operation failed, \QA.$ZBAT
0014-E Device does not exist, \QA.$ZBAT
```

- This example lists SHUTDOWN SCHEDULER events recorded in the log file of the scheduler shut down in the previous example:

```
SHUTDN SHUTDOWN SCHEDULER \QA.$:0:61:17098028 U_255,255
H_\QA.$ZTN0.#PTY4
ABORT JOB ZBAT-0036 by NetBatch J_36 \QA.$Z618:17375533
U_255,255 H_\QA.$ZTN0.#PTY4
DOSTOP \QA.$Z618:17375533 J_36 U_255,255 H_\QA.$ZTN0.#PTY4
STOP CC_6 (ABEND) by \QA.$ZBAT:17357869 U_205,70
EXECUTOR-PROGRAM J_36 \QA.$Z618:17375533
UPDATE EXECUTOR E2 S_ON
FINISH JOB ZBAT-0036 T_0:0:0:8 J_36 P_DELAY
DELETE JOB ZBAT-0036 J_36
STOP CC_0 EXECUTOR-PROGRAM J_35 \QA.$Z617:17375277
UPDATE EXECUTOR E1 S_ON
FINISH JOB ZBAT-0035 T_0:0:0:12 J_35 P_DELAY
DELETE JOB ZBAT-0035 J_35
SHUTDN SHUTDOWN SCHEDULER specified and all jobs completed
DOSTOP Myself
```

- This example shows the cold start of a scheduler shut down by the SHUTDOWN SCHEDULER command. The cold start purges jobs from the old scheduler.

```
22> PPD $SCHD
Name Primary Backup Ancestor
$SCHD 0,271 1,268 $C8
23> BATCHCOM $SCHD; STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
25 X 255,205 1:Hold DEFAULT
26 Y 255,205 17:30:00 DEFAULT
27 Z 255,205 READY DEFAULT
24> BATCHCOM $SCHD; SHUTDOWN SCHEDULER
Scheduler shutting down
25> PPD $SCHD
(Process does not exist)
26> NETBATCH /NAME $SCHD, NOWAIT/ $DATA7.SCHD !
27> PPD $SCHD
Name Primary Backup Ancestor
$SCHD 0,276 $C8
28> BATCHCOM $SCHD; ADD SCHEDULER
Scheduler added
```

```
29> PPD $SCHD
Name Primary Backup Ancestor
$SCHD 0,276 $C8
30> BATCHCOM $SCHD; START SCHEDULER
Scheduler started
31> PPD $SCHD
Name Primary Backup Ancestor
$SCHD 0,276 2,275 $C8
32> BATCHCOM $SCHD; STATUS JOB *
2117-I No JOB selected
```

START EXECUTOR Command

Use the START EXECUTOR command to start executors whose state is OFF or STOP. The command makes the executors available for use by jobs and changes their states to ON.

```
START [ EXECUTOR ] [ / OU[T] [ file-name ] / ] executor-ID
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

executor-ID

is an executor name, or a range of executor names when specified with the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all executors.

Considerations

- The START EXECUTOR command is available to NetBatch supervisors only.
- An executor whose state is OFF is not available for use and can be:
  - An executor added by the ADD EXECUTOR command but not yet started
  - An executor stopped by the STOP EXECUTOR command
  - An executor whose state is STOP is in use by a job but was the subject of a STOP EXECUTOR command. When the job finishes, the scheduler stops the executor and change the executor’s state to OFF.
- You can omit the object keyword EXECUTOR from the START EXECUTOR command only when EXECUTOR is the current assumed object. For more information, see ASSUME EXECUTOR Command on page 6-71.

Examples

- The START EXECUTOR command in this example starts an executor named A0:

```
42} STATUS EXECUTOR A0
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
----------------------- --- ------ ---- -------------------
A0 0 OFF
43} START EXECUTOR A0
Executor A0 started
44} STATUS EXECUTOR A0
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
----------------------- --- ------ ---- -------------------
A0 0 ON
```

- The START EXECUTOR command in this example starts all OFF executors:

```
83} STATUS EXECUTOR *
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
----------------------- --- ------ ---- -------------------
RISC-0 0 OFF
RISC-1 1 OFF
RISC-2 2 OFF
RISC-3 3 OFF
84} START EXECUTOR *
Executor RISC-0 started
Executor RISC-1 started
Executor RISC-2 started
Executor RISC-3 started
85} STATUS EXECUTOR *
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
----------------------- --- ------ ---- -------------------
RISC-0 0 ON
RISC-1 1 ON
RISC-2 2 ON
RISC-3 3 ON
```

- This example lists the scheduler log file events recorded as a result of the previous START EXECUTOR * command:

```
START EXECUTOR RISC-0 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
UPDATE EXECUTOR RISC-0 S_ON U_255,255 H_\MELBDEV.$ZTN0.#PTY4
START EXECUTOR RISC-1 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
UPDATE EXECUTOR RISC-1 S_ON U_255,255 H_\MELBDEV.$ZTN0.#PTY4
START EXECUTOR RISC-2 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
UPDATE EXECUTOR RISC-2 S_ON U_255,255 H_\MELBDEV.$ZTN0.#PTY4
START EXECUTOR RISC-3 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
UPDATE EXECUTOR RISC-3 S_ON U_255,255 H_\MELBDEV.$ZTN0.#PTY4
```

START SCHEDULER Command

Use the START SCHEDULER command to make available for use a scheduler you are cold starting or warm starting. For cold starts, use the command after running the scheduler program NETBATCH and executing the ADD SCHEDULER command. For warm starts, use the command immediately after running the scheduler program.

```
START [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

Considerations

- The START SCHEDULER command is available to NetBatch supervisors only.
- The START SCHEDULER command starts a scheduler whose primary process only is running and whose database exists. The command creates the scheduler's backup process and opens its database files.
- To create a primary scheduler process, use the TACL RUN command to run the scheduler program NETBATCH.
- To create a scheduler database, use the ADD SCHEDULER command.

△ **Caution.** The ADD SCHEDULER command purges existing scheduler database files (except BATCHCTL and log files) before creating and initializing new files. Do not use the command unless you are sure loss of the existing files will not affect your organization. If in doubt, back up the files before using the command.

- The START SCHEDULER command uses information from files in the scheduler database to start the scheduler.
- If you cold started the scheduler, you must configure it before you can submit jobs for execution. Scheduler configuration involves changing default scheduler attributes (if necessary), adding executors and classes, and so on.
- If you warm started the scheduler, its configuration is the same as when the scheduler stopped, except for the log file. For more information, see Section 3, Scheduler Planning, Configuration, and Management.
- You can omit the object keyword SCHEDULER from the START SCHEDULER command only when SCHEDULER is the current assumed object. For more information, see ASSUME SCHEDULER Command on page 6-73.

Examples

- This example shows the cold start of a scheduler and the effect of the START SCHEDULER command in that process:

```
$DATA7.TEST 55> NETBATCH /NAME $TEST, NOWAIT/ $DATA7.TEST !
$DATA7.TEST 56> PPD $TEST
Name Primary Backup Ancestor
$TEST 0,60 $Z743
$DATA7.TEST 57> FILEINFO
Code EOF … Owner RWEP PExt SExt
BATCHCTL O 847 552 … 255,255 "OOOO" 2 2
LOGAAA O 847 528 … 255,255 "NCNU" 50 100
$DATA7.TEST 58> BATCHCOM $TEST; ADD SCHEDULER
Scheduler added
$DATA7.TEST 59> PPD $TEST
Name Primary Backup Ancestor
$TEST 0,60 $Z743
$DATA7.TEST 60> FILEINFO
Code EOF … Owner RWEP PExt SExt
ATTACH 847 0 … 255,255 "OOOO" 100 200
ATTACH0 847 0 … 255,255 "OOOO" 32 32
BATCHCTL O 847 552 … 255,255 "OOOO" 2 2
CHKQUE 847 0 … 255,255 "OOOO" 64 32
CHKQUE0 847 0 … 255,255 "OOOO" 64 32
EXECUTO0 847 0 … 255,255 "OOOO" 2 2
EXECUTOR 847 0 … 255,255 "OOOO" 6 6
JOB 847P 0 … 255,255 "OOOO" 60 60
JOBCLAS0 847 0 … 255,255 "OOOO" 2 2
JOBCLASS 847 0 … 255,255 "OOOO" 2 2
LOGAAA O 847 924 … 255,255 "NCNU" 50 100
NBATTX 847 0 … 255,255 "OOOO" 16 16
NBATTX0 847 0 … 255,255 "OOOO" 16 16
$DATA7.TEST 61> BATCHCOM $TEST; START SCHEDULER
Scheduler started
$DATA7.TEST 62> PPD $TEST
Name Primary Backup Ancestor
$TEST 0,60 1,41 $Z743
$DATA7.TEST 63> FILEINFO
Code EOF … Owner RWEP PExt SExt
ATTACH O 847 8192 … 255,255 "OOOO" 100 200
ATTACH0 O 847 0 … 255,255 "OOOO" 32 32
BATCHCTL O 847 552 … 255,255 "OOOO" 2 2
CHKQUE O 847 0 … 255,255 "OOOO" 64 32
CHKQUE0 O 847 0 … 255,255 "OOOO" 64 32
EXECUTO0 O 847 0 … 255,255 "OOOO" 2 2
EXECUTOR O 847 0 … 255,255 "OOOO" 6 6
JOB O 847P 0 … 255,255 "OOOO" 60 60
JOBCLAS0 O 847 0 … 255,255 "OOOO" 2 2
JOBCLASS O 847 0 … 255,255 "OOOO" 2 2
LOGAAA O 847 1584 … 255,255 "NCNU" 50 100
NBATTX O 847 0 … 255,255 "OOOO" 16 16
NBATTX0 O 847 0 … 255,255 "OOOO" 16 16
```

- The log-file events resulting from the cold start of the scheduler in the previous example are:

```
CREATE Scheduler Primary \MELBDEV.$TEST CPU 0
COLD Purge any existing scheduler database files
OPEN U_255,255 P_BATCHCOM \MELBDEV.$:0:61:17122860
ADD SCHEDULER \MELBDEV.$:0:61:17122860 U_255,255
H_\MELBDEV.$ZTN0.#PTY6
CLOSE \MELBDEV.$:0:61:17122860 U_255,255
OPEN U_255,255 P_BATCHCOM \MELBDEV.$:0:61:17123116
START SCHEDULER \MELBDEV.$:0:61:17123116 U_255,255
H_\MELBDEV.$ZTN0.#PTY6
CREATE Scheduler Backup \MELBDEV.$TEST CPU 1 U_255,255
H_\MELBDEV.$ZTN0.#PTY6
START \MELBDEV.$TEST:17660205
```

- This example shows the result of a START SCHEDULER command that tries to start a scheduler whose database does not exist:

```
> PPD $BTCH
(Process does not exist)
> FILES $DATA7.BTCH
No files match \MELBDEV.$DATA7.BTCH.*
> NETBATCH /NAME $BTCH, NOWAIT/ $DATA7.BTCH !
> FILES $DATA7.BTCH
$DATA7.BTCH
BATCHCTL LOGAAA
> BATCHCOM $BTCH; START SCHEDULER
2069-E Use ADD SCHEDULER to create and initialize the
database
```

- This example shows the result of a START SCHEDULER command that tries to act on a started scheduler:

```
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 0,61 3,29 $Z605
> BATCHCOM $ZBAT; START SCHEDULER
2055-E Scheduler is already started; command ignored
```

STATUS ATTACHMENT-SET Command

Use the STATUS ATTACHMENT-SET command to list attachment sets and the names and owners of jobs using those sets.

```
S[TATUS] [ ATTACHMENT-SET ] [ / OU[T] [ file-name ] / ]
{ [ ( user-ID ) ] attachment-set-ID | #CURRENT
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit file-name), BATCHCOM suppresses output.

user-ID

specifies a user ID or a range of user IDs specified with the asterisk (*) and question mark (?) wild-card characters. (*user-ID* must be in *group-name.user-name* or *group-ID,user-ID* form.) The default is the user ID of the current user.

attachment-set-ID

is one of:

attachment-set-name

specifies an attachment-set name or a range of names when specified with the asterisk (*) and question mark (?) wild-card characters.

attachment-set-number

is a scheduler-generated number identifying an attachment set created by means of the #CURRENT variable.

*

specifies all attachment sets.

#CURRENT

causes BATCHCOM to display the attachment set specified by the #CURRENT variable. If #CURRENT has a null value, BATCHCOM displays:

```
0345-E Undefined substitution
```

Considerations

- The STATUS ATTACHMENT-SET command is available to all users.

- ADP (an abbreviation of ASSIGNs, DEFINEs, and PARAMs) is an alias of the object keyword ATTACHMENT-SET.
- The STATUS ATTACHMENT-SET command displays attachment-set information in columnar format:

Column	Description
ATTACHMENT SET	Lists attachment-set names
IN USE BY JOBS	For each attachment set, lists the names and owners of jobs using the set. “None” appears if the set is not in use.

- You can omit the object keyword ATTACHMENT-SET from the STATUS ATTACHMENT-SET command only when ATTACHMENT-SET is the current assumed object. For more information, see ASSUME ATTACHMENT-SET Command on page 6-69.

Examples

- This example shows the STATUS ATTACHMENT-SET command listing all attachment sets:

```
46} STATUS ATTACHMENT-SET (*.*)*
ATTACHMENT SET IN USE BY JOBS
------------------------------ --------------------------
(NB.USER)11 (NB.USER)EARLY
(NB.USER)NBU-A (NB.MANAGER)GRAVEYARD
(NB.MANAGER)NBM-A None
(NB.MANAGER)NBM-B (SUPER.NB)LATE
(NB.MANAGER)BILLINGS
(NB.MANAGER)TIMESHEETS
(SUPER.NB)13 None
(SUPER.NB)SNB-A None
(SUPER.SUPER)SS-A None
(SUPER.SUPER)SS-B (SUPER.SUPER)PURGE-FILES
```

- This example shows the STATUS ATTACHMENT-SET command listing all attachment sets owned by super-group users (255, *n*). The attachment-set-ID of ? restricts the listing to sets having single-character names or numbers.

```
26} STATUS ATTACHMENT-SET (255,*)?
ATTACHMENT SET IN USE BY JOBS
------------------------------ --------------------------
(SUPER.FPP)2 (SUPER.SUPER)HOURS-WORKED
(SUPER.FPP)3 (SUPER.FPP)MANAGEMENT-RPRT
(SUPER.FPP)R None
(SUPER.SUPER)1 (SUPER.SUPER)SALARIES
(SUPER.SUPER)4 None
```

- The STATUS ATTACHMENT-SET command in this example lists attachment sets owned by NB.USER. The attachment-set-ID of M?2 restricts the listing to sets having three-character names beginning with M and ending with 2.

```
21} STATUS ATTACHMENT-SET (NB.USER)M?2
ATTACHMENT SET IN USE BY JOBS
-------------------------------- -------------------------
(NB.USER)MX2 None
(NB.USER)MY2 None
(NB.USER)MZ2 None
```

- This STATUS ATTACHMENT-SET command lists the current user's attachment sets:

```
53} CHANGEUSER SUPER.NB psswrd
53} STATUS ATTACHMENT-SET *
ATTACHMENT SET IN USE BY JOBS
-------------------------------- -------------------------
(SUPER.NB)1 (SUPER.NB)CLEANUP
(SUPER.NB)BUILDS None
(SUPER.NB)COMPARES (SUPER.NB)COMPARE-OLD-NEW
(SUPER.NB)COMPILES (SUPER.NB)COBOL-COMPILE
This example shows the STATUS ATTACHMENT-SET command
listing the
value of the #CURRENT variable at various stages during a
session. At the start of the
session, the variable has a null value. Its value is also
null at the end of the session.
1} STATUS ATTACHMENT-SET #CURRENT
-^-0345-
E Undefined substitution
2} ADD ATTACHMENT-SET UN
Attachment-set (NB.USER)UN added
3} STATUS ATTACHMENT-SET #CURRENT
ATTACHMENT SET IN USE BY JOBS
-------------------------------- -------------------------
(NB.USER)UN None
4} ADD ATTACHMENT-SET DEUX
Attachment-set (NB.USER)DEUX added
5} STATUS ATTACHMENT-SET #CURRENT
ATTACHMENT SET IN USE BY JOBS
-------------------------------- -------------------------
(NB.USER)DEUX None
6} ADD ATTACHMENT-SET #CURRENT
Attachment-set (NB.USER)3 added
7} STATUS ATTACHMENT-SET #CURRENT
ATTACHMENT SET IN USE BY JOBS
-------------------------------- -------------------------
(NB.USER)3 None
8} ALTER ATTACHMENT-SET UN, SECURITY "GOGO"
Attachment-set (NB.USER)UN altered
9} STATUS ATTACHMENT-SET #CURRENT
ATTACHMENT SET IN USE BY JOBS
-------------------------------- -------------------------
(NB.USER)UN None
```

```
10} RESET ATTACHMENT-SET; STATUS ATTACHMENT-SET #CURRENT
-^-0345-
E Undefined substitution
```

STATUS EXECUTOR Command

Use the STATUS EXECUTOR command to display executors' names, CPUs, and states. If an executor is in use by a job, the command also displays the job's number and its class name.

```
S[TATUS] [ EXECUTOR ] [ / OU[T] [ file-name ] / ] executor-ID
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

executor-ID

is an executor name, or a range of executor names when specified with the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all executors.

Considerations

- The STATUS EXECUTOR command is available to all users.
- The STATUS EXECUTOR command displays executor information in columnar format:

Column	Description
EXECUTOR	Lists executor names.
CPU	For each executor, lists the number of the executor's CPU.
STATE	Lists the state of each executor. For information on executor states, see Table 6-7 on page 6-164.
JOB	For each executor in use by a job, lists the number of the job.
CLASS	For each executor in use by a job, lists the name of the job's class.

- You can omit the object keyword EXECUTOR from the STATUS EXECUTOR command only when EXECUTOR is the current assumed object. For more information, see ASSUME EXECUTOR Command on page 6-71.

Table 6-7. Executor States

State	Description
ACTIVE	The executor is in use by a job.
DELETE	The executor is in use by a job. When the job finishes, the scheduler deletes the executor. An executor in the DELETE state can be: ● An executor that was the subject of a DELETE EXECUTOR command while in use by a job. ● A temporary executor. The scheduler creates temporary executors for all jobs with the AT attribute and all jobs operated on by the RUNNOW JOB command. A temporary executor has a scheduler-assigned name of the form __TEMP_EXEC_ *job-number*, where *job-number* is the number of the job using the executor; for example, __TEMP_EXEC_496. The scheduler selects the CPU of a temporary executor from available CPUs on the scheduler's node.
DOWN	The executor is not available for use because its CPU is down.
OFF	The executor is not available for use. An executor in the OFF state can be: ● An executor added by the ADD EXECUTOR command but not yet started ● An executor stopped by the STOP EXECUTOR command To start or restart an OFF executor, use the START EXECUTOR command.
ON	The executor is available for use.
STOP	The executor is in use by a job but was the subject of a STOP EXECUTOR command. When the job finishes, the scheduler stops the executor and changes the executor's state to OFF.

Example

This example shows the result of a STATUS EXECUTOR command:

```
31} STATUS EXECUTOR *
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
---------------------- --- ------ ---- --------------------
CLX-0 0 STOP 99 MANUFACTURING
CLX-0-SPARE1 0 OFF
CLX-0-SPARE2 0 DELETE 102 ACCOUNTS
CLX-1 1 ACTIVE 100 SALES
CLX-1-SPARE 1 ON
CLX-2 2 ACTIVE 101 DISTRIBUTION
CLX-2-SPARE 2 ON
CLX-3 3 DOWN
```

STATUS JOB Command

Use the STATUS JOB command to display jobs' numbers, names, owners, processing states, classes, SELPRI attributes, and log-file spooler-job numbers. The command also displays run statistics and information about executor-program processes.

```
S[TATUS] [ JOB] [ / OU[T] [ file-name ] / ]{ job-ID }
{ ( [ [ NOT] job-ID ,] ... [NOT] filter [ , [NOT] filter
]...)}
[SEL[PRI] ] [ DET[AIL] ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

job-ID

specifies a job name or number, or a range of job names. To specify a range of job names, use the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all jobs.

NOT

excludes jobs that satisfy the specified job-selection criterion.

```
STATUS JOB (WK94*, NOT WK9408*)
```

filter

specifies one of these job-selection criteria:

A[TTACHMENT]-S[ET]BBC[(AALVS1([(*user-ID*)] *attachment-set-ID*,#CURRENT))

selects jobs that are using the specified attachment set. For information about entering the attachment-set name, see ATTACHMENT-SET Job Attribute on page 7-19.

```
STATUS JOB (ATTACHMENT-SET (255,255)SUPER)
```

CL[ASS] *class-ID*

selects jobs belonging to the specified class or classes. *class-ID* specifies a class name or a range of class names. To specify a range of class names, use the asterisk (*) and question mark (?) wild-card characters.

IN *file-ID*

selects jobs that are using the specified input file or files. *file-ID* specifies an input-file name or a range of input-file names. To specify a range of input-file names, use the asterisk (*) and question mark (?) wild-card characters.

```
STATUS JOB (IN \DEV.$DATA7.*.IN*)
```

STATE *state*

selects jobs that are in the specified state. *state* is one of these job-state keywords: CALENDAR, EVENT, EXECUTING, HOLD, READY, RUNNEXT, RUNNOW, SPECIAL, SPECIAL-*n*, STALL, SUSPENDED, TAPE, TIME, and WAITON.

```
STATUS JOB (STATE SPECIAL, NOT STATE SPECIAL-2)
```

S[TATUS]-E[XECUTING] is an alias of STATUS JOB (STATE EXECUTING). S[TATUS]-R[EADY] is an alias of STATUS JOB (STATE READY).

U[SER] *user-ID*

selects jobs owned by the specified user or users. *user-ID* specifies a user ID or a range of user IDs in *group-name.user-name* or *group-ID,user-ID* form. To specify a range of user IDs, use the asterisk (*) and question mark (?) wild-card characters.

```
STATUS JOB (USER FPP.*, NOT USER 205,255)
```

S[TATUS]-U[SER] [*user-ID*] is an alias of STATUS JOB (U[SER] *user-ID*).

WAITO[N] *master-job*

selects dependent jobs whose WAITON attributes include the specified *master-job* name or names. *master-job* specifies a master-job name or a range of master-job names. To specify a range of master-job names, use the asterisk (*) and question mark (?) wild-card characters.

```
STATUS JOB (WAITON WK9408*, NOT WAITON WK940826)
```

BATCHCOM performs an OR operation when processing job-selection criteria of the same type and an AND operation when processing criteria of different types. For example, this command selects from class DEFAULT all TIME jobs whose names begin with A or B:

```
STATUS JOB (A*, B*, STATE TIME, CLASS DEFAULT)
```

SELPRI

causes BATCHCOM to display the jobs' SELPRI attributes instead of their log files' spooler-job numbers.

`DETAIL`

causes BATCHCOM to display jobs' run statistics and information about executor-program processes. For details, see Figure 6-5 on page 6-168 and Figure 6-6 on page 6-169.

Considerations

- The STATUS JOB command is available to all users.
- The STATUS JOB display varies, depending on whether the command includes the DETAIL qualifier and whether jobs are executing, over limit, or suspended.
- For STATUS JOB commands that do not include the DETAIL qualifier, the display appears in the format:

```
7} STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
2 TEST-JOB 255,255 EXECUTING DEFAULT
```

- The contents of each column in the display are:

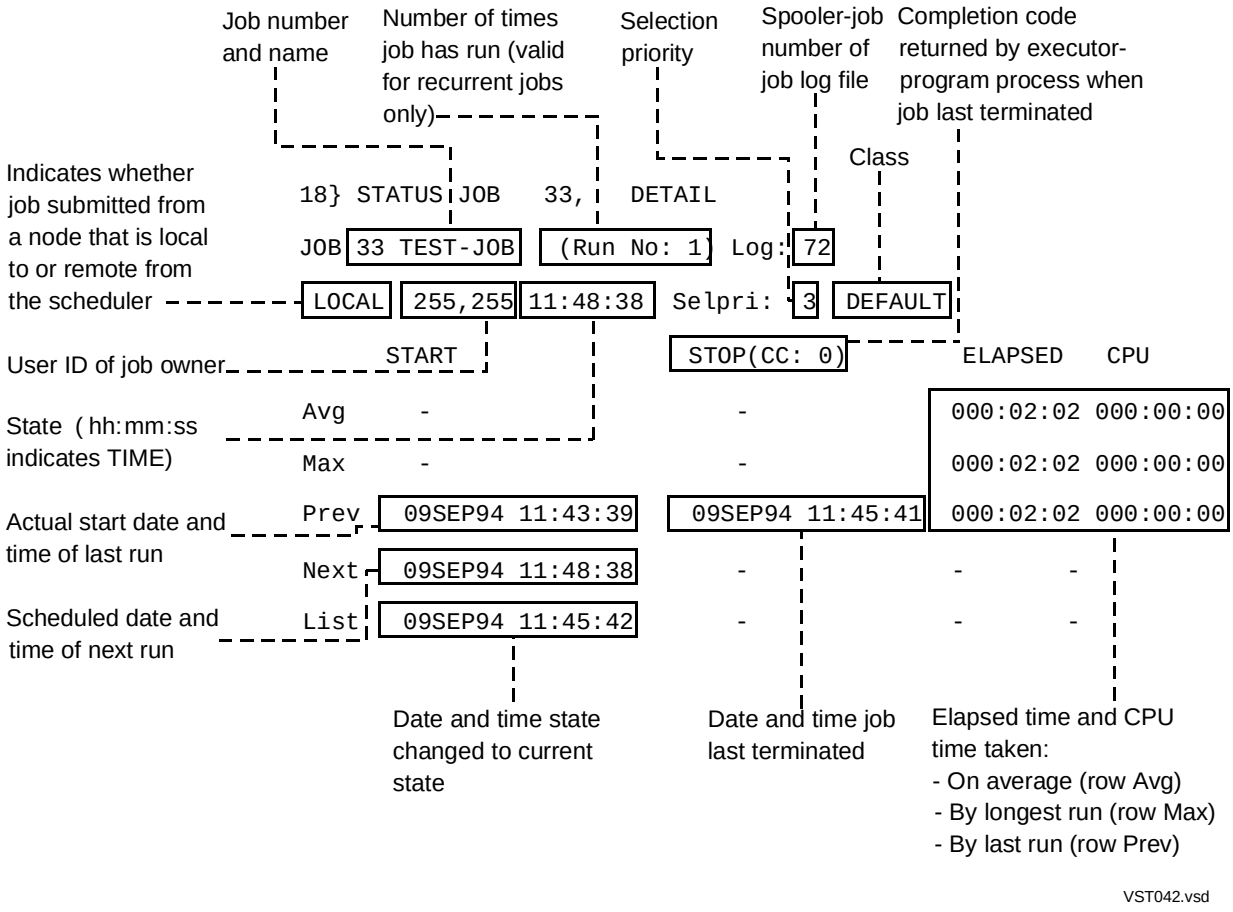
Column	Description
JOB	Lists the job number of each job.
JOBNAME	Lists the name of each job.
USERID	Lists the user ID of the owner of each job.
LOG or SEL	LOG is displayed when STATUS JOB command excludes SELPRI qualifier. Lists the spooler-job number of the log file of each job whose OUT attribute specifies a spooler location. Appears only for executing, over-limit, and suspended jobs and for jobs that have run in the past but whose state is now SPECIAL- *n* or TIME. SEL is displayed when STATUS JOB command includes SELPRI qualifier. Lists the SELPRI attribute of each job.
STATE	Lists the processing state of each job. For information on job states, see Table 6-8 on page 6-169.
CLASSNAME	Lists the name of the class of each job.

- For STATUS JOB commands that include the DETAIL qualifier, the display appears in this format for each job that is not executing, over limit, or suspended. Figure 6-5 on page 6-168 gives an annotated version of the display.

```
18} STATUS JOB 33, DETAIL
JOB 33 TEST-JOB (Run No: 1) Log: 72
LOCAL 255,255 11:48:38 Selpri: 3 DEFAULT
START STOP(CC: 0) ELAPSED CPU
Avg - - 000:02:02
000:00:00
Max - - 000:02:02
000:00:00
```

```
Prev 09SEP94 11:43:39 09SEP94 11:45:41 000:02:02
000:00:00
Next 09SEP94 11:48:38 - - -
List 09SEP94 11:45:42 - - -
```

Figure 6-5. Status JOB..., DETAIL Command Display for Jobs That Are Not Executing, Over Limit, or Suspended



- For STATUS JOB commands that include the DETAIL qualifier, the display appears in this format for each executing, over-limit, and suspended job. Figure 6-6 on page 6-169 gives an annotated version of the display.

```
17} STATUS JOB 33, DETAIL
JOB 33 TEST-JOB (Run No: 1) Log: 72
LOCAL 255,255 OVER LIMIT Selpri: 3 DEFAULT E1
START STOP ELAPSED CPU
Avg - - 000:00:00
000:00:00
Max - - 000:00:00
000:00:00
Curr 09SEP94 11:43:39 - 000:01:04
000:00:00
Program DLPFR Pid Cpu,Pin Pri CpuTime State
\MELRISK
```

```
DELAY DL F $Z315 1,46 120 0:0:0 FORCED LOW ,
RUNNABLE
```

Figure 6-6. JOB..., DETAIL C ommand- Display for Jobs That Are Not Executing, Over Limit, or Suspended

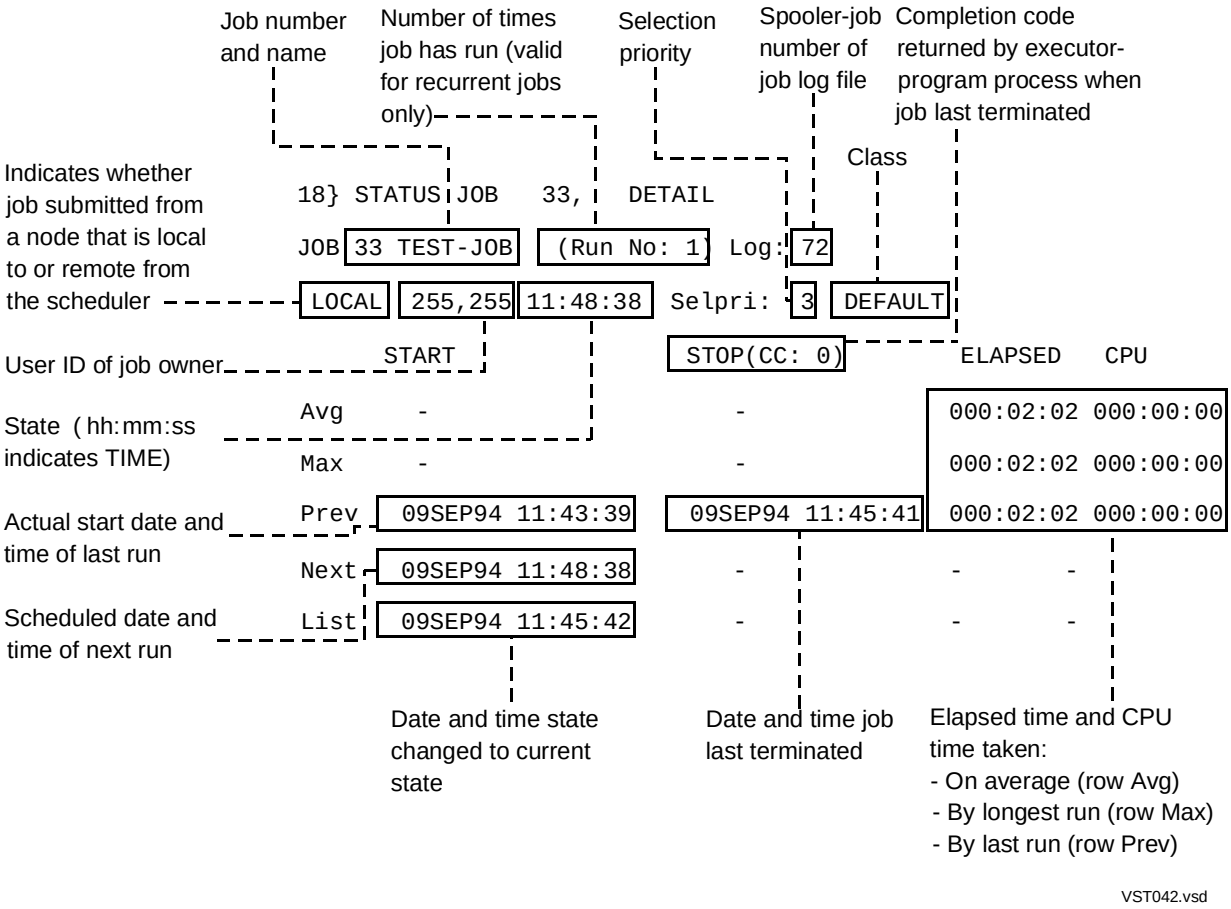


Table 6-8. Job States (page 1 of 4)

State	Description
EVENT	The job has the WAITON attribute and does not run until released. When released, the job's state changes to READY or TAPE, depending on its tape drive requirement. For more information on the WAITON attribute, see WAITON Job Attribute on page 7-119.
EXECUTING	The job is running.
OVER LIMIT	The job exceeded the execution time limit specified by the LIMIT attribute but is still running. For more information on the LIMIT attribute, see LIMIT Job Attribute on page 7-75.

Table 6-8. Job States (page 2 of 4)

State	Description
READY	The job is available for execution. The scheduler selects it when an executor associated with the job's class is available and no RUNNEXT or RUNNOW jobs are waiting. The job remains in the READY state if its class has the attribute INITIATION OFF or if the job was submitted when the scheduler has its attribute INITIATION OFF. This attribute prevents jobs belonging to the class from running. To make the jobs available for execution, change the value of the INITIATION attribute to ON by using the ALTER CLASS or ALTER SCHEDULER command.
RUNNEXT	The job runs as soon as an executor associated with its class is available. The scheduler selects the job ahead of READY jobs but after RUNNOW jobs. The job remains in the RUNNEXT state when an executor is available if either of these conditions exists: ● The job has the TAPEDRIVES attribute and requires more drives than are available. For more information about this attribute, see TAPEDRIVES Job Attribute on page 7-109 and TAPEDRIVES Scheduler Attribute on page 7-110. ● The job's class has the attribute INITIATION OFF. This attribute prevents jobs belonging to the class from running. To make the jobs available for execution, change the value of the INITIATION attribute to ON by using the ALTER CLASS command.
RUNNOW	The job was the subject of a RUNNOW JOB command or has the AT attribute. It is in this state momentarily until execution begins. The job remains in the RUNNOW state if its class or scheduler has the attribute INITIATION OFF. This attribute prevents jobs belonging to the class from running. To make the jobs available for execution, change the value of the INITIATION attribute to ON by using the ALTER CLASS or ALTER SCHEDULER command.
SPECIAL- *n*	The job is on hold for one of these reasons: ● The SUBMIT JOB command that submitted the job to the scheduler specified the HOLD ON attribute. ● An ALTER JOB command altered the value of the job's HOLD attribute from OFF to ON before the job ran. ● The scheduler put the job on hold after it ran and stopped because the job had the attribute HOLDAFTER ON. ● The scheduler put the job on hold during execution because of a problem requiring user intervention. *n* indicates the reason the job is on hold and has one of these values. To make any SPECIAL-*n* job available for execution, set the value of its HOLD attribute to OFF.

Table 6-8. Job States (page 3 of 4)

State	Description		
	n	**State Column**	**Description**
	1	1:Hold	The job is on hold for one of the first three reasons listed in the preceeding description of SPECIAL-*n*.
	2	2:NB failed	The job was running when an event other than the execution of an ABORT SCHEDULER or SHUTDOWN SCHEDULER command stopped its scheduler's processes. Usually, this state results from execution of a TACL STOP command that specifies the job's scheduler processes.
	3	3:NEWP.err	The scheduler tried to create a new process for the job's executor program, but failed. For information about the cause of the failure, see the scheduler log file or job log file.
	4	4:Start err	The scheduler successfully created a new process for the job's executor program, but the program failed during startup. For information on the cause of the failure, see the scheduler log file or job log file.
	5	5:Fail RsOn	The job has the RESTART ON attribute and abended. The job did not restart because its attributes include IFFAILS OFF. This state applies only to jobs with the CALENDAR or EVERY attribute.
	6	6:Fail RsOff	The job has the RESTART OFF attribute and abended. The job did not restart because its attributes include IFFAILSOFF. This state only applies to jobs with the CALENDAR or EVERY attribute.
	7	7:CAL error	An error occurred when the scheduler tried to open the job's BATCHCAL calendar file. For information on the cause of the error, see the scheduler log file or job log file.
	8	8:CAL exprd	The job's BATCHCAL calendar file has run out of dates.
	9	9:STALL	The job has the attribute STALL ON and failed while running. For more information on the STALL attribute, see STALL Job Attribute on page 7-102.

Table 6-8. Job States (page 4 of 4)

State	Description
SUSPENDED	The job was the subject of a SUSPEND JOB command. For more information, see SUSPEND JOB Command on page 6-190. To activate a suspended job, use the ACTIVATE JOB command.
TAPE	The job has the TAPEDRIVES attribute and requires more drives than are available. For more information, see TAPEDRIVES Job Attribute on page 7-109 and TAPEDRIVES Scheduler Attribute on page 7-110.
TIME (hh:mm:ss)	The job has one of these attributes specifying its run time: AFTER, AT, CALENDAR, EVERY, or WAIT. On satisfaction of the attribute condition, the state changes to one of: • EVENT if the job also has the WAITON attribute and has not been released • TAPE if the job requires more tape drives than are available and its master jobs (if any) have released it • READY on satisfaction of all preceding conditions

- Aliases exist for these forms of the STATUS JOB command:

Command	Alias
STATUS JOB (STATE EXECUTING)	S[TATUS]-E[XECUTING]
STATUS JOB (STATE READY)	S[TATUS]-R[EADY]
STATUS JOB (U[SER] *user-ID*)	S[TATUS]-U[SER] [*user-ID*]

  user-ID in the S[TATUS]-U[SER] [*user-ID*] command specifies a user ID or a range of user IDs in *group-name.user-name* or *group-ID*,*user-ID* form. To specify a range of user IDs, use the asterisk (*) and question mark (?) wild-card characters. The default is the user ID of the current user.

- You can omit the object keyword JOB from the STATUS JOB command only when JOB is the current assumed object. For more information, see ASSUME JOB Command on page 6-72.

Examples

- This example shows the STATUS JOB command listing all jobs in a scheduler:

```
3} STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ---------- -------------
316 DEVELOPMENT 255,255 2109 4:Start err DVLPMNT
317 EVERYDAY 133,2 18:15:00 STANDARD-CLASS
318 TOTALS 255,205 EVENT STANDARD-CLASS
319 SUBTOTALS 255,255 1:Hold STANDARD-CLASS
320 TEST 205,100 2111 9:STALL STANDARD-CLASS
321 CLEANUP 205,100 2121 3:NEWP. err STANDARD-CLASS
```

```
322 CAL-JOB-1 205,70 7:CAL error STANDARD-CLASS
323 USERS 205,70 READY DEFAULT
324 BACKUP 8,255 TAPE STANDARD-CLASS
325 QUICK 8,1 RUN-NEXT DEFAULT
326 ACCOUNTS 8,1 2115 5:Fail RsOn STANDARD-CLASS
327 ADMIN 133,2 2117 6:Fail RsOffSTANDARD-CLASS
328 CAL-JOB-2 205,70 8:CAL exprd STANDARD-CLASS
329 STATISTICS 255,8 2119 2:NB failed STANDARD-CLASS
330 URGENT 255,8 RUN-NOW URGENT-JOBS
331 COMPILE 133,2 2122 SUSPENDED STANDARD-CLASS
332 Q3 205,70 2123 EXECUTING STANDARD-CLASS
333 WK940902 255,255 2127 OVER LIMIT DEFAULT
```

- This example shows the effect of the STATUS JOB command's SELPRI qualifier:

```
19} STATUS JOB 809
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
809 J1 133,2 492 EXECUTING DEFAULT
20} STATUS JOB 809, SELPRI
JOB STATUS
JOB JOBNAME USERID SEL STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
809 J1 133,2 7 EXECUTING DEFAULT
```

STATUS SCHEDULER Command

Use the STATUS SCHEDULER command to display scheduler details that include:

- The scheduler's process name, primary and backup CPUs and PINs, database location, log file name, and current time
- A count of the scheduler's executors, classes, and jobs in each of their respective states
- The number of active and suspended job processes, configured and in-use tape drives, and attachment sets

```
S[TATUS] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

Considerations

- The STATUS SCHEDULER command is available to all users.

- The time displayed by the STATUS SCHEDULER command is the system date and time on the node where the scheduler process is running.
- The STATUS SCHEDULER command displays information about a scheduler's executors, classes, jobs, job processes, attachment sets, and tape drives in columnar format. This table describes the contents of each column:

Column	Description
EXECUTORS	Lists the number of executors in each executor state. For information about executor states, see STOP EXECUTOR Command on page 6-178.
JOBS	Lists the number of jobs in each job state. For information about job states, see STATUS JOB Command on page 6-165.
CLASSES	Lists the number of classes in each class state. For information about class states, see INFO CLASS Command on page 6-103.
TAPE DRIVES	Lists the number of tape drives specified by the scheduler's TAPEDRIVES attribute and the number of those drives in use by jobs.
PROCESSES	Lists the number of active and suspended job processes.
ATTACHMENT SETS	Lists the number of attachment sets.

- You can omit the object keyword SCHEDULER from the STATUS SCHEDULER command only when SCHEDULER is the current assumed object. For more information, see ASSUME SCHEDULER Command on page 6-73.

Example

This example shows the result of a STATUS SCHEDULER command:

```
115} STATUS SCHEDULER
SCHEDULER STATUS
Process : \MELBDEV.$ZBAT Primary : 0,59 Backup : 2,101
Database: \MELBDEV.$DATA7.ZBAT
Logfile : \MELBDEV.$ZTN0.#PTY6
Time : 07OCT94 14:10:51
EXECUTORS JOBS CLASSES PROCESSES
---------- -------------- -------- --------------
OFF 1 READY 2 OFF 3 ACTIVE 4
ON 2 EXECUTING 1 ON 5 SUSPENDED 2
ACTIVE 1 SPECIAL 1
STOP 0 TIME 1
DOWN 0 EVENT 0 TAPE DRIVES ATTACHMENT SETS
DELETE 0 SUSPENDED 1 ------------ --------------
RUN NEXT 0 CONFIGURED 2 DEFINED 1
RUN NOW 0 IN USE 2
TAPE 1
```

STATUS-HISTORY Command

Use the STATUS-HISTORY command to list scheduler log-file events generated by a job. The command lists events from BEGIN (job initiation) through FINISH (job completion) for each run of the job.

```
S[TATUS]-H[ISTORY] [ / OU[T] [ file-name ] / ] job-ID
[ , N[B]-[LOG] log-file ] [ , RUN number ]
[ , AF[TER] [ date ] [ time ] ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

job-ID

specifies a job name or number.

log-file

specifies the name of the scheduler log file from which BATCHCOM is to start searching for log-file events. The file must be a disk log file (BATCHCOM cannot search non-disk log files). The default is the current log file.

BATCHCOM expands a partially qualified log-file name by using the defaults specified in the last VOLUME command. If there was no such command, the defaults current at the start of the session apply.

number

limits the search to log-file events for the specified job run. The default is the current run for an executing, over-limit, or suspended job and the most recent run for a job that is not executing, over limit, or suspended.

date

specifies the date from which BATCHCOM is to start searching for log-file events. You can enter the date in any of these forms. (For descriptions of the date forms, see AFTER Job Attribute on page 7-9.)

```
[ yy] yy [ m] m [ d] d mmm [ d] d [ d] d mmm day
[ yy] yy mmm [ d] d mmm [ d] d [ yy] yy [ d] d mmm [ yy] yy
[ yy] yymmdd
```

[*d*] *d mmm* and *mmm* [*d*] *d* refer to the current year. If you want to use one of these date forms and specify time, time must appear before date, not after.

The current date applies if you omit date. (The current date is the system date on the node where the scheduler is running.)

time

specifies the time from which BATCHCOM is to start searching for log-file events. You can enter the time in any of these forms. (For descriptions of the time forms, see AFTER Job Attribute on page 7-9.)

```
[ h] h:[ m] m[:[ s] s]BBC[( AALVS1(A[M],P[M]) )
BBC[( AALVS1(MIDD[AY],NOO[N]) )
MIDN[IGHT]
```

The start time of the current or most recent run applies if you omit time.

Considerations

- The STATUS-HISTORY command is available to all users. However, you must have read access to a log file for BATCHCOM to display its events.
- BATCHCOM terminates a search through a range of log files when it encounters a file secured against read access. To continue the search, enter a STATUS-HISTORY command that specifies the next read-accessible log file in the range you want searched.
- Different combinations of NB-LOG, RUN, and AFTER produce different results:

NB-LOG	RUN	AFTER	Result
No	No	No	Current run if job is executing, over limit, or suspended. If job is not executing, over limit, or suspended, the most recent run in current log
No	No	Yes	All runs in current log, starting from run whose BEGIN event occurs after specified date and time
No	Yes	No	Earliest occurrence of specified run in current log
No	Yes	Yes	Earliest occurrence of specified run in current log, but only if BEGIN event occurs after specified date and time
Yes	No	No	All runs from beginning of specified log through current log
Yes	No	Yes	All runs from specified log through current log, starting from run whose BEGIN event occurs after specified date and time
Yes	Yes	No	Earliest occurrence of specified run in logs from specified log through current log
Yes	Yes	Yes	Earliest occurrence of specified run in logs from specified log through current log, starting from run whose BEGIN event occurs after specified date and time

- The BEGIN log-file event description contains a job's run number appended to the job's name. For example, run number 399 of job DG2 would appear as:

```
BEGIN JOB (SUPER.SUPER)DG2:399 …
```

- The run number for a recurrent job (a job with the CALENDAR or EVERY attribute) increments by one each time the job runs. The run number for a nonrecurrent job is always one, no matter how many times the job runs.
- When STATUS-HISTORY command output goes to the current terminal, BATCHCOM displays the name of the log file being scanned on line 25. For example:

  ```
  Scanning : \MELRISK.$QA.ZBAT.LOGAAL
  ```

- When command output goes to a location other than the current terminal, the name of the log file being scanned appears in the output. For example:

  ```
  Scanning : \MELRISK.$QA.ZBAT.LOGAAN
  Scanning : \MELRISK.$QA.ZBAT.LOGAAO
  Scanning : \MELRISK.$QA.ZBAT.LOGAAP
  Scanning : \MELRISK.$QA.ZBAT.LOGAAQ
  Scanning : \MELRISK.$QA.ZBAT.LOGAAR
  14SEP94 10:04:51 BEGIN JOB …
  ```

- On a scheduler warm start when the scheduler's previous log file was a disk file and the scheduler encounters a file-system error on that file, this warning appears:

  ```
  *WARNING* Scheduler log-file chaining lost:
  Error file-system-error-no. Continuing …
  ```

  The warning, which has no effect on the scheduler, indicates a break in the disk log-file sequence. As a result, log-file scanning by the STATUS-HISTORY command terminates at the break with a "file not found" error instead of continuing through subsequent files. To continue the search, enter a STATUS-HISTORY command that specifies the next read-accessible log file in the range you want searched.

Example

This example shows the STATUS-HISTORY command listing the log-file events generated by the most recent run of job 29:

```
10} STATUS-HISTORY 29
08SEP94 16:12:01 BEGIN JOB (SUPER.SUPER)DG2:829 E_E4 J_29
P_DELAY U_255,255
08SEP94 16:12:02 LIST JOB DG2 EXECUTING J_29
08SEP94 16:12:02 START EXECUTOR-PROGRAM U_255,255 J_29
P_DELAY \MELRISK.$X747:8994225
08SEP94 16:13:02 STOP CC_0 EXECUTOR-PROGRAM J_29
\MELRISK.$X747:8994225
08SEP94 16:13:02 FINISH JOB DG2 T_0:0:0:2 J_29 P_DELAY
Output 5 Lines
```

STOP EXECUTOR Command

Use the STOP EXECUTOR command to stop executors whose state is ACTIVE or ON. The command makes the executors unavailable for use by jobs and changes their state to OFF.

```
STO[P] [ EXECUTOR ] [ / OU[T] [ file-name ] / ] executor-ID
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

executor-ID

is an executor name, or a range of executor names when specified with the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all executors.

Considerations

- The STOP EXECUTOR command is available to NetBatch supervisors only.
- A job using an executor that is the subject of a STOP EXECUTOR command finishes before the scheduler stops the executor. In this case, the executor's state pending stoppage is STOP.
- To restart an executor whose state is OFF or STOP, use the START EXECUTOR command.
- You can omit the object keyword EXECUTOR from the STOP EXECUTOR command only when EXECUTOR is the current assumed object. For more information, see ASSUME EXECUTOR Command on page 6-71.

Examples

- This example shows the effect of a STOP EXECUTOR * command:

```
86} STATUS EXECUTOR *
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
----------------------- --- ------ ---- -------------------
RISC-0 0 ON
RISC-1 1 ON
RISC-2 2 ON
RISC-3 3 ON
87} STOP EXECUTOR *
Executor RISC-0 stopped
Executor RISC-1 stopped
Executor RISC-2 stopped
```

```
Executor RISC-3 stopped
88} STATUS EXECUTOR *
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
----------------------- --- ------ ---- --------------------
RISC-0 0 OFF
RISC-1 1 OFF
RISC-2 2 OFF
RISC-3 3 OFF
```

- This example lists the scheduler log file events recorded as a result of the previous STOP EXECUTOR * command:

```
STOP EXECUTOR RISC-0 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
UPDATE EXECUTOR RISC-0 S_OFF U_255,255 H_\MELBDEV.$ZTN0.#PTY4
STOP EXECUTOR RISC-1 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
UPDATE EXECUTOR RISC-1 S_OFF U_255,255 H_\MELBDEV.$ZTN0.#PTY4
STOP EXECUTOR RISC-2 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
UPDATE EXECUTOR RISC-2 S_OFF U_255,255 H_\MELBDEV.$ZTN0.#PTY4
STOP EXECUTOR RISC-3 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
UPDATE EXECUTOR RISC-3 S_OFF U_255,255 H_\MELBDEV.$ZTN0.#PTY4
```

- This example shows the STOP EXECUTOR command stopping an executor in use by a job. The scheduler allows the job to finish before stopping the executor. Note the change in executor state from ACTIVE through STOP to OFF.

```
6} STATUS EXECUTOR PUBLICATIONS
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
----------------------- --- ------ ---- --------------------
PUBLICATIONS 0 ACTIVE 5 BOOKS
7} STOP EXECUTOR PUBLICATIONS
Executor PUBLICATIONS stopped
8} STATUS EXECUTOR PUBLICATIONS
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
----------------------- --- ------ ---- --------------------
PUBLICATIONS 0 STOP 5 BOOKS
9} STATUS EXECUTOR PUBLICATIONS
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
----------------------- --- ------ ---- --------------------
PUBLICATIONS 0 OFF
```

STOP JOB Command

Use the STOP JOB command to stop executing, over-limit, or suspended jobs. The command does not stop processes dissociated from a job by the TACL RUN command option JOBID or by the JOBID-ZERO job attribute.

```
STO[P] [ JOB ] [ / OU[T] [ file-name ] / ] { job-ID }
{ ( [ [NOT] job-ID , ]… [NOT] filter [ , [NOT] filter ]… )}
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

job-ID

specifies a job name or number, or a range of job names. To specify a range of job names, use the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all jobs.

NOT

excludes jobs that satisfy the specified job-selection criterion.

filter

specifies one of these job-selection criteria (see STATUS JOB Command on page 6-165):

```
A[TTACHMENT]-S[ET] [ [ ( user-ID ) ] attachment-set-ID
#CURRENT ]
CL[ASS] class-ID
IN file-ID
STATE state
U[SER] user-ID
WAITO[N] master-job
```

Considerations

- The STOP JOB command is available to all users, but these conditions apply:
- NetBatch supervisors can stop jobs belonging to any user.
- Users who are not NetBatch supervisors can stop any job whose input file is a disk file to which they have write access. If the input file does not exist or is a device or process, only the owner and NetBatch supervisors can stop the job.

  If you are not a NetBatch supervisor, a STOP JOB command that specifies *job-ID* with wild-card characters stops only your own jobs.

To stop another user's job if you have write access to its input file, specify the job's full name or number in *job-ID*.

- The STOP JOB command deletes nonrecurring jobs whose state is EXECUTING, OVER LIMIT, or SUSPENDED and whose attributes do not include HOLDAFTER ON. (A nonrecurring job does not have the CALENDAR or EVERY attribute.) The command does not delete executing, over-limit, or suspended jobs whose attributes include HOLDAFTER ON or jobs whose state is EVENT, READY, RUNNEXT, RUNNOW, SPECIAL-*n*, TAPE, or TIME. (For information about deleting these jobs, see DELETE JOB Command on page 6-84.)
- The scheduler reschedules an executing, over-limit, or suspended recurrent job that you delete with the STOP JOB command. (A recurrent job is a job whose attributes include CALENDAR or EVERY.) The rescheduled job is eligible to start at the next time specified by its attributes. To delete the job completely, use the DELETE JOB command after the STOP JOB command.
- You can omit the object keyword JOB from the STOP JOB command only when JOB is the current assumed object. For more information, see ASSUME JOB Command on page 6-72.

Examples

- This example shows execution of a STOP JOB command and the resulting scheduler log-file events:

```
> BATCHCOM $ZBAT; STOP JOB 61
Job TEST Jobnumber 61 stopped
> FUP COPY LOGABN,,SHARE .
.
OPEN U_255,255 P_BATCHCOM \QA.$:0:59:17109804
ABORT JOB TEST by NetBatch J_61 P_DELAY \QA.$Z639:17383725
U_255,255 H_\QA.$ZTN0.#PTY4
DOSTOP \QA.$Z639:17383725 J_61 U_255,255 H_\QA.$ZTN0.#PTY4
STOP CC_6 (ABEND) by \QA.$ZBAT:17381421 U_255,255
EXECUTOR-PROGRAM J_61 \QA.$Z639:17383725
UPDATE EXECUTOR E2 S_ON
FINISH JOB TEST T_0:0:0:9 J_61 P_DELAY
DELETE JOB TEST J_61
CLOSE \QA.$:0:59:17109804 U_255,255
```

- This example shows the effects of various STOP JOB commands executed by different users:

```
1} CHANGEUSER 205,70 password
1} STATUS JOB *
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ---------- -------------
220 DEVELOPMENT 255,255 1632 4:Start err DVLPMNT
221 EVERYDAY 133,2 18:15:00 STANDARD-CLASS
222 TOTALS 255,205 EVENT STANDARD-CLASS
223 SUBTOTALS 255,255 1:Hold STANDARD-CLASS
224 TEST 205,100 1635 9:STALL STANDARD-CLASS
```

```
225 CLEANUP 205,100 1645 3:NEWP. err STANDARD-CLASS
226 CAL-JOB-1 205,70 7:CAL error STANDARD-CLASS
227 USERS 205,70 READY DEFAULT
228 BACKUP 8,255 TAPE STANDARD-CLASS
229 QUICK 8,1 RUN-NEXT DEFAULT
230 ACCOUNTS 8,1 1639 5:Fail RsOn STANDARD-CLASS
231 ADMIN 133,2 1641 6:Fail RsOffSTANDARD-CLASS
232 CAL-JOB-2 205,70 8:CAL exprd STANDARD-CLASS
233 STATISTICS 255,8 1643 2:NB failed STANDARD-CLASS
234 URGENT 255,8 RUN-NOW URGENT-JOBS
235 COMPILE 133,2 1646 SUSPENDED STANDARD-CLASS
236 Q3 205,70 1648 EXECUTING STANDARD-CLASS
2} STOP JOB *
Job Q3 job number 236 stopped
3} STOP JOB COMPILE
2132-E Your user code does not give you access to COMPILE
4} CHANGEUSER 255,8 password
4} STOP JOB *
Job COMPILE job number 235 stopped
5} STOP JOB 231
2146-E Job not executing or suspended; STOP command ignored
```

- This example shows the STOP JOB command stopping processes associated with a job. The command does not stop the processes dissociated from the job by the TACL RUN command option JOBID.

```
> STATUS *, USER
Process Pri PFR %WT Userid Program file Hometerm
$C2 0,51 150 R 000 205,70 $SYS.SYS.TACL $TRM2.#A
$C2 B 2,22 150 001 205,70 $SYS.SYS.TACL $TRM2.#A
> BATCHCOM /IN WXYZ/ $ZBAT
SUBMIT JOB DELAYS, EXECUTOR-PROGRAM TACL, IN WXYZIN
Job DELAYS job number 249 submitted
> FUP COPY WXYZIN
DELAY /NAME $W, NOWAIT/ 5 MINS
DELAY /NAME $X, JOBID 0, NOWAIT/ 5 MINS
DELAY /NAME $Y, JOBID 8, NOWAIT/ 5 MINS
DELAY /NAME $Z/ 5 MINS
> STATUS *, USER
Process Pri PFR %WT Userid Program file Hometerm
$C2 0,51 150 R 000 205,70 $SYS.SYS.TACL $TRM2.#A
$W 0,84 119 000 205,70 $SYS.SYS.DELAY $ZBAT
$Y 0,92 119 000 205,70 $SYS.SYS.DELAY $ZBAT
$Z 0,97 119 000 205,70 $SYS.SYS.DELAY $ZBAT
$Y744 0,100 120 001 205,70 $SYS.SYS.TACL $ZBAT
$X 0,101 119 000 205,70 $SYS.SYS.DELAY $ZBAT
$C2 B 2,22 150 001 205,70 $SYS.SYS.TACL $TRM2.#A
> STATUS *, GMOMJOBID $ZBAT.249
Process Pri PFR %WT Userid Program file Hometerm
$W 0,84 119 000 205,70 $SYS.SYS.DELAY $ZBAT
$Z 0,97 119 000 205,70 $SYS.SYS.DELAY $ZBAT
$Y744 0,100 120 001 205,70 $SYS.SYS.TACL $ZBAT
> BATCHCOM $ZBAT; STOP JOB 249
Job DELAYS job number 249 stopped|
> STATUS *, GMOMJOBID $ZBAT.249
```

```
> STATUS *, USER
Process Pri PFR %WT Userid Program file Hometerm
$C2 0,51 150 R 000 205,70 $SYS.SYS.TACL $TRM2.#A
$Y 0,92 119 000 205,70 $SYS.SYS.DELAY $ZBAT
$X 0,101 119 000 205,70 $SYS.SYS.DELAY $ZBAT
$C2 B 2,22 150 001 205,70 $SYS.SYS.TACL $TRM2.#A
```

SUBMIT JOB Command

Use the SUBMIT JOB command to submit jobs to a scheduler.

```
SUB[MIT] [ JOB ] [ / OU[T] [ file-name ] / ] [ job-name-1 ]
[ , LIK[E] job-name-2 ] [ , attribute ]…
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

job-name-1

is the name of the job you want to submit. The name can contain from 1 to 24 letters and numbers. It also can contain hyphens but must begin with a letter and end with a letter or number. The name cannot contain spaces.

Note. To avoid confusion, do not use these keywords and keyword aliases as job names: ADP, ATTACHMENT-SET, CLASS, EXECUTOR, JOB, JOBCLASS, and SCHEDULER.

Omitting *job-name-1* makes the scheduler generate a job name of the form ZBAT- *nnnn*, where *nnnn* is the job's scheduler-assigned number. The scheduler replaces a user-specified ZBAT- *nnnn* job name with a generated name.

LIKE

specifies that all attributes are to match those of job *job-name-2*. LIKE overrides job attributes in the working-attributes set and job defaults (see Table 6-9 on page 6-185). *attribute* overrides *job-name-2* attributes of the same type.

job-name-2

is the name of an existing job whose attributes job *job-name-1* is to match.

attribute

is one of these job attributes (see Section 7, Attributes):

```
AF[TER] [ date ] [ time ]
AT [ date ] [ time ]
A[TTACHMENT]-S[ET]
[ [ ( user-ID ) ] attachment-set-ID | #CURRENT ]
```

```
CA[LENDAR] [ file-name ]
CL[ASS] class-name
DES[CRIPTION] " [ string ] "
EV[ERY] [ weeks WEEK[S] ]
[ days D[AYS] ]
[ hours [ : mins ] [HOURS] ]
[ hours H[OURS] [ mins MIN[UTES] ] ]
[ crontab-entry ]
E[XECUTOR]-P[ROGRAM] file-name
EXT[SWAP] { $volume | file-name }
HIG[HPIN] { OF[F] | ON }
HOLD { OF[F] | ON }
HOLDA[FTER] { OF[F] | ON }
IF[FAILS] { OF[F] | ON }
IN [ file-name ]
J[OB]-L[OG] [ log-file ]
J[OBID]-Z[ERO] { OF[F] | ON }
LIB [ file-name ]
LIM[IT] hours [ : mins ]
MAXPRINTL[INES] { number | NON[E] }
MAXPRINTP[AGES] { number | NON[E] }
ME[M] number
NA[ME] $process-name
OU[T] [ file-name ]
PF[S] number
PR[I] number
P[URGE]-[IN]-[FILE] { OF[F] | ON }
REST[ART] { OF[F] | ON }
RUND { OF[F] | ON }
SA[VEABEND] { OF[F] | ON }
SEL[PRI] number
STAL[L] { OF[F] | ON }
STARTU[P] " param-set "
S[TOP]-[ON]-[ABEND] { OF[F] | ON }
SWA[P] { $volume | file-name }
TA[PEDRIVES] number
TERM $process-name
V[OLUME] { \node. [ $volume ] [ , " security " ] }
{ [ \node. ] $volume [ , " security " ] }
{ [ \node. ] [ $volume ] , " security " }
WAIT hours [ : mins ]
WAITO[N] [ job-name [ case ] ]
[ ( job-name [ case ] [ , job-name [ case ] ]… ) ]
```

attribute overrides attributes of the same type in the working-attributes set or specified by *job-name-2*. It also overrides job defaults (see Table 6-9 on page 6-185) when they do not appear in the working set. If you omit *attribute* and do not specify LIKE, the job adopts its attributes from the working-attributes set and job defaults.

Table 6-9. Job Defaults (page 1 of 2)

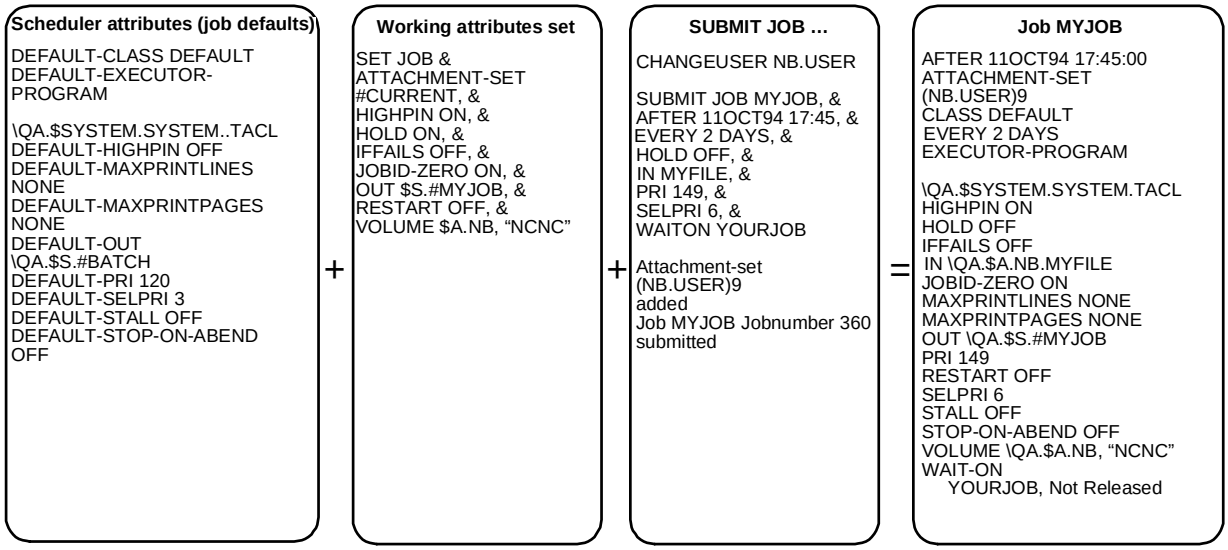
Attribute	Default Value
AFTER	No AFTER attribute.
AT	No AT attribute.
ATTACHMENT-SET	No ATTACHMENT-SET attribute.
CALENDAR	No CALENDAR attribute.
CLASS	DEFAULT-CLASS scheduler attribute.
DESCRIPTION	No DESCRIPTION attribute.
EVERY	No EVERY attribute.
EXECUTOR-PROGRAM	DEFAULT-EXECUTOR-PROGRAM scheduler attribute.
EXTSWAP	No EXTSWAP attribute. Extended swap file created on volume specified by SWAP attribute of DEFINE =_DEFAULTS. If DEFINE does not have SWAP attribute, volume selected by operating system.
HIGHPIN	DEFAULT-HIGHPIN scheduler attribute.
HOLD	No HOLD attribute.
HOLDAFTER	No HOLDAFTER attribute.
IFFAILS	No IFFAILS attribute.
IN	No IN attribute. Job uses scheduler as its input file.
JOB-LOG	No JOB-LOG attribute.
JOBID-ZERO	OFF.
LIB	No LIB attribute. Executor program uses library file specified when program last ran. The program runs without a library file if that is how it ran previously. The current library file applies if program is running.
LIMIT	No LIMIT attribute.
MAXPRINTLINES	DEFAULT-MAXPRINTLINES scheduler attribute.
MAXPRINTPAGES	DEFAULT-MAXPRINTPAGES scheduler attribute.
MEM	No MEM attribute. Job has memory pages allotted according to memory-pages value specified in executor program. This default also applies when MEM attribute specifies a number less than that value.
NAME	No NAME attribute. Executor-program process assigned system-generated name.
OUT	DEFAULT-OUT scheduler attribute.
PFS	No PFS attribute. Executor program has PFS size specified in program.
PRI	DEFAULT-PRI scheduler attribute.
PURGE-IN-FILE	No PURGE-IN-FILE attribute.
RESTART	No RESTART attribute.

Table 6-9. Job Defaults (page 2 of 2)

Attribute	Default Value
RUND	No RUND attribute.
SAVEABEND	No SAVEABEND attribute.
SELPRI	DEFAULT-SELPRI scheduler attribute.
STALL	DEFAULT-STALL scheduler attribute.
STARTUP	No STARTUP attribute.
STOP-ON-ABEND	DEFAULT-STOP-ON-ABEND scheduler attribute.
SWAP	No SWAP attribute. Swap file created on volume specified by SWAP attribute of DEFINE=_DEFAULTS. If DEFINE does not have SWAP attribute, volume selected by operating system.
TAPEDRIVES	No TAPEDRIVES attribute.
TERM	No TERM attribute. Job uses scheduler as its home terminal.
VOLUME	Node, volume, subvolume, and security defaults set by the last SET JOB VOLUME command. If there was no such command, the default values are the defaults current when the session began.
WAIT	No WAIT attribute.
WAITON	No WAITON attribute.

Figure 6-7 illustrates some of the attribute conditions.

Figure 6-7. Example of Attribute Defaulting-Job



Considerations

- The SUBMIT JOB command is available to all users.
- When the scheduler accepts a submitted job, BATCHCOM displays a message that includes the scheduler-generated job number. For example:

  ```
  Job ZBAT-0088 Jobnumber 88 submitted
  ```

  You can use the job's number instead of its name in commands that require you to specify a job. For example, to list the attributes of the job identified in the preceding message, enter INFO JOB ZBAT-0088 or INFO JOB 88.

- The scheduler assigns job numbers consecutively in the range 1 through 32767 for systems running G-series RVUs, and 1 through 9999 for systems running H-series RVUs. But it skips the numbers already assigned. These job numbering considerations also apply:

  - Numbering continues from where it left off when you warm start a stopped scheduler. For example, if the last number assigned before scheduler shutdown was 361, the first number assigned after warm start would be 362.
  - Numbering starts at 1 when you cold start a scheduler, regardless of whether the scheduler has run in the past.
  - Numbering is cyclic, beginning again at 1 after reaching the maximum job number (32767 for system running G-series RVUs, and 9999 for systems running H-series RVUs). The maximum number of job records the scheduler can store in its database is 32767 for systems running G-series RVUs and 9999 for systems running H-series RVUs.

- A job is eligible for execution on acceptance by the scheduler unless delayed by time, HOLD, TAPEDRIVES, or WAITON attributes. (Time attributes are AFTER, AT, CALENDAR, EVERY, and WAIT.) The scheduler executes the job on satisfaction of all attribute conditions and if these conditions exist:

  - The job's class and the scheduler have the attribute INITIATION ON.
  - An available, started executor is associated with the job's class. If the job's attributes include AT or the job was operated on by the RUNNOW JOB command, the scheduler creates a temporary executor that it deletes when the job finishes. For more information about temporary executors, see RUNNOW JOB Command on page 6-128 and AT Job Attribute on page 7-15.

- The scheduler assigns job ownership to the user logged on at job submission time. To display the ID of this user, execute a SHOW JOB command.
- This consideration discusses how daylight-saving time (DST) transitions affect jobs. It applies only to nodes loaded with DST transition tables. It does not apply where system times change by means of the TACL SETTIME command. To avoid unexpected results, do not submit a job to run in the period lost at the start or gained at the end of DST. If you must submit jobs to run in these periods:

- Start of DST—clocks set ahead of local standard time (LST): Jobs due to run in the period lost run immediately after DST starts because their run times are then in the past. As a result, the jobs run earlier than expected by up to the amount of time lost. For example, DST starts at 0200 LST when clocks go forward one hour. A job submitted at 0130 LST with the attribute AT 02:30 runs at 0300 DST (that is, 30 minutes after submission instead of 60 minutes after). A job submitted at 0130 LST with the attribute EVERY 06 hrs runs at 08:30 DST (for frequency less than a DAY, the EVERY clause calculates the time interval in GMT).
- End of DST—clocks reset to LST: Jobs due to run in the period gained run at their specified times in LST, but not in DST. As a result, the jobs run later than expected by up to the amount of time gained. For example, DST ends at 0300 DST when clocks go back one hour. A job submitted at 0230 DST with the attribute AT 03:00 runs 90 minutes later at 0300 LST (that is, the second occurrence of 0300). A job submitted at 0130 DST with the attribute EVERY 06 hrs runs at 06:30 LST.

- ADD is an alias of the command keyword SUBMIT.
- You can omit the object keyword JOB from the SUBMIT JOB command only when JOB is the current assumed object. For more information, see ASSUME JOB Command on page 6-72.

Examples

- This example shows the submission of a job. The job's attributes include scheduler-supplied defaults and attributes from the working-attributes set.

```
22} INFO SCHEDULER; SHOW JOB
SCHEDULER ATTRIBUTES
backupcpu: 0
at-allowed: On
submit-allowed: On
initiation: On
default-executor-program: \MELBDEV.$SYSTEM.SYSTEM.TACL
max-pri: 199
default-pri: 120
max-concurrent-jobs: 20,20
default-selpri: 3
default-maxprintlines: None
default-maxprintpages: None
tapedrives: 2
default-out: \MELBDEV.$S.#BATCH
default-class: DEFAULT
default-stall: Off
default-stop-on-abend: Off
default-highpin: Off
catchup: On
ems: Off
JOB ATTRIBUTES
volume: \MELBDEV.$A.NB, "NCNU"
```

```
jobid-zero: Off
user: 133,2
23} SUBMIT JOB
Job ZBAT-0089 Jobnumber 89 submitted
24} INFO JOB 89
JOB ATTRIBUTES for ZBAT-0089
jobnumber: 89
volume: \MELBDEV.$A.NB, "NCNU"
out: \MELBDEV.$S.#BATCH
executor-program: \MELBDEV.$SYSTEM.SYSTEM.TACL
jobid-zero: Off
pri: 120
selpri: 3
maxprintlines: None
maxprintpages: None
class: DEFAULT
stall: Off
stop-on-abend: Off
highpin: Off
submit: 11OCT94 13:53:47
alter: 11OCT94 13:53:48
user: 133,2
next-runtime: 11OCT94 13:53:48
```

- This example shows the effect of the scheduler attribute SUBMIT-ALLOWED OFF on job submissions:

```
29} INFO SCHEDULER, SUBMIT-ALLOWED
SCHEDULER ATTRIBUTES
submit-allowed: Off
30} SUBMIT JOB
2128-E SUBMIT-ALLOWED is OFF; the scheduler is not accepting
job submissions
```

- This example shows the SUBMIT JOB command submitting a job to scheduler $ZBAT. The example also lists events related to the job's submission and execution as recorded in the scheduler log file.

```
> LOGON 205,70, password
> BATCHCOM $ZBAT; SUBMIT JOB, EXECUTOR-PROGRAM DELAY, STARTUP
"1 MINS"
Job ZBAT-0092 Jobnumber 92 submitted
> BATCHCOM $ZBAT; STATUS JOB 92
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- --------- -------------
92 ZBAT-0092 205,70 689 EXECUTING DEFAULT
> STATUS EXECUTOR *
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
---------------------- --- ------ ---- -------
E1 1 ACTIVE 92 DEFAULT
> FUP COPY $DATA7.ZBAT.LOGABL,,SHARE .
.
ADD JOB ZBAT-0092 C_DEFAULT:3 J_92 U_205,70
H_\MELBDEV.$ZTN0.#PTY4
```

```
BEGIN JOB (FPP.USER)ZBAT-0092:1 E_E1 L_689 J_92 P_DELAY
\MELBDEV.$Z714:17394989 U_205,70
UPDATE EXECUTOR E1 S_ACTIVE
LIST JOB ZBAT-0092 EXECUTING J_92
START EXECUTOR-PROGRAM U_205,70 J_92 P_DELAY
\MELBDEV.$Z714:17394989
STOP CC_0 EXECUTOR-PROGRAM J_92 \MELBDEV.$Z714:17394989
UPDATE EXECUTOR E1 S_ON
FINISH JOB ZBAT-0092 T_0:0:0:12 J_92 P_DELAY
DELETE JOB ZBAT-0092 J_92
```

SUSPEND JOB Command

Use the SUSPEND JOB command to suspend executing processes associated with jobs. The command does not suspend processes dissociated from a job by the TACL RUN command option JOBID or by the JOBID-ZERO job attribute.

```
SUS[PEND] [ JOB ] [ / OU[T] [ file-name ] / ]
{ job-ID }
{ ( [ [NOT] job-ID ,]… [NOT] filter [ , [NOT] filter ]… )
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

job-ID

specifies a job name or number, or a range of job names. To specify a range of job names, use the asterisk (*) and question mark (?) wild-card characters. An asterisk on its own specifies all jobs.

NOT

excludes jobs that satisfy the specified job-selection criterion.

filter

specifies one of these job-selection criteria (described in STATUS JOB Command on page 6-165):

```
A[TTACHMENT]-S[ET] [ [ ( user-ID ) ] attachment-set-ID
#CURRENT ]
CL[ASS] class-ID
IN file-ID
STATE state
U[SER] user-ID
WAITO[N] master-job
```

Considerations

- The SUSPEND JOB command is available to all users, but these conditions apply:
  - NetBatch supervisors can suspend jobs belonging to any user.
  - Users who are not NetBatch supervisors can suspend any job whose input file is a disk file to which they have write access. If the input file does not exist or is a device or process, only the owner and NetBatch supervisors can suspend the job.

    If you are not a NetBatch supervisor, a SUSPEND JOB command that specifies *job-ID* with wild-card characters suspends only your own jobs. To suspend another user's job if you have write access to its input file, specify the job's full name or number in *job-ID*.
- Suspended processes associated with a job cannot execute instructions until you reactivate the job by using the ACTIVATE JOB command.
- A suspended job whose attributes include EVERY accumulates a run backlog if suspended for longer than the EVERY interval and the scheduler has the attribute CATCHUP ON. When you reactivate the job, it runs repeatedly until the backlog clears. For more information about run backlogs (including how to prevent them from running), see ACTIVATE JOB Command on page 6-32.
- A suspended job whose attributes include CALENDAR does not accumulate a run backlog while suspended. When you reactivate the job, it finishes executing but does not run again until the next future CALENDAR time.
- The executor in use by a suspended job is not available for use by other jobs while the job remains suspended. To release the executor for use by other jobs, do one of:
  - Reactivate the suspended job by using the ACTIVATE JOB command. The scheduler releases the executor when the job finishes executing.
  - Stop the suspended processes of the job by using the STOP JOB command. The scheduler reschedules the job if it is recurrent or delete the job if nonrecurrent. (To prevent deletion of the job, set the job's HOLDAFTER attribute to ON before using the STOP JOB command.)
- You can omit the object keyword JOB from the SUSPEND JOB command only when JOB is the current assumed object. For more information, see ASSUME JOB Command on page 6-72.

Example

This example shows SUBMIT JOB, SUSPEND JOB, and ACTIVATE JOB commands submitting, suspending, and reactivating a job and the resulting scheduler log-file events:

```
> BATCHCOM $ZBAT; SUBMIT JOB, EXECUTOR-PROGRAM DELAY, STARTUP
"4 MINS"
Job ZBAT-0098 Jobnumber 98 submitted
> BATCHCOM $ZBAT; SUSPEND JOB 98
Job ZBAT-0098 Jobnumber 98 suspended
> BATCHCOM $ZBAT; ACTIVATE JOB 98
Job ZBAT-0098 Jobnumber 98 activated
> FUP COPY LOGABO,,SHARE .
.
ADD JOB ZBAT-0098 C_DEFAULT:3 J_98 U_255,255
H_\MELBDEV.$ZTN0.#PTY4
BEGIN JOB (SUPER.SUPER)ZBAT-0098:1 E_E1 L_696 J_98 P_DELAY
\MELBDEV.$Z725:17397805 U_255,255
UPDATE EXECUTOR E1 S_ACTIVE
LIST JOB ZBAT-0098 EXECUTING J_98
START EXECUTOR-PROGRAM U_255,255 J_98 P_DELAY
\MELBDEV.$Z725:17397805
SUSPND JOB ZBAT-0098 J_98 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
LIST JOB ZBAT-0098 SUSPENDED J_98 U_255,255
H_\MELBDEV.$ZTN0.#PTY4
A'VATE JOB ZBAT-0098 J_98 U_255,255 H_\MELBDEV.$ZTN0.#PTY4
LIST JOB ZBAT-0098 EXECUTING J_98 U_255,255
H_\MELBDEV.$ZTN0.#PTY4
STOP CC_0 EXECUTOR-PROGRAM J_98 \MELBDEV.$Z725:17397805
UPDATE EXECUTOR E1 S_ON
FINISH JOB ZBAT-0098 T_0:0:0:12 J_98 P_DELAY
DELETE JOB ZBAT-0098 J_98
```

SWITCHCPU SCHEDULER Command

Use the SWITCHCPU SCHEDULER command to make a scheduler's primary process run in the CPU of its backup process, and the backup process run in the CPU of its primary process.

```
SWITCHC[PU] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

Considerations

- The SWITCHCPU SCHEDULER command is available to NetBatch supervisors only.
- The SWITCHCPU SCHEDULER command switches scheduler processes only between the current primary and backup CPUs. You cannot use the BACKUPCPU attribute with the command to specify a different backup CPU.
- You can omit the object keyword SCHEDULER from the SWITCHCPU SCHEDULER command only when SCHEDULER is the current assumed object. For more information, see ASSUME SCHEDULER Command on page 6-73.

Example

This example shows the SWITCHCPU SCHEDULER command switching the processes of scheduler $ZBAT between CPUs 0 and 2:

```
117} STATUS SCHEDULER .
Process : \MELBDEV.$ZBAT Primary : 0,59 Backup : 2,101 .
118} SWITCHCPU SCHEDULER
Scheduler CPUs switched
119} STATUS SCHEDULER .
Process : \MELBDEV.$ZBAT Primary : 2,101 Backup : 0,59
Scheduler log-file events recording the CPU switch from the
previous example are:
****** U_255,255 H_\MELBDEV.$ZTN0.#PTY4
CHKPNT takeover from DOIT^SCHEDULER+3120I U_255,255
H_\MELBDEV.$ZTN0.#PTY4
SWITCH CPU Scheduler process, primary 0 , backup 2 U_255,255
H_\MELBDEV.$ZTN0.#PTY4
CHKPNT Save duplicate request reply, sync id: 39
\MELBDEV.$:0:73:17094444 U_255,255
H_\MELBDEV.$ZTN0.#PTY4
```

SWITCHLOG SCHEDULER Command

Use the SWITCHLOG SCHEDULER command to close the current scheduler log file and open another.

```
SWITCHL[OG] [ SCHEDULER ] [ / OU[T] [ file-name ] / ]
[ log-file ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

log-file

is the name of a scheduler log file. The file can be a device, a process, or any type of unstructured disk file except an EDIT file. If you specify a disk file, these conditions apply:

- If the file exists and is not an EDIT file, the scheduler opens it and appends events. If the file is an EDIT file, the command fails.
- If the file does not exist or if you omit *log-file*, the scheduler creates a file (file code 847) named *svol*.LOG*xxx*. *svol* specifies the volume and subvolume of the previous log file (if that file was a disk file) or, if there was no such file, the scheduler's database subvolume. *xxx* are characters in the range AAA through ZZZ.

  The default LOG *xxx* sequence is LOGAAA, LOGAAB, … LOGAAZ, LOGABA, LOGABB, … LOGZZZ, LOGAAA, and so on. If a name in the sequence exists, the scheduler uses the next available name.

log-file cannot specify a DEFINE.

Considerations

- The SWITCHLOG SCHEDULER command is available to NetBatch supervisors only.
- A scheduler-created disk log file adopts these attributes of the previous log if that log was a disk file: primary and secondary extent sizes, maximum extents, buffer size, expiration time, RWEP security, owner, and volume and subvolume. The default extent size of a scheduler-created disk log file is 50 primary extents and 100 secondary extents. The default maximum extents the file can have is 100. (The scheduler can record up to 154,375 events in a file of this size.)
- When a log file is full, the scheduler automatically closes it and opens a new file with the default name *svol*.LOG *xxx*.
- You can omit the object keyword SCHEDULER from the SWITCHLOG SCHEDULER command only when SCHEDULER is the current assumed object. For more information, see ASSUME SCHEDULER Command on page 6-73.

Examples

- This SWITCHLOG SCHEDULER command closes the current log file and opens another with the default name LOG *xxx*:

```
1} STATUS SCHEDULER
SCHEDULER STATUS
Process : \MELBDEV.$TEST Primary : 0,60 Backup : 1,41
Database: \MELBDEV.$DATA7.TEST
Logfile : \MELBDEV.$NB.LOGS.TESTLOG .
.
2} SWITCHLOG SCHEDULER
```

```
Scheduler logfile switched
3} STATUS SCHEDULER
SCHEDULER STATUS
Process : \MELBDEV.$TEST Primary : 0,60 Backup : 1,41
Database: \MELBDEV.$DATA7.TEST
Logfile : \MELBDEV.$DATA7.TEST.LOGAAB .
.
```

- These examples list the scheduler log file events that record the results of the previous SWITCHLOG SCHEDULER command:
  - Last event in log file $NB.LOGS.TESTLOG:

    ```
    LOG \MELBDEV.$NB.LOGS.TESTLOG TO
    \MELBDEV.$DATA7.TEST.LOGAAB
    ```

  - First two events in log file $DATA7.TEST.LOGAAB:

    ```
    LOG \MELBDEV.$NB.LOGS.TESTLOG TO
    \MELBDEV.$DATA7.TEST.LOGAAB
    LOG SWITCHLOG \MELBDEV.$NB.LOGS.TESTLOG TO
    \MELBDEV.$DATA7.TEST.LOGAAB U_255,255
    H_\MELBDEV.$ZTN0.#PTY6
    ```

- This example shows the SWITCHLOG SCHEDULER command changing a scheduler log file to the Event Management Service (EMS) collector process $0:

```
> BATCHCOM $TEST; STATUS SCHEDULER
SCHEDULER STATUS
Process : \MELBDEV.$TEST Primary : 0,60 Backup : 1,41
Database: \MELBDEV.$DATA7.TEST
Logfile : \MELBDEV.$DATA7.TEST.LOGAAB .
.
> BATCHCOM $TEST; SWITCHLOG SCHEDULER $0
Scheduler logfile switched
> BATCHCOM $TEST; STATUS SCHEDULER
SCHEDULER STATUS
Process : \MELBDEV.$TEST Primary : 0,60 Backup : 1,41
Database: \MELBDEV.$DATA7.TEST
Logfile : \MELBDEV.$0 .
.
```

SYSTEM Command

Use the SYSTEM command to specify the default node for a BATCHCOM session. BATCHCOM uses this default when expanding partial file names specified in BATCHCOM commands.

```
SY[STEM] [ / OU[T] [ file-name ] / ] [ \node ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT

file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

node

is a node name. The default is the node where the process that creates the BATCHCOM process is running.

Considerations

- The SYSTEM command is available to all users.
- Entering SYSTEM by itself resets the default node to the node current at the start of the session.

Example

This example shows the effect of two SYSTEM commands. The first command sets the default node to \MELBDEV. The second command restores the default node to the node current when the session began (\MELBQAT).

```
1} VOLUME
VOLUME \MELBQAT.$VOL1.SUBVOL1, "NCNC"
2} SYSTEM \MELBDEV
SYSTEM \MELBDEV.$VOL1.SUBVOL1, "NCNC"
3} SYSTEM
SYSTEM \MELBQAT.$VOL1.SUBVOL1, "NCNC"
```

VOLUME Command

Use the VOLUME command to specify the default node, volume, and subvolume for a BATCHCOM session. BATCHCOM uses these defaults when expanding partial file names specified in BATCHCOM commands. You also can use the VOLUME command to specify the default security for disk files BATCHCOM creates during a session.

```
V[OLUME] [ / OU[T] [ file-name ] / ] [ [ \node. ] volume ]
[ , " security " ]
```

file-name

specifies a command output file. The file can be a device, a process, or a disk file. For a disk file, BATCHCOM appends output if the file exists, but creates an EDIT file if the file does not exist. If you specify / OUT / (that is, omit *file-name*), BATCHCOM suppresses output.

node

is a node name.

volume

is one of:

$*volume-name.subvolume-name*

are volume and subvolume names.

$*volume-name*

is a volume name.

subvolume-name

is a subvolume name. If you specify *subvolume-name* but omit $*volume-name*, BATCHCOM resets the default volume to the volume from the current TACL environment.

security

is a four-character string specifying Guardian security for read, write, execute, and purge (RWEP) file access. For each type of access, specify one of these codes to indicate the required security level:

Code	Description
A	Anyone—Lets any local user access the file.
N	Network—Lets any local or remote user access the file.
G	Group—Lets any local user in the owner's group access the file.
C	Community—Lets any local or remote user in the owner's group access the file.
O	Owner—Lets only the local owner access the file.
U	User—Lets the owner access the file from a local or remote node.
-	Hyphen—Lets only the local super ID (255,255) access the file. Available as an option for the W, E, and P security attributes only.

Considerations

- The VOLUME command is available to all users.
- BATCHCOM adopts its initial node, volume, subvolume, and security defaults from the current TACL environment. To revert to these defaults at any time during a session, type VOLUME on its own, then press RETURN.
- BATCHCOM maintains the VOLUME attribute of the DEFINE =_DEFAULTS in parallel with the current default node, volume, and subvolume. A VOLUME command changes DEFINE =_DEFAULTS in the working-attributes set. A SET ATTACHMENT-SET (DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME …)

command changes the current default node, volume, and subvolume. For example:

```
1} VOLUME
VOLUME $SYSTEM.NETBATCH, "NNNC"
2} SHOW ATTACHMENT-SET DEFINE =_DEFAULTS
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$SYSTEM.NETBATCH
3} VOLUME \MELBQAT.$QAT2.ISPFILES
VOLUME \MELBQAT.$QAT2.ISPFILES, "NNNC"
4} SHOW ATTACHMENT-SET DEFINE =_DEFAULTS
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
\MELBQAT.$QAT2.ISPFILES
```

Example

This example shows the effect of two VOLUME commands. The first command displays defaults inherited by BATCHCOM from the TACL environment. The second command sets the default volume, subvolume, and disk file security for the session. These defaults enable BATCHCOM to find the file specified by the OBEY command. They also specify the security attributes of the file created as a result of execution of that command.

```
$DATA7.BTCH 15> FILEINFO $NB.NBFILES.*
$NB.NBFILES
Code EOF Last Modification Owner RWEP PExt SExt
X 101 2086 14-Apr-93 16:54:31 205,70 "NCNC" 2 2
$DATA7.BTCH 16> BATCHCOM
1} VOLUME
VOLUME \MELBDEV.$DATA7.BTCH, "NCNC"
2} OBEY X
0011-E File or record does not exist, X
3} VOLUME $NB.NBFILES, "AAAA"
VOLUME \MELBDEV.$NB.NBFILES, "AAAA"
4} OBEY X
STATUS JOB /OUT Y/ *
5} EXIT
$DATA7.BTCH 17> FILEINFO $NB.NBFILES.*
$NB.NBFILES
Code EOF Last Modification Owner RWEP PExt SExt
X 101 2086 14-Apr-93 16:54:31 205,70 "NCNC" 2 2
Y 101 442 15-Apr-93 11:14:36 205,70 "AAAA" 4 20
```

! Command

Use the ! (exclamation point) command to reexecute immediately a command line from BATCHCOM's history buffer.

```
! [ num | - num | text ]
```

num

is a positive integer identifying a line in the history buffer.

- num

is a negative integer identifying a line in the history buffer relative to the current line.

text

is a character string identifying the latest line in the history buffer beginning with the string.

Considerations

- The ! command is available to all users.
- If no line in the history buffer matches your specification, this message appears:

  ```
  0541-I No such line
  ```

- Entering ! with no parameter immediately reexecutes the latest line from the history buffer (! is the same as ! -1).
- BATCHCOM stores all commands in its history buffer except CHANGEUSER, COMMENT, and HISTORY.

Examples

These examples of the ! command relate to these commands in the history buffer:

```
1} OPEN $SCHD
2} STATUS SCHEDULER
3} SWITCHCPU SCHEDULER
4} OPEN $ZBAT
5} STATUS SCHEDULER
6} SWITCHLOG SCHEDULER
```

- To execute the latest line from the buffer:

  ```
  7} !
  7} SWITCHLOG SCHEDULER
  Scheduler logfile switched
  8}
  ```

- To execute command line 1 from the buffer:

  ```
  7} ! 1
  7} OPEN $SCHD
  NETBATCH SERVER - T9190D30 … Time: 06OCT94 12:41:40
  8}
  ```

- To execute the fourth command line preceding the current command line:

```
7} ! -4
7} SWITCHCPU SCHEDULER
Scheduler CPUs switched
8}
```

- To execute the latest command line that begins with ST:

```
7} ! ST
7} STATUS SCHEDULER
SCHEDULER STATUS
Process : \MELBDEV.$SCHD Primary : 1,67 Backup : 3,38
Database: \MELBDEV.$DATA7.SCHD
Logfile : \MELBDEV.$DATA7.SCHD.LOGAAE
Time : 06OCT94 12:42:19 .
.
8}
```

? Command

Use the ? (question mark) command to display command lines from BATCHCOM's history buffer.

```
? [ num | - num | text ]
```

num

is a positive integer identifying a line in the history buffer.

- num

is a negative integer identifying a line in the history buffer relative to the current line.

text

is a character string identifying the latest line in the history buffer beginning with the string.

Considerations

- The ? command is available to all users.
- If no line in the history buffer matches your specification, this message appears:

```
0541-I No such line
```

- Entering ? with no parameter displays the latest line from the history buffer (? is the same as ? -1).
- You can execute a displayed command by pressing RETURN after the BATCHCOM prompt returns.

- BATCHCOM stores all commands in its history buffer except CHANGEUSER, COMMENT, and HISTORY.

Examples

These examples of the ? command relate to these commands in the history buffer:

```
3} STATUS SCHEDULER
4} ASSUME JOB
5} SUBMIT TRIAL-BAL, IN TB, OUT $S, WAITON (MISC-DR, MISC-CR)
6} SUBMIT MISC-DR, IN MSCDR, OUT $S, WAITON MISC-CR
7} SUBMIT MISC-CR, IN MSCCR, OUT $S, HOLD ON
8} ALTER SCHEDULER, DEFAULT-OUT $S.#ACCNTS
9} ALTER MISC-CR, HOLD OFF
```

- To display the latest line from the buffer:

```
10} ?
10} ALTER MISC-CR, HOLD OFF
10}
```

- To display command line 5 from the buffer:

```
10} ? 5
10} SUBMIT TRIAL-BAL, IN TB, OUT $S, WAITON (MISC-DR, MISC
-CR)
10}
```

- To display the second command line preceding the current command line:

```
10} ? -2
10} ALTER SCHEDULER, DEFAULT-OUT $S.#ACCNTS
10}
```

- To display the latest command line that begins with ST:

```
10} ? ST
10} STATUS SCHEDULER
10}
SCHEDULER STATUS
Process : \MELBQAT.$SCHD Primary : 0,97 Backup : 1,101
Database: $DATA7.SCHD
Logfile : $DATA7.SCHD.LOGAAC
Time : 25MAR93 09:47:18 .
.
11}
```

  This example also shows the effect (command reexecution) of pressing RETURN after BATCHCOM displays the command.

7 Attributes

This section describes the syntax, operation, and results of all attachment-set, class, execurot, job, and scheduler attributes:

Topic	Page
Attribute Reference Summary	7-1
Attribute Descriptions	7-8

For information on abbreviating attribute keywords and a list of keyword aliases, see Keywords on page 6-7.

Attribute Reference Summary

This subsection is a quick-reference guide to the syntax of attachment-set, class, executor, job, and scheduler attributes.

Attachment-Set Attributes

- **ASSIGN** specifies the name and attributes of an ASSIGN:

  ```
  ( ASSI[GN] ASSIGN-name , ASSIGN-attributes )
  ```

- **DEFINE** specifies the name and attributes of a DEFINE:

  ```
  ( DEFI[NE] DEFINE-name-1 , [ LIK[E] DEFINE-name-2 , ] [ CLASS
  DEFINE-class , ]
  DEFINE-attribute [ , DEFINE-attribute ]… )
  ```

- **PARAM** specifies the name and value of a PARAM:

  ```
  ( PA[RAM] PARAM-name PARAM-value )
  ```

- **SECURITY** controls user access to an attachment set:

  ```
  SEC[URITY] " security "
  ```

- **TEMPORARY** determines whether the scheduler automatically deletes an attachment set not assigned to a job:

  ```
  TEM[PORARY] {OF[F] | ON }
  ```

Class Attribute

- **INITIATION** specifies whether jobs from the class are available for execution by the scheduler:

  ```
  INI[TIATION] {OF[F] | ON }
  ```

Executor Attributes

- **CLASS** specifies an executor's classes:

  ```
  CL[ASS] { class-name | ( class-name [ , class-name ]… ) | *)
  }
  ```

- **CPU** assigns an executor to a CPU:

  ```
  CP[U] cpu-number
  ```

Job Attributes

- **AFTER** specifies the date and time after which a job becomes eligible for execution. The attribute also enables you to change a nonexecuting job's AT attribute to AFTER.

  ```
  AF[TER] [ date ] [ time ]
  ```

- **AT** specifies the date and time at which the scheduler is to create a temporary executor for and execute a job:

  ```
  AT [ date ] [ time ]
  ```

- **ATTACHMENT-SET** assigns up to three attachment sets to a job or dissociates attachment sets from a job:

  ```
  A[TTACHMENT]-S[ET] { attachment-set | ( attachment-set [ ,
  attachment-set ]… ) }
  ```

- **CALENDAR** specifies the name of the BATCHCAL file containing a job's run calendar. You can remove the attribute from the job by specifying CALENDAR without an accompanying file name.

  ```
  CA[LENDAR] [ file-name ]
  ```

- **CLASS** specifies a job's class:

  ```
  CL[ASS] class-name
  ```

- **DESCRIPTION** contains text that describes a job:

  ```
  DES[CRIPTION] " [ string ] "
  ```

- **EVERY** specifies that job execution occur at regular, specified intervals:

  ```
  EV[ERY] [ weeks WEEK[S]
  days D[AYS]
  hours [ : mins ] [HOURS]
  hours H[OURS] [ mins MIN[UTES] ]
  crontab-entry) ]
  ```

  crontab-entry is this five-field entry:

  ```
     minute hour day-of-month month day-of-week
  ```

- **EXECUTOR-PROGRAM** specifies the program file of the program the scheduler starts as the initial process of a job:

  `E[XECUTOR]-P[ROGRAM] file-name`

- **EXTSWAP** specifies the name of the swap file for the default extended data segment of a job's executor-program process:

  `EXT[SWAP] { $volume-name | file-name }`

- **HIGHPIN** determines whether a job's executor-program process runs at a low PIN or at a high PIN (available only in D20 or later versions of the NetBatch product):

  `HIG[HPIN] { OF[F] | ON }`

- **HOLD** determines whether a job is available for execution:

  `HOLD { OF[F] | ON }`

- **HOLDAFTER** determines whether the scheduler puts a job on hold when the job finishes executing:

  `HOLDA[FTER] { OF[F] | ON }`

- **IFFAILS** determines whether the scheduler reschedules a recurrent job that fails during execution. The attribute works in combination with the job's RESTART, STALL, and STOP-ON-ABEND attributes.

  `IF[FAILS] { OF[F] | ON }`

- **IN** specifies the name of a job's input file:

  `IN [ file-name ]`

- **JOB-LOG** directs log-file events for a job to a specified file:

  `J[OB]-L[OG] [ log-file ]`

- **JOBID-ZERO** determines whether the scheduler assigns a GMOMJOBID to a job's executor-program process:

  `J[OBID]-Z[ERO] { OF[F] | ON }`

- **LIB** specifies the name of the user library file for a job's executor program:

  `LIB [ file-name ]`

- **LIMIT** specifies an execution time limit for a job:

  `LIM[IT] hours [ : mins ]`

- **MAXPRINTLINES** specifies the maximum number of print lines for a job's spooler output file:

  `MAXPRINTL[INES] { number | NON[E] }`

- **MAXPRINTPAGES** specifies the maximum number of print pages for a job's spooler output file:

  ```
  MAXPRINTP[AGES] { number | NON[E] }
  ```

- **MEM** specifies the minimum number of 2048-byte memory pages allotted to a job's executor-program process for user data:

  ```
  ME[M] number
  ```

- **NAME** specifies a name for a job's executor-program process:

  ```
  NA[ME] $process-name
  ```

- **OUT** specifies the output file to which the scheduler writes data produced by an executing job:

  ```
  OU[T] [ file-name ]
  ```

- **PFS** specifies the size in bytes of a job's executor-program process file segment (PFS):

  ```
  PF[S] number
  ```

- **PRI** specifies the execution priority of a job's executor-program process:

  ```
  PR[I] number
  ```

- **PURGE-IN-FILE** determines whether the scheduler is to purge a job's input file when it deletes the job:

  ```
  P[URGE]-[IN]-[FILE] { OF[F] | ON
  ```

- **RESTART** determines whether the scheduler restarts a job that stops with completion code 7 (restart request sent to the scheduler) or terminates because of CPU failure. The attribute works in combination with the job's IFFAILS, STALL, and STOP-ON-ABEND attributes.

  ```
  REST[ART] { OF[F] | ON }
  ```

- **RUND** specifies whether a job's executor program enters the Guardian debug facility Debug or the Inspect interactive symbolic debugger when the program runs:

  ```
  RUND { OF[F] | ON }
  ```

- **SAVEABEND** specifies whether a job's executor-program process is to create a save file if the process traps or abends:

  ```
  SA[VEABEND] { OF[F] | ON }
  ```

- **SELPRI** specifies the selection priority of a job in its class:

  ```
  SEL[PRI] number
  ```

- **STALL** determines whether the scheduler puts in the SPECIAL-9 state a failed job it would otherwise reschedule or delete. The attribute works in combination with the job's IFFAILS, RESTART, and STOP-ON-ABEND attributes.

  ```
  STAL[L] { OF[F] | ON }
  ```

- **STARTUP** specifies one or more program parameters the scheduler sends a job's executor-program process in the startup message:

  ```
  STARTU[P] " param-set "
  ```

- **STOP-ON-ABEND** determines whether the scheduler stops a job and all its processes if any process of the job terminates because of CPU failure; abends with any completion code; or stops with completion code -3, -2, -1, 2, 3, 4, 5, or 6.

  ```
  S[TOP]-[ON]-[ABEND] { OF[F] | ON }
  ```

- **SWAP** specifies the name of the swap file for the user data stack segment of a job's executor-program process:

  ```
  SWA[P] { $volume-name | file-name }
  ```

- **TAPEDRIVES** specifies the number of tape drives required by a job:

  ```
  TA[PEDRIVES] number
  ```

- **TERM** specifies the home terminal of a job's executor-program process:

  ```
  TERM $process-name
  ```

- **VOLUME** specifies the default node, volume, and subvolume for unqualified file references in a job's input file. You also can use the attribute to specify the default security for disk files created by the job.

  ```
  V[OLUME] { \node. [ volume ] [ , " security " ]
  [ \node. ] volume [ , " security " ]
  [ \node. ] [ volume ] , " security ") }
  ```

- **WAIT** delays execution of a job for a specified period from the current time on the node where the job's scheduler is running:

  ```
  WAIT hours [ : mins ]
  ```

- **WAITON** specifies the names of up to 50 jobs on which execution of another job depends. You can remove the attribute from the job by specifying WAITON without any options.

  ```
  WAITO[N] [ job-name [ case ]
  ( job-name [ case ] [ , job-name [ case ] ]… ) ]
  ```

  case is one of:

  ```
  REL[EASE]
  STO[P]
  STO[P]-[ABEND]
  ```

Scheduler Attributes

- **AT-ALLOWED** determines whether users without execute access to the NETBATCH program file (that is, users who are not NetBatch supervisors) can submit jobs with the AT attribute:

  ```
  AT-A[LLOWED] { OF[F] | ON }
  ```

- **BACKUPCPU** specifies the preferred CPUs for a scheduler's backup process:

  ```
  B[ACKUPCPU] { cpu-number-1 [ , cpu-number-2 ] | * }
  ```

- **CATCHUP** determines whether jobs with the EVERY attribute accumulate run backlogs:

  ```
  CAT[CHUP] { OF[F] | ON }
  ```

- **DEFAULT-CLASS** specifies the class of a job submitted without the CLASS job attribute. It also specifies the class of an executor added without the CLASS executor attribute.

  ```
  D[EFAULT]-C[LASS] class-name
  ```

- **DEFAULT-EXECUTOR-PROGRAM** specifies the executor program of a job submitted without the EXECUTOR-PROGRAM attribute:

  ```
  D[EFAULT]-E[XECUTOR]-[PROGRAM] file-name
  ```

- **DEFAULT-HIGHPIN** specifies the HIGHPIN attribute of a job submitted without that attribute (available only in D20 or later versions of the NetBatch product):

  ```
  D[EFAULT]-H[IGHPIN] { OF[F] | ON }
  ```

- **DEFAULT-MAXPRINTLINES** specifies the maximum number of output-file print lines of a job submitted without the MAXPRINTLINES attribute:

  ```
  D[EFAULT]-MAXPRINTL[INES] { number | NON[E] }
  ```

- **DEFAULT-MAXPRINTPAGES** specifies the maximum number of output-file print pages of a job submitted without the MAXPRINTPAGES attribute:

  ```
  D[EFAULT]-MAXPRINTP[AGES] { number | NON[E] }
  ```

- **DEFAULT-OUT** specifies the output file of a job submitted without the OUT attribute:

  ```
  D[EFAULT]-O[UT] file-name
  ```

- **DEFAULT-PRI** specifies the execution priority of the executor-program process of a job submitted without the PRI attribute:

  ```
  D[EFAULT]-P[RI] number
  ```

- **DEFAULT-SELPRI** specifies the selection priority of a job submitted without the SELPRI attribute:

  ```
  D[EFAULT]-SE[LPRI] number
  ```

- **DEFAULT-STALL** specifies the STALL attribute of a job submitted without that attribute:

  ```
  D[EFAULT]-ST[ALL] { OF[F] | ON }
  ```

- **DEFAULT-STOP-ON-ABEND** specifies the STOP-ON-ABEND attribute of a job submitted without that attribute:

  ```
  D[EFAULT]-[STOP]-[ON]-[ABEND] { OF[F] | ON }
  ```

- **EMS** enables or disables event-message generation while the scheduler is running:

  ```
  EM[S] { ER[RORS] | OF[F] | ON }
  ```

- **INITIATION** enables or disables job startup by the scheduler:

  ```
  INI[TIATION] { OF[F] | ON }
  ```

- **LOCALNAMES** makes the scheduler treat jobs submitted from licensed requesters on the specified nodes as local jobs, not as remote jobs:

  ```
  LO[CALNAMES] [ \remote-node
  ( \remote-node [ , \remote-node ]… ) ]
  ```

- **MAX-CONCURRENT-JOBS** specifies the maximum number of concurrent jobs and temporary executors for the scheduler:

  ```
  M[AX]-[CONCURRENT]-[JOBS] max-concurrent-jobs
  [ , max-temporary-executors ]
  ```

- **MAX-PRI** specifies an upper execution-priority limit for executor-program processes:

  ```
  M[AX]-[PRI] number
  ```

- **SUBMIT-ALLOWED** allows or disallows job submission by permitting or preventing use of the SUBMIT JOB command:

  ```
  S[UBMIT]-A[LLOWED] { OF[F] | ON }
  ```

- **TAPEDRIVES** specifies the number of tape drives available for use by jobs:

  ```
  TA[PEDRIVES] number
  ```

Attribute Descriptions

This subsection contains descriptions and examples of the syntax, operation, and results of all attributes, in alphabetical order by object type:

- Attachment-set attributes:

  ASSIGN DEFINE PARAM SECURITY TEMPORARY

- Class attribute:

  INITIATION

- Executor attributes:

  CLASS CPU

- Job attributes:

AFTER	IN	RESTART
AT	JOB-ZERO	RUND
ATTACHMENT-SET	JOBID-ZERO	SAVEABEND
CALENDAR	LIB	SELPRI
CLASS	LIMIT	STALL
DESCRIPTION	MAXPRINTLINES	STARTUP
EVERY	MAXPRINTPAGES	STOP-ON-ABEND
EXECUTOR-PROGRAM	MEM	SWAP
EXTSWAP	NAME	TAPEDRIVES
HIGHPIN	OUT	TERM
HOLD	PFS	VOLUME
HOLDAFTER	PRI	WAIT
IFFAILS	PURGE-IN-FILE	WAITON

- Scheduler attributes:

AT-ALLOWED	DEFAULT-SELPRI
BACKUPCPU	DEFAULT-STALL
CATCHUP	DEFAULT-STOP-ON-ABEND
DEFAULT-CLASS	EMS
DEFAULT-EXECUTOR-PROGRAM	INITIATION
DEFAULT-HIGHPIN	LOCALNAMES
DEFAULT-MAXPRINTLINES	MAX-CONCURRENT-JOBS
DEFAULT-MAXPRINTPAGES	MAX-PRI
DEFAULT-OUT	SUBMIT-ALLOWED
DEFAULT-PRI	TAPEDRIVE

AFTER Job Attribute

The AFTER job attribute specifies the date and time after which a job becomes eligible for execution. The attribute also lets you change a nonexecuting job's AT attribute to AFTER.

```
AF[TER] [ date ] [ time ]
```

date

is the date on which the job becomes eligible for execution. You can enter the date in any of these forms:

```
[ d] d mmm
[ d] d mmm [ yy] yy
[ yy] yy [ m] m [ d] d
[ yy] yy mmm [ d] d
[ yy] yymmdd
mmm [ d] d
mmm [ d] d [ yy] yy
day
```

[d] d

is a number in the range 1 through 31 specifying the day of the month.

dd

is a two-digit number in the range 01 through 31 specifying the day of the month.

[m] m

is a number in the range 1 through 12 specifying the month.

mm

is a two-digit number in the range 01 through 12 specifying the month.

mmm

is a character string specifying the month. The options are:

JAN[UARY]	APR[IL]	JUL[Y]	OCT[OBER]
FEB[RUARY]	MAY	AUG[UST]	NOV[EMBER]
MAR[CH]	JUN[E]	SEP[TEMBER]	DEC[EMBER]

[yy] yy

is a two-digit or four-digit number specifying the year. When you specify a two-digit number, BATCHCOM adds 1900 to that number to calculate the year. If the sum is less than the current year minus 25, BATCHCOM interprets the year

as the sum plus 100. For example, in 1993, BATCHCOM interprets 05 as the year 2005, 67 as 2067, and 94 as 1994.

day

is a day of the week. `day` causes BATCHCOM to generate an AFTER date of the next day whose name matches `day`. The options are:

```
MON[DAY]   WED[NESDAY]   FRI[DAY]      SUN[DAY]
TUE[SDAY]  THU[RSDAY]    SAT[URDAY]
```

[*d*] *d mmm* and *mmm* [*d*] *d* refer to the current year. To use one of these date forms and specify *time*, *time* must appear before *date*, not after.

For spaces in a date, you can substitute slashes (/), periods (.), or hyphens (-). You can omit spaces if they separate alphabetic and numeric date components (for example, you can enter 1993JUL04 instead of 1993 JUL 04). You cannot leave out spaces between numeric date components (for example, you can enter 1993 7 04, but not 1993704).

The current date applies if you omit `date`. (The current date is the system date on the node where the scheduler is running.)

time

is the time after which the job becomes eligible for execution. You can enter the time in any of these forms:

```
[ h] h:[ m] m[:[ s] s] [ A[M] | P[M] ]
[ MIDD[AY] | NOO[N] ]
MIDN[IGHT]
```

`[ h] h`

is a number in the range 0 through 23 specifying the hour of the day.

`[ m] m`

is a number in the range 0 through 59 specifying the minute of the hour.

`[ s] s`

is a number in the range 0 through 59 specifying seconds.

`[ A[M] | P[M] ]`

specifies, for a time entered in 12-hour format, whether that time occurs before midday (AM) or after midday (PM).

`[ MIDD[AY] | NOO[N] ]`

specifies the middle of the day (specifically, 12 o’clock in the day).

`MIDN[IGHT]`

specifies the middle of the night (specifically, 12 o'clock at night).

The time 00:00:00 applies if you omit *`time`*.

Considerations

- The scheduler treats a job submitted without the AFTER, AT, or WAIT attributes like a job with the attribute WAIT 0:0. Such a job is eligible to run immediately on submission unless delayed by another attribute such as HOLD or TAPEDRIVES.
- The AFTER, AT, and WAIT job attributes are mutually exclusive. A job can have any one of the attributes but not two or all three. By way of illustration, consider job X, whose attributes include AFTER 17:00. The job runs at 1615 if altered at 1610 by the command ALTER JOB X, WAIT 00:05. (If you specify more than one of AFTER, AT, and WAIT in a command, the last attribute specified takes precedence. For example, job Y submitted at 0900 by the command SUBMIT JOB Y, AFTER 09:45, WAIT 00:15 runs at 0915.)
- You can display the run time of a job by specifying AFTER or AT in an INFO JOB command. For example:

```
8} INFO JOB X, AFTER
JOB ATTRIBUTES for X
jobnumber: 26
after: 14DEC94 23:00:00
next-runtime: 14DEC94 23:00:00
```

  If the job's state is SPECIAL- *n*, BATCHCOM does not display the run time.

- To change a nonexecuting job's AT attribute to AFTER, use the ALTER JOB command to specify AFTER without *`date`* and *`time`*. (Omitting *`date`* and *`time`* leaves intact the original *`date`* and *`time`*.) For example:

```
21} SUBMIT JOB X, AT 05DEC94 23:59
Job X Jobnumber 22 submitted
22} INFO JOB X, AT, AFTER
JOB ATTRIBUTES for X
jobnumber: 22
at: 05DEC94 23:59:00
next-runtime: 05DEC94 23:59:00
23} ALTER JOB X, AFTER
Job X Jobnumber 22 altered
24} INFO JOB X, AT, AFTER
JOB ATTRIBUTES for X
jobnumber: 22
after: 05DEC94 23:59:00
next-runtime: 05DEC94 23:59:00
```

Example

This example shows the AFTER attribute of a job specifying the time after which the job becomes eligible for execution:

```
25} SUBMIT JOB YEAR-END, AFTER 31DEC94 MIDDAY
Job YEAR-END Jobnumber 23 submitted
26} STATUS JOB YEAR-END
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
23 YEAR-END 255,255 31DEC94 DEFAULT
27} INFO JOB YEAR-END, AFTER
JOB ATTRIBUTES for YEAR-END
jobnumber: 23
after: 31DEC94 12:00:00
next-runtime: 31DEC94 12:00:00
The job submitted on Tuesday, September 27, in this example
 will become eligible for
execution at the start of Monday, October 3:
11> TIME
September 27, 1994 16:30:35
12> BATCHCOM $ZBAT; SUBMIT JOB MONDAY-MORNING, AFTER MONDAY
Job MONDAY-MORNING Jobnumber 24 submitted
13> BATCHCOM $ZBAT; INFO JOB MONDAY-MORNING, AFTER
JOB ATTRIBUTES for MONDAY-MORNING
jobnumber: 24
after: 03OCT94 00:00:00
next-runtime: 03OCT94 00:00:00
```

ASSIGN Attachment-Set Attribute

The ASSIGN attachment-set attribute specifies the name and attributes of an ASSIGN. (An ASSIGN is a parameter that assigns the name of an actual file to a logical file name in a program. It also can specify the file's creation and open attributes.) The scheduler passes the ASSIGN to the executor-program process of a job using the attachment set when the job starts.

```
( ASSI[GN] ASSIGN-name , ASSIGN-attributes )
```

ASSIGN-name

is the name of an ASSIGN. The name can contain from 1 through 31 letters, numbers, hyphens, and circumflexes. The name cannot contain spaces.

Note. To avoid confusion, do not use the attribute keywords ASSIGN, DEFINE, and PARAM as ASSIGN names.

ASSIGN-attributes

specifies ASSIGN attributes in this form. For descriptions of the options for each syntax item, see the description of the ASSIGN command in the*TACL Reference Manual*.

logical-unit , *actual-file-name* [, *create-open-spec*]…

logical-unit

is the name to use as a substitute for the actual file name, specified in one of these forms:

```
[ logical-file
program-unit.logical-file
*.logical-file ]
```

actual-file-name

is the name of the actual file to which *logical-unit* refers.

create-open-spec

specifies one of these file creation and open attributes:

```
B[LOCK] block-size
C[ODE] file-code [ EXCL[USIVE]
S[HARED]
P[ROTECTED] ]
E[XT] ( pri-extent-size [ , sec-extent-size ] )
[ IN[PUT]
O[UTPUT]
I-[O] ]
R[EC] record-size
```

Consideration

For more information about ASSIGNs, see the *TACL Reference Manual* and the *TACL Programming Guide*.

Examples

- This example shows the scheduler passing ASSIGNs from a job's attachment set to the job's TACL executor-program process:

```
> LOGON FPP.QA, psswrd
> FUP COPY INFILE
ASSIGN
> BATCHCOM $ZBAT; ADD ATTACHMENT-SET B, (ASSIGN FILE-A,
$NB.NEWFILE.A, EXCLUSIVE, INPUT, EXT (100, 200), CODE
789, REC 1024, BLOCK 4096), (ASSIGN DAILY-LOG,
$DATA7.LOGS.DAYLOG)
Attachment-set (FPP.QA)B added
> BATCHCOM $ZBAT; SUBMIT JOB Y, EXECUTOR-PROGRAM TACL, IN
INFILE, OUT OUTFILE, ATTACHMENT-SET B
```

```
Job Y job number 270 submitted
> FUP COPY OUTFILE
TACL (T9205Dxx - DDMMMYY), Operating System Dxx .
.
ASSIGN
DAILY-LOG
Physical file: $DATA7.LOGS.DAYLOG
FILE-A
Physical file: $NB.NEWFILE.A
Primary extent: 100
Secondary extent: 200
File code: 789
Exclusion: EXCLUSIVE
Access: INPUT
Record: 1024
Block: 4096
```

- This example shows various attachment-set commands adding, altering, and displaying an attachment set's ASSIGN attributes. The TACL environment is the source of ASSIGNs ACCNTS-RCVBL and ACCNTS-PYBL.

```
> LOGON SUPER.FPP, psswrd
> ASSIGN ACCNTS-RCVBL, $A.ACCOUNTS.RCVBL
> ASSIGN ACCNTS-PYBL, $A.ACCOUNTS.PYBL
> ASSIGN
ACCNTS-RCVBL
Physical file: $A.ACCOUNTS.RCVBL
ACCNTS-PYBL
Physical file: $A.ACCOUNTS.PYBL
> BATCHCOM $ZBAT
1} SHOW ATTACHMENT-SET ASSIGN
ATTACHMENT-SET ATTRIBUTES
attachments: ASSIGN ACCNTS-PYBL, $A.ACCOUNTS.PYBL
ASSIGN ACCNTS-RCVBL, $A.ACCOUNTS.RCVBL
2} SET ATTACHMENT-SET (ASSIGN ACCNTS-JRNL, $A.ACCOUNTS.JRNL,
PROTECTED, I-O, EXT (256, 512), CODE 905, REC 512, BLOCK
2048)
3} ADD ATTACHMENT-SET ACCOUNTS, (ASSIGN ACCNTS-GLDGR,
$A.ACCOUNTS.GLDGR)
Attachment-set (SUPER.FPP)ACCOUNTS added
4} INFO ATTACHMENT-SET ACCOUNTS, ASSIGN *
ATTACHMENT-SET ATTRIBUTES for (SUPER.FPP)ACCOUNTS
attachments: ASSIGN ACCNTS-GLDGR, $A.ACCOUNTS.GLDGR
ASSIGN ACCNTS-JRNL, $A.ACCOUNTS.JRNL,
PROTECTED, I-O, EXT (256,512), CODE
905, REC 512, BLOCK 2048
ASSIGN ACCNTS-PYBL, $A.ACCOUNTS.PYBL
ASSIGN ACCNTS-RCVBL, $A.ACCOUNTS.RCVBL
5} ALTER ATTACHMENT-SET ACCOUNTS, (ASSIGN ACCNTS-PYBL,
$DATA7.ADMIN.PYBL)
Attachment-set (SUPER.FPP)ACCOUNTS altered
7} DELETE ATTACHMENT-SET ACCOUNTS, ASSIGN ACCNTS-RCVBL
Attachment-set (SUPER.FPP)ACCOUNTS altered
8} INFO ATTACHMENT-SET ACCOUNTS, ASSIGN *
ATTACHMENT-SET ATTRIBUTES for (SUPER.FPP)ACCOUNTS
```

```
attachments: ASSIGN ACCNTS-GLDGR, $A.ACCOUNTS.GLDGR
ASSIGN ACCNTS-JRNL, $A.ACCOUNTS.JRNL,
PROTECTED, I-O, EXT (256,512), CODE
905, REC 512, BLOCK 2048
ASSIGN ACCNTS-PYBL, $DATA7.ADMIN.PYBL
```

AT Job Attribute

The AT job attribute specifies the date and time at which the scheduler is to execute a job. The attribute makes the scheduler create a temporary executor for the job.

```
AT [ date ] [ time ]
```

date

is the date on which the scheduler executes the job. You can enter the date in any of these forms. (For descriptions of the date forms, see AFTER Job Attribute on page 7-9.)

```
[ d] d mmm
[ d] d mmm [ yy] yy
[ yy] yy [ m] m [ d] d
[ yy] yy mmm [ d] d
[ yy] yymmdd
mmm [ d] d
mmm [ d] d [ yy] yy
day
```

[*d*] *d mmm* and *mmm* [*d*] *d* refer to the current year. To use one of these date forms and specify *time*, *time* must appear before *date*, not after. The current date applies if you omit *date*. (The current date is the system date on the node where the scheduler is running.)

time

is the time at which the scheduler will execute the job. You can enter the time in any of these forms. (For descriptions of the time forms, see AFTER Job Attribute on page 7-9.)

```
[ h] h:[ m] m[:[ s] s] [ A[M] | P[M] ]
[ MIDD[AY] | NOO[N] ]
MIDN[IGHT]
```

The time 00:00:00 applies if you omit *time*.

Considerations

- The scheduler treats a job submitted without the AFTER, AT, or WAIT attributes like a job with the attribute WAIT 0:0. Such a job is eligible to run immediately on submission unless delayed by another attribute such as HOLD or TAPEDRIVES.

- Users without execute access to the NETBATCH program file (that is, users who are not NetBatch supervisors) cannot submit jobs with the AT attribute when the scheduler's AT-ALLOWED attribute is set to OFF (the default value). For more information, see AT-ALLOWED Scheduler Attribute on page 7-18.
- The scheduler creates temporary executors for all jobs with the AT attribute and for all jobs that are operated on by the RUNNOW JOB command. The scheduler deletes the executors when the jobs finish.

  A temporary executor has a scheduler-assigned name of the form `__TEMP_EXEC_`*`job-number`*, where *`job-number`* is the number of the job using the executor; for example, __TEMP_EXEC_496. The scheduler selects the CPU of a temporary executor from available CPUs on the scheduler's node.
- A job with the AT attribute does not run if any of these conditions exists:
  - The job has the TAPEDRIVES attribute and requires more drives than are available. For more information about this attribute, see TAPEDRIVES Job Attribute on page 7-109 and TAPEDRIVES Scheduler Attribute on page 7-110.
  - The job's class has the attribute INITIATION OFF. This attribute prevents jobs belonging to the class from running. To make the jobs available for execution, change the attribute to INITIATION ON by using the ALTER CLASS command.
  - The scheduler would exceed its temporary-executors limit by running the job.

    In this case, the job runs when the scheduler can create a temporary executor without exceeding the limit. For more information, see MAX-CONCURRENT-JOBS Scheduler Attribute on page 7-79.
- The AFTER, AT, and WAIT job attributes are mutually exclusive. A job can have any one of the attributes, but not two or three. Consider job X, whose attributes include AFTER 17:00. The job runs at 1615 if altered at 1610 by the command ALTER JOB X, WAIT 00:05. (If you specify more than one of AFTER, AT, and WAIT in a command, the last attribute specified takes precedence. For example, job Y submitted at 0900 by the command SUBMIT JOB Y, AFTER 09:45, WAIT 00:15 runs at 0915.)
- To display the run time of a job, specify AFTER or AT in an INFO JOB command. For example:

```
8} INFO JOB X, AFTER
JOB ATTRIBUTES for X
jobnumber: 26
after: 14DEC94 23:00:00
next-runtime: 14DEC94 23:00:00
```

  BATCHCOM does not display the run time if the job's state is SPECIAL-*n*.

- To change a nonexecuting job's AT attribute to \AFTER, use the ALTER JOB command to specify AFTER without *date* and *time*. (Omitting *date* and *time* leaves intact the original *date* and *time*.) For example:

```
21} SUBMIT JOB X, AT 05DEC94 23:59
Job X Jobnumber 22 submitted
22} INFO JOB X, AT, AFTER
JOB ATTRIBUTES for X
jobnumber: 22
at: 05DEC94 23:59:00
next-runtime: 05DEC94 23:59:00
23} ALTER JOB X, AFTER
Job X Jobnumber 22 altered
24} INFO JOB X, AT, AFTER
JOB ATTRIBUTES for X
jobnumber: 22
after: 05DEC94 23:59:00
next-runtime: 05DEC94 23:59:00
```

Example

This example shows the submission and execution of a job with the AT attribute. The scheduler creates a temporary executor for the job, then deletes the executor when the job finishes.

```
16} STATUS EXECUTOR *
2117-I No EXECUTOR selected
17} SUBMIT JOB, EXECUTOR-PROGRAM DELAY, STARTUP "3 MINS", AT
Job ZBAT-0006 Jobnumber 6 submitted
18} STATUS JOB 6
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
6 ZBAT-0006 205,70 563 EXECUTING DEFAULT
19} STATUS EXECUTOR *
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
----------------------- --- ------ ---- -------
__TEMP_EXEC_6 2 DELETE 6 DEFAULT
20} STATUS JOB 6
2099-E JOB does not exist
|21} STATUS EXECUTOR *
2117-I No EXECUTOR selected
```

AT-ALLOWED Scheduler Attribute

The AT-ALLOWED scheduler attribute determines whether users without execute access to the NETBATCH program file (that is, users who are not NetBatch supervisors) can submit jobs with the AT attribute. (Jobs with the AT attribute create temporary executors that can overload your system. Preventing nonprivileged users from using the attribute helps you prevent system overload.)

```
AT-A[LLOWED] { OF[F] | ON
```

`OFF`

prevents users who are not NetBatch supervisors from submitting jobs with the AT attribute. BATCHCOM displays this message when a scheduler with the AT-ALLOWED OFF attribute rejects a submitted job:

```
2056-E AT-ALLOWED is currently OFF; submit AFTER time
```

`ON`

allows all users to submit jobs with the AT attribute.

Considerations

- A scheduler adopts the attribute AT-ALLOWED OFF by default when cold started.
- The AT-ALLOWED OFF attribute only affects submission of jobs with the AT attribute. It does not affect other scheduler operations. (Changing AT-ALLOWED from ON to OFF does not prevent the scheduler from starting jobs submitted with the AT attribute before the change.)

Example

This example shows the effect of the AT-ALLOWED attribute when set to OFF. The attribute prevents a user who is not a NetBatch supervisor from submitting a job with the AT attribute but lets a NetBatch supervisor submit the job.

```
> LOGON 205,70, psswrd
> FILEINFO NETBATCH
$A.MAN13
Code … Owner RWEP …
NETBATCH O 100L … 255,255 "AOGO" …
> BATCHCOM $ZBAT; INFO SCHEDULER, AT-ALLOWED
SCHEDULER ATTRIBUTES
at-allowed: Off
> BATCHCOM $ZBAT; SUBMIT JOB, AT 12:00
2056-E AT-ALLOWED is currently OFF; submit AFTER time
> LOGON 255,205, psswrd
> BATCHCOM $ZBAT; SUBMIT JOB, AT 12:00
Job ZBAT-0001 Jobnumber 1 submitted
```

ATTACHMENT-SET Job Attribute

The ATTACHMENT-SET job attribute assigns up to three attachment sets to a job. You also can use the attribute to dissociate attachment sets from a job.

```
A[TTACHMENT]-S[ET] [ attachment-set
( attachment-set [ attachment-set ]… ) ]
```

attachment-set

specifies an attachment set in one of these forms:

[(*user-ID*)] *attachment-set-ID*

user-ID

specifies the user ID of the attachment-set owner. (*user-ID* must be in *group-name.user-name* or *group-ID*,*user-ID* form.) The default is the user ID of the current user.

attachment-set-ID

is one of:

attachment-set-name

specifies an attachment-set name.

attachment-set-number

specifies a scheduler-generated number identifying an attachment set created by means of the #CURRENT variable.

#CURRENT

is the attachment set specified by the #CURRENT variable. (To display the value of this variable, use the command INFO ATTACHMENT-SET #CURRENT, DETAIL. When the variable has a null value, an “undefined substitution” message appears.) If the variable has a null value, the scheduler creates an attachment set with these attributes:

Attribute	Value
ASSIGN	ASSIGN attributes from working-attributes set
DEFINE	DEFINE attributes from working-attributes set
PARAM	PARAM attributes from working-attributes set
SECURITY	UUUU unless overridden by SECURITY attribute in working-attributes set
TEMPORARY	ON unless overridden by TEMPORARY attribute in working-attributes set

Considerations

- You must have execute access to the attachment set you want to assign to a job. For example, NB.USER can assign set (SUPER.SUPER)ADP2 to a job if the set's security is "OAAA" but not if the security is "OAGA."
- A job can have up to three attachment sets. The order in which you specify the sets is the order in which the scheduler supplies them to the job. For example, specifying sets C, B, and A in that order makes the scheduler process set C first, set B second, and set A third.

  If the name of an ASSIGN, DEFINE, or PARAM from a set conflicts with a name from a set specified earlier, the scheduler overwrites the earlier ASSIGN, DEFINE, or PARAM with the details of the later ASSIGN, DEFINE, or PARAM.
- To dissociate an attachment set from a job, use the ALTER JOB command to set the job's ATTACHMENT-SET attribute to the names of the attachment sets that are to remain assigned to the job. To dissociate all attachment sets from the job, use the ALTER JOB command to specify ATTACHMENT-SET without any options.
- BATCHCOM maintains the #CURRENT variable as follows:
  - Sets the variable's value on execution of an ADD or ALTER attachment-set command to the set specified by the command. For example:

    ```
    2} INFO ATTACHMENT-SET #CURRENT
    -^-0345-
    E Undefined substitution
    3} ADD ATTACHMENT-SET (NB.USER)S1; INFO ADP #CURRENT
    Attachment-set (NB.USER)S1 added
    ATTACHMENT-SET ATTRIBUTES for (NB.USER)S1
    security: "UUUU"
    temporary: Off
    4} ALTER ATTACHMENT-SET (NB.USER)XYZ, SECURITY "AAAA"
    Attachment-set (NB.USER)XYZ altered
    5} INFO ATTACHMENT-SET #CURRENT
    ATTACHMENT-SET ATTRIBUTES for (NB.USER)XYZ
    security: "AAAA"
    temporary: Off
    ```
  - Sets the variable's value to null on execution of a RESET or SET attachment-set command or an OPEN scheduler command. For example:

    ```
    9} INFO ATTACHMENT-SET #CURRENT
    ATTACHMENT-SET ATTRIBUTES for (SUPER.FPP)DATE
    security: "NNNN"
    temporary: Off
    10} OPEN $SCHD
    NETBATCH SERVER - T9190D20 …
    11} INFO ATTACHMENT-SET #CURRENT
    -^-0345-
    E Undefined substitution
    ```

Attachment-set commands that do not affect the #CURRENT variable are ASSUME, DELETE, INFO, SHOW, and STATUS.

- ADP (an abbreviation of ASSIGNs, DEFINEs, and PARAMs) is an alias of the attribute keyword ATTACHMENT-SET.

Examples

- This example shows the ATTACHMENT-SET attribute assigning, changing, and dissociating a job's attachment set:

```
37} SUBMIT JOB COMPILE-JOB, IN BUILD, OUT =OUT, HOLD ON,
ATTACHMENT-SET (NB.USER)COMPILE-C30
Job COMPILE-JOB job number 13 submitted
38} INFO JOB 13, ATTACHMENT-SET
JOB ATTRIBUTES for COMPILE-JOB
jobnumber: 13
attachment-set: (NB.USER)COMPILE-C30
39} ALTER JOB 13, ATTACHMENT-SET (SUPER.FPP)COMPILE-D20
Job COMPILE-JOB job number 13 altered
40} INFO JOB 13, ATTACHMENT-SET
JOB ATTRIBUTES for COMPILE-JOB
jobnumber: 13
attachment-set: (SUPER.FPP)COMPILE-D20
41} ALTER JOB 13, ATTACHMENT-SET
Job COMPILE-JOB job number 13 altered
42} INFO JOB 13, ATTACHMENT-SET
JOB ATTRIBUTES for COMPILE-JOB
jobnumber: 13
```

- This example shows a job adopting the #CURRENT attachment set:

```
49} CHANGEUSER NB.USER psswrd
49} INFO ATTACHMENT-SET #CURRENT, DETAIL
-^-0345-
E Undefined substitution
50} ALTER ATTACHMENT-SET COMPILE-C30, (ASSIGN TAL,
$SYSTEM.SYSTEM.TAL)
Attachment-set (NB.USER)COMPILE-C30 altered
51} INFO ATTACHMENT-SET #CURRENT, DETAIL
ATTACHMENT-SET ATTRIBUTES for (NB.USER)COMPILE-C30
security: "UUUU"
temporary: Off
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$DATA7.WORK
ASSIGN TAL, $SYSTEM.SYSTEM.TAL
52} SUBMIT JOB TAL-COMPILE, HOLD ON, ATTACHMENT-SET #CURRENT
Job TAL-COMPILE job number 14 submitted
53} INFO JOB 14, ATTACHMENT-SET
JOB ATTRIBUTES for TAL-COMPILE
jobnumber: 14
attachment-set: (NB.USER)COMPILE-C30
```

- This example shows the #CURRENT variable causing the scheduler to create an attachment set for a job. The scheduler creates the set because #CURRENT has a

null value (shown by the “undefined substitution” message). The set’s attributes come from the working-attributes set.

```
67} CHANGEUSER SUPER.FPP psswrd
67} SHOW ATTACHMENT-SET
ATTACHMENT-SET ATTRIBUTES
security: "NNNN"
temporary: Off
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.B
ASSIGN P, Q
PARAM X Y
68} INFO ATTACHMENT-SET #CURRENT, DETAIL
-^-0345-
E Undefined substitution
69} SUBMIT JOB TEST, HOLD ON, ATTACHMENT-SET #CURRENT
Attachment-set (SUPER.FPP)5 added
Job TEST job number 15 submitted
70} INFO JOB 15, ATTACHMENT-SET
JOB ATTRIBUTES for TEST
jobnumber: 15
attachment-set: (SUPER.FPP)5
71} INFO ATTACHMENT-SET 5, DETAIL
ATTACHMENT-SET ATTRIBUTES for (SUPER.FPP)5
security: "NNNN"
temporary: Off
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.B
ASSIGN P, Q
PARAM X Y
```

- This example first shows the propagation of ASSIGNs, DEFINEs, and PARAMs from the TACL working-attributes set to that of BATCHCOM. The example then shows the #CURRENT variable causing the automatic creation of a job’s attachment set. The ASSIGNs, DEFINEs, and PARAMs in this newly created set are those from BATCHCOM’s working-attributes set.

```
> LOGON SUPER.FPP, psswrd
> ASSIGN
A
Physical file: $DATA7.NB.A
B
Physical file: $BIG1.USERS.B
> INFO DEFINE **
Define Name =OUT
CLASS SPOOL
LOC \MELBQAT.$S
Define Name =_DEFAULTS
CLASS DEFAULTS
VOLUME $SYSTEM.NETBATCH
> PARAM
DAY .THURSDAY.
DATE .18NOV93.
> BATCHCOM $ZBAT
1} SHOW ATTACHMENT-SET
```

```
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =OUT, CLASS SPOOL, LOC \MELBQAT.$S
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$SYSTEM.NETBATCH
ASSIGN A, $DATA7.NB.A
ASSIGN B, $BIG1.USERS.B
PARAM DAY THURSDAY
PARAM DATE 18NOV93
2} INFO ATTACHMENT-SET #CURRENT, DETAIL
-^-0345-
E Undefined substitution
3} SUBMIT JOB X, HOLD ON, ATTACHMENT-SET #CURRENT
Attachment-set (SUPER.FPP)50 added
Job X job number 511 submitted
4} INFO JOB X, ATTACHMENT-SET
JOB ATTRIBUTES for X
jobnumber: 511
attachment-set: (SUPER.FPP)50
5} INFO ATTACHMENT-SET 50, DETAIL
ATTACHMENT-SET ATTRIBUTES for (SUPER.FPP)50
security: "UUUU"
temporary: On
attachments: DEFINE =OUT, CLASS SPOOL, LOC \MELBQAT.$S
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$SYSTEM.NETBATCH
ASSIGN A, $DATA7.NB.A
ASSIGN B, $BIG1.USERS.B
PARAM DAY THURSDAY
PARAM DATE 18NOV93
```

BACKUPCPU Scheduler Attribute

The BACKUPCPU scheduler attribute specifies the preferred CPUs for a scheduler's backup process. The scheduler creates the process initially in response to a START SCHEDULER command and re-creates it whenever the primary or backup process stops.

```
B[ACKUPCPU] { cpu-number-1 [ , cpu-number-2 ] | *
```

cpu-number-1

is the number of a CPU configured for the scheduler's node. The scheduler tries to create its backup process in this CPU, failing which it tries *cpu-number-2*. If *cpu-number-2* is not specified, the scheduler randomly selects any available CPU. *cpu-number-1* cannot specify the same CPU as *cpu-number-2*.

cpu-number-2

is the number of a CPU configured for the scheduler's node. The scheduler tries to create its backup process in this CPU after trying *cpu-number-1*. If *cpu-number-2* is not available, the scheduler randomly selects any available CPU.

cpu-number-2 cannot specify the same CPU as *cpu-number-1*.

*

specifies any available CPU on the scheduler's node. The scheduler selects the CPU at random.

Considerations

- The default value of the BACKUPCPU attribute when cold starting a scheduler is the number of the CPU of the scheduler's primary process.
- When *cpu-number-1* or *cpu-number-2* specifies a nonexistent CPU, the command fails with a message advising the allowable CPU number range for the node. For example:

  ```
  2063-E BACKUPCPU out of range 0 to 15, or * for any CPU
  ```

- Altering a scheduler's BACKUPCPU attribute with the ALTER SCHEDULER command alters the attribute's value in the scheduler database only. The command does not force an actual backup CPU change. (Such a change occurs only when the scheduler's primary or backup process stops.)
- The BACKUPCPU attribute has no effect on the result of a SWITCHCPU SCHEDULER command. (The command can only switch scheduler processes between the current primary and backup CPUs.)

Examples

- This example first shows the cold start of a scheduler that adopts the default BACKUPCPU attribute. It then shows the ALTER SCHEDULER command changing the attribute and the effect of the change when the backup restarts after stopping.

```
> NETBATCH /NAME $ZBAT, NOWAIT/ $DATA7.ZBAT !
> BATCHCOM $ZBAT; ADD SCHEDULER; START SCHEDULER
Scheduler added
Scheduler started
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 0,51 1,55 $X849
> BATCHCOM $ZBAT; INFO SCHEDULER, BACKUPCPU
SCHEDULER ATTRIBUTES
backupcpu: 0
> BATCHCOM $ZBAT; ALTER SCHEDULER, BACKUPCPU 0,2
Scheduler altered
> BATCHCOM $ZBAT; INFO SCHEDULER, BACKUPCPU
SCHEDULER ATTRIBUTES
backupcpu: 0,2
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 0,51 1,55 $X849
> STOP 1,55
> PPD $ZBAT
```

```
Name Primary Backup Ancestor
$ZBAT 0,51 2,58 $X849
```

- This example shows that the BACKUPCPU attribute has no effect on the result of a SWITCHCPU SCHEDULER command:

```
> BATCHCOM $ZBAT; INFO SCHEDULER, BACKUPCPU
SCHEDULER ATTRIBUTES
backupcpu: 1,3
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 0,51 2,58 $X849
> BATCHCOM $ZBAT; SWITCHCPU SCHEDULER
Scheduler CPUs switched
> PPD $ZBAT
Name Primary Backup Ancestor
$ZBAT 2,58 0,51 $X849
```

CALENDAR Job Attribute

The CALENDAR job attribute specifies the name of the BATCHCAL file containing a job's run calendar. The calendar lists the dates and times at which the scheduler automatically runs the job.

```
CA[LENDAR] [ file-name ]
```

file-name

is the name of a BATCHCAL-generated calendar file (file code 848). For information about generating a calendar file, see Section 5, Run Calendar Generation and Display.

BATCHCOM expands a partially qualified file name by using the defaults specified in the last VOLUME command. If there was no such command, the defaults current at the start of the session apply.

If you specify CALENDAR without *file-name* in an ALTER JOB command, the scheduler removes the CALENDAR attribute from the job. Specifying CALENDAR without *file-name* in a SET JOB command makes BATCHCOM remove the CALENDAR attribute from the working-attributes set.

Considerations

- A job submitted without the CALENDAR attribute can run immediately on submission unless delayed by another attribute. When execution finishes, the scheduler reschedules the job to run at the next future CALENDAR time.
- The NetBatch expression for a job with the CALENDAR or EVERY attribute is "recurrent job." These attributes cause the job to run repeatedly (recur) at a specified interval. Conversely, a nonrecurrent job is a job whose attributes do not include CALENDAR or EVERY.

- The scheduler accepts a job whose CALENDAR attribute specifies a nonexistent calendar file, but puts the job in the SPECIAL-7 state. For example:

```
5} SUBMIT JOB X, CALENDAR NOFILE
Job X job number 331 submitted
0531-W Error 11 opening CALENDAR file.
6} STATUS JOB X
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
331 X 205,70 7:CAL error DEFAULT
```

  Error 11 in message 0531-W is the file-system error “file not found.”

  To resolve the SPECIAL-7 state and make the job ready for execution:

  1. Generate the specified calendar file or change the CALENDAR attribute to specify an existing file.
  2. Change the job’s HOLD ON attribute to HOLD OFF by using the ALTER JOB command.

- The scheduler accepts a job whose run calendar contains no future times, but puts the job in the SPECIAL-8 state. (A future time is a time ahead of the current system date and time on the scheduler’s node.) For example:

```
7} SUBMIT JOB Y, CALENDAR OLDTIMES
Job Y job number 332 submitted
0532-W CALENDAR file has expired.
8} STATUS JOB Y
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
332 Y 205,70 8:CAL exprd DEFAULT
```

  To resolve the SPECIAL-8 state and make the job ready for execution:

  1. Regenerate the specified calendar file with future times or change the CALENDAR attribute to specify another file containing future times.
  2. Change the job’s HOLD ON attribute to HOLD OFF by using the ALTER JOB command.

- The scheduler rejects a job whose CALENDAR attribute specifies a file not generated by BATCHCAL. For example:

```
9} SUBMIT JOB Z, CALENDAR EDITFILE
2068-E CALENDAR file is not a valid calendar file; it must
be created by BATCHCAL
10} STATUS JOB Z
2099-E JOB does not exist
```

- A suspended job whose attributes include CALENDAR does not accumulate a run backlog while suspended. When you reactivate the job, it finishes executing but does not run again until the next future CALENDAR time. For more information about backlogs, see ACTIVATE JOB Command on page 6-32.

- The CALENDAR and EVERY job attributes are mutually exclusive:

- A job can have one of the attributes, but not both.
- CALENDAR overrides EVERY when assigned to a job whose attributes include EVERY, and EVERY overrides CALENDAR when assigned to a job whose attributes include CALENDAR.

- For information about how the scheduler treats a recurrent job that fails, see the descriptions of the IFFAILS Job Attribute on page 7-64, RESTART Job Attribute on page 7-95, STALL Job Attribute on page 7-102, and STOP-ON-ABEND Job Attribute on page 7-105.
- For information about recurrent jobs that are also dependent jobs, see WAITON Job Attribute on page 7-119.

Example

This example first shows the generation of a BATCHCAL calendar file from an EDIT source file. It then shows BATCHCAL displaying the next run time from the generated file. Finally, the example shows the submission and next run time of a job whose CALENDAR attribute specifies the BATCHCAL file.

```
> FILEINFO CALIN
$DATA7.NB
Code …
CALIN 101 …
> BATCHCAL /IN CALIN/
$DATA7.NB.CALOUT == Name of the generated calendar file
06:00 == Default start time
1993 * LAST == Run on last day of every month
- 1993 06 LAST == Not run on last day of June
1993 06 15 23:59 == Run on June 15 at 23:59
> FILEINFO CALOUT
$DATA7.NB
Code …
CALOUT 848 …
> BATCHCAL /IN CALOUT/ NEXT-DATE
1993-01-31 06:00:00
> BATCHCOM $ZBAT; SUBMIT JOB MONTH-END, CALENDAR CALOUT,
IFFAILS ON
Job MONTH-END job number 326 submitted
> BATCHCOM $ZBAT; INFO JOB MONTH-END, AFTER, CALENDAR,
IFFAILS
JOB ATTRIBUTES for MONTH-END
jobnumber: 326
calendar: \MELBDEV.$DATA7.NB.CALOUT
iffails: On
next-runtime: 31JAN93 06:00:00
```

CATCHUP Scheduler Attribute

The CATCHUP scheduler attribute determines whether jobs with the EVERY attribute accumulate run backlogs. Run backlogs can accumulate when the jobs are on hold or suspended for longer than the EVERY interval or run for longer than the interval.

```
CAT[CHUP] { OF[F] | ON }
```

OFF

prevents job run backlogs from accumulating. When jobs with the EVERY attribute are taken off hold, are reactivated, or terminate, the scheduler increments their next run times by the EVERY interval until the times are in the future.

ON

lets job run backlogs accumulate. When jobs with the EVERY attribute are taken off hold, are reactivated, or terminate, the scheduler runs them continuously until their next run times are in the future.

△ **Caution.** Changing a scheduler's CATCHUP attribute from ON to OFF does not prevent the scheduler from running jobs with run backlogs. In these circumstances, the scheduler runs each job with a backlog once at the next run time shown before recalculating the next run time to a future time. To prevent backlogs from ever accumulating:

- Set the CATCHUP attribute to OFF when you initialize the scheduler's database during a cold start.
- Avoid changing the value of the CATCHUP attribute when the scheduler is running.

Considerations

- A scheduler adopts the attribute CATCHUP ON by default when cold started.
- When the CATCHUP attribute is set to OFF and you warm start the scheduler, all jobs with the EVERY attribute start at the next future run time. This applies to jobs that are on hold (state SPECIAL-*n*) and that are later taken off hold by the ALTER JOB command. It also applies to jobs that remain suspended and are later reactivated.

Examples

- This example shows the cold start of a scheduler with the attribute CATCHUP OFF:

```
> NETBATCH /NAME $ZBAT, NOWAIT/ $DATA7.ZBAT !
> BATCHCOM $ZBAT; ADD SCHEDULER, CATCHUP OFF
Scheduler added
> BATCHCOM $ZBAT; START SCHEDULER
Scheduler started
```

- This example shows how a recurrent suspended job functions when reactivated and CATCHUP is set to ON. The job runs continuously until it clears the run backlog it accumulated while suspended, then it runs at the specified interval.

```
3} INFO SCHEDULER, CATCHUP
SCHEDULER ATTRIBUTES
catchup: On
4} SUBMIT JOB CATCHUP-YES, EXECUTOR-PROGRAM DELAY, STARTUP "1
MINS", EVERY 0:02 HOURS
Job CATCHUP-YES Jobnumber 21 submitted
5} SUSPEND JOB 21
Job CATCHUP-YES Jobnumber 21 suspended
6} ACTIVATE JOB 21
Job CATCHUP-YES Jobnumber 21 activated
7} STATUS JOB 21
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
21 CATCHUP-YES 255,255 4339 EXECUTING DEFAULT
8} STATUS-HISTORY 21, NB-LOG $TRASH.IJ30.LOGAAN
16SEP94 13:38:13 BEGIN JOB (SUPER.SUPER)CATCHUP-YES:1 …
16SEP94 13:38:15 LIST JOB CATCHUP-YES EXECUTING J_21 …
16SEP94 13:38:15 START EXECUTOR-PROGRAM U_255,255 J_21 …
16SEP94 13:38:34 SUSPND JOB CATCHUP-YES J_21 U_255,255 …
16SEP94 13:38:35 LIST JOB CATCHUP-YES SUSPENDED J_21 …
16SEP94 13:43:46 A'VATE JOB CATCHUP-YES J_21 U_255,255 …
16SEP94 13:43:46 LIST JOB CATCHUP-YES EXECUTING J_21 …
16SEP94 13:43:46 STOP CC_0 EXECUTOR-PROGRAM J_21 …
16SEP94 13:43:47 FINISH JOB CATCHUP-YES T_0:0:0:15 J_21 …
16SEP94 13:43:50 BEGIN JOB (SUPER.SUPER)CATCHUP-YES:2 … .
16SEP94 13:44:52 FINISH JOB CATCHUP-YES T_0:0:0:15 J_21 …
16SEP94 13:44:55 BEGIN JOB (SUPER.SUPER)CATCHUP-YES:3 … .
16SEP94 13:45:57 FINISH JOB CATCHUP-YES T_0:0:0:14 J_21 …
16SEP94 13:45:59 BEGIN JOB (SUPER.SUPER)CATCHUP-YES:4 … .
16SEP94 13:47:01 FINISH JOB CATCHUP-YES T_0:0:0:16 J_21 …
16SEP94 13:47:03 BEGIN JOB (SUPER.SUPER)CATCHUP-YES:5 … .
16SEP94 13:48:05 FINISH JOB CATCHUP-YES T_0:0:0:16 J_21 …
16SEP94 13:48:13 BEGIN JOB (SUPER.SUPER)CATCHUP-YES:6 … .
16SEP94 13:49:15 FINISH JOB CATCHUP-YES T_0:0:0:14 J_21 …
16SEP94 13:50:14 BEGIN JOB (SUPER.SUPER)CATCHUP-YES:7 … .
16SEP94 13:51:17 FINISH JOB CATCHUP-YES T_0:0:0:15 J_21 … .
```

- This example shows how a recurrent suspended job functions when reactivated and CATCHUP is set to OFF. When reactivated, the job finishes executing but does not run again until the next future run time. The job does not accumulate a run backlog while suspended.

```
4} INFO SCHEDULER, CATCHUP
SCHEDULER ATTRIBUTES
catchup: Off
5} SUBMIT JOB CATCHUP-NO, EXECUTOR-PROGRAM DELAY, STARTUP "1
MINS", EVERY 0:02 HOURS
Job CATCHUP-NO Jobnumber 2 submitted
6} SUSPEND JOB 2
Job CATCHUP-NO Jobnumber 2 suspended
7} ACTIVATE JOB 2
```

```
Job CATCHUP-NO Jobnumber 2 activated
8} STATUS JOB 2
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
2 CATCHUP-NO 255,255 4393 15:48:48 DEFAULT
9} STATUS-HISTORY 2, NB-LOG $DATA7.ZBAT.LOGAAI
16SEP94 15:42:51 BEGIN JOB (SUPER.SUPER)CATCHUP-NO:1 …
16SEP94 15:42:52 LIST JOB CATCHUP-NO EXECUTING J_2 …
16SEP94 15:42:52 START EXECUTOR-PROGRAM U_255,255 J_2 …
16SEP94 15:43:10 SUSPND JOB CATCHUP-NO J_2 U_255,255 …
16SEP94 15:43:10 LIST JOB CATCHUP-NO SUSPENDED J_2 …
16SEP94 15:47:23 A'VATE JOB CATCHUP-NO J_2 U_255,255 …
16SEP94 15:47:23 LIST JOB CATCHUP-NO EXECUTING J_2 …
16SEP94 15:47:23 STOP CC_0 EXECUTOR-PROGRAM J_2 …
16SEP94 15:47:24 FINISH JOB CATCHUP-NO T_0:0:0:12 J_2 …
16SEP94 15:48:52 BEGIN JOB (SUPER.SUPER)CATCHUP-NO:2 … .
16SEP94 15:49:53 FINISH JOB CATCHUP-NO T_0:0:0:13 J_2 …
16SEP94 15:50:52 BEGIN JOB (SUPER.SUPER)CATCHUP-NO:3 … .
16SEP94 15:51:53 FINISH JOB CATCHUP-NO T_0:0:0:13 J_2 … .
```

CLASS Executor Attribute

The CLASS executor attribute specifies an executor's classes, thereby linking the classes and hence their jobs to the executor's CPU. This link enables the scheduler to start, in the specified CPU, the initial process (the executor program) of each job.

```
CL[ASS] { class-name
( class-name [ , class-name ]… )
*)
```

class-name

is the name of a class.

*

specifies all classes, thus making the executor available for use by jobs from any class.

Considerations

- An executor added without the CLASS attribute adopts the DEFAULT-CLASS scheduler attribute. To display the value of this attribute, use the INFO SCHEDULER command.
- You can use the CLASS attribute to assign up to eight existing classes to an executor. (An existing class is one added to the scheduler by the ADD CLASS command.) You also can use the attribute to dissociate classes from an executor.
- To assign a class to an existing executor, specify the name of the class and the names of the executor's pre-existing classes. (If you specify only the name of the new class, the scheduler dissociates all other classes from the executor.)

- The order in which you assign classes to an executor determines the order in which the scheduler scans them for jobs. For example, assigning classes B and A to an executor (in that order) makes the scheduler scan class B before it scans class A. Jobs in class B therefore run before jobs in class A.
- You can assign a class to more than one executor.
- To dissociate a class from an executor, specify the names of the classes that are to remain assigned to the executor. For more information, see ADD EXECUTOR Command on page 6-42 and ALTER EXECUTOR Command on page 6-59.
- JOBCLASS is an alias of the attribute keyword CLASS.

Examples

- This example shows various executor commands configuring the classes of three executors:

```
40} SET EXECUTOR CLASS (FIRST, SECOND, THIRD, DEFAULT)
41} SHOW EXECUTOR CLASS
EXECUTOR ATTRIBUTES
classes: FIRST
SECOND
THIRD
DEFAULT
42} ADD EXECUTOR EX-1, CPU 1
Executor EX-1 added
43} ADD EXECUTOR EX-2, CPU 2, CLASS DEFAULT
Executor EX-2 added
44} ADD EXECUTOR EX-3, CPU 3, CLASS (THIRD, SECOND, FIRST)
Executor EX-3 added
45} INFO EXECUTOR *, CLASS
EXECUTOR ATTRIBUTES for EX-1
classes: FIRST
SECOND
THIRD
DEFAULT
EXECUTOR ATTRIBUTES for EX-2
classes: DEFAULT
EXECUTOR ATTRIBUTES for EX-3
classes: THIRD
SECOND
FIRST
46} ALTER EXECUTOR EX-1, CLASS (FIRST, SECOND, THIRD)
Executor EX-1 altered
47} INFO EXECUTOR EX-1, CLASS
EXECUTOR ATTRIBUTES for EX-1
classes: FIRST
SECOND
THIRD
```

- This example shows an executor adopting its CLASS attribute from the DEFAULT-CLASS scheduler attribute:

```
56} INFO SCHEDULER, DEFAULT-CLASS
SCHEDULER ATTRIBUTES
default-class: STANDARD
57} SHOW EXECUTOR
EXECUTOR ATTRIBUTES
58} ADD EXECUTOR CPU-0, CPU 0
Executor CPU-0 added
59} INFO EXECUTOR CPU-0
EXECUTOR ATTRIBUTES for CPU-0
cpu: 0
classes: STANDARD
```

CLASS Job Attribute

The CLASS job attribute specifies the class to which a job belongs. The class links the job to an executor and hence to the executor's CPU. This link lets the scheduler start, in the specified CPU, the job's executor-program process.

```
CL[ASS] class-name
```

class-name

is the name of a class.

Considerations

- A job submitted without the CLASS attribute adopts the DEFAULT-CLASS scheduler attribute. To display the value of this attribute, use the INFO SCHEDULER command.
- The scheduler selects jobs from a class in this order:

  1. Jobs whose state is RUNNOW. These jobs have the highest priority. They execute as soon as you enter the RUNNOW JOB command. For more information, see RUNNOW JOB Command on page 6-128.
  2. Jobs whose state is RUNNEXT. These jobs take priority over all jobs except those whose state is RUNNOW and execute as soon as there is an available executor for their class. When more than one job is in the RUNNEXT state, selection is by SELPRI attribute. If RUNNEXT jobs have the same SELPRI attribute, selection is by submission time on a first-in, first-out (FIFO) basis. For more information, see RUNNEXT JOB Command on page 6-125.
  3. Jobs whose state is READY. The scheduler selects these jobs by their SELPRI attribute. Selection of jobs with the same SELPRI attribute and run time is by submission time on a FIFO basis.

  For more information about job selection, see Section 1, NetBatch Introduction.

- JOBCLASS is an alias of the attribute keyword CLASS.

Example

This example shows a job adopting its CLASS attribute from the scheduler's DEFAULT-CLASS attribute. The class has the attribute INITIATION OFF, which prevents the job from running, as does the executor whose state is OFF. The example shows the correction of both these conditions, which enables the job to run.

```
87} INFO SCHEDULER, DEFAULT-CLASS
SCHEDULER ATTRIBUTES
default-class: DEFAULT
88} INFO CLASS DEFAULT
CLASS ATTRIBUTE for DEFAULT
initiation: Off
89} INFO EXECUTOR *
EXECUTOR ATTRIBUTES for ANY-JOB
cpu: 0
classes: DEFAULT
90} STATUS EXECUTOR ANY-JOB
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
------------------------ --- ------ ---- -----
ANY-JOB 0 OFF
91} SUBMIT JOB X, IN DELAY04
Job X job number 2 submitted
92} STATUS JOB X
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
2 X 255,205 READY DEFAULT
93} ALTER CLASS DEFAULT, INITIATION ON
Class DEFAULT altered
94} STATUS JOB X
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
2 X 255,205 READY DEFAULT
95} START EXECUTOR ANY-JOB
Executor ANY-JOB started
96} STATUS JOB X
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
2 X 255,205 224 EXECUTING DEFAULT
```

CPU Executor Attribute

The CPU executor attribute assigns an executor to a CPU. This assignment lets the scheduler execute in the CPU the initial processes (the executor programs) of jobs in the executor's classes.

```
CP[U] cpu-number
```

cpu-number

is the number of a CPU configured for the scheduler's node. If you specify a nonexistent CPU, BATCHCOM displays a message advising the allowable CPU number range for the node. For example:

```
2074-E CPU must be 0 to 15
```

Considerations

- You cannot specify more than one CPU per executor. However, you can assign any number of executors to a single CPU.
- The CPU attribute applies to both local and remote jobs. For example, a job with a remote executor program runs in the CPU specified by its executor, but on the remote node. If the specified CPU is unavailable on the remote node, the scheduler runs the job in the node's highest-numbered available CPU. The job does not run if its executor's CPU is down on the local node, even though the corresponding remote CPU might be available.
- The scheduler plays no part in determining the CPU of processes created by an executor program. (The program performs this function independently of the scheduler.)
- These considerations relate to CPU failure:
  - When a CPU fails, the scheduler updates the state of any executor assigned to that CPU:

State Before CPU Failure	State After CPU Failure	State After CPU Reload
ACTIVE	DOWN	ON
DELETE	Executor deleted	Executor deleted
OFF	OFF	OFF
ON	DOWN	ON
STOP	OFF	OFF

  - Processes of a job using an executor whose CPU fails continue to run if each has a backup. The scheduler's treatment of the job if any process has no backup depends on the values of the job's IFFAILS, RESTART, STALL, and STOP-ON-ABEND attributes. For more information, see IFFAILS Job Attribute

on page 7-64, RESTART Job Attribute on page 7-95, STALL Job Attribute on page 7-102, and STOP-ON-ABEND Job Attribute on page 7-105.

Examples

- This example shows the CPUs of a job's processes. The job's TACL executor-program process $X228 runs in CPU 0 (specified by the CPU attribute of the job's executor). FUP processes $X and $Y created by the TACL process run in the CPU specified by their RUN commands. FUP process $Z runs in the CPU of the TACL process. (This setting is the TACL default for processes whose RUN commands do not include the CPU run option. For more information, see the *TACL Reference Manual.*)

```
> STATUS *, TERM
Process … Userid Program file Hometerm
$C4 1,36 … 205,70 TACL $T4.#A
$ZBAT 1,48 … 255,255 NETBATCH $T4.#A
$C4 B 2,29 … 205,70 TACL $T4.#A
$ZBAT B 3,30 … 255,255 NETBATCH $T4.#A
> FUP COPY FUPS
FUP /OUT $NULL, NAME $X, NOWAIT, CPU 1/ FILES *
FUP /OUT $NULL, NAME $Y, NOWAIT, CPU 2/ FILES *
FUP /OUT $NULL, NAME $Z/ FILES *
> BATCHCOM $ZBAT; INFO EXECUTOR *
EXECUTOR ATTRIBUTES for FUP-JOBS
cpu: 0
classes: DEFAULT
> BATCHCOM $ZBAT; SUBMIT JOB F, EXECUTOR-PROGRAM TACL, IN
FUPS, CLASS DEFAULT
Job F job number 499 submitted
> BATCHCOM $ZBAT; STATUS JOB 499
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
499 F 205,70 67 EXECUTING DEFAULT
> BATCHCOM $ZBAT; STATUS EXECUTOR FUP-JOBS
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
----------------------- --- ------ ---- -------
FUP-JOBS 0 ACTIVE 499 DEFAULT
> STATUS *, USER
Process … Userid Program file Hometerm
$X228 0,30 … 205,70 TACL $ZBAT
$Z 0,65 … 205,70 FUP $ZBAT
$X 1,31 … 205,70 FUP $ZBAT
$C4 1,36 … 205,70 TACL $T4.#A
$C4 B 2,29 … 205,70 TACL $T4.#A
$Y 2,34 … 205,70 FUP $ZBAT
```

- This example shows commands specifying the CPU attributes of executors for schedulers running on four-CPU and eight-CPU nodes:

```
9} OPEN \MELBDEV.$ZBAT
NETBATCH SERVER - T9190D20 …
```

```
10} INFO EXECUTOR DEV-1, CPU
2087-E EXECUTOR DEV-1 does not exist
11} ADD EXECUTOR DEV-1, CLASS *, CPU 4
2074-E CPU must be 0 to 3
12} ADD EXECUTOR DEV-1, CLASS *, CPU 3
Executor DEV-1 added
13} OPEN \MELBQAT.$ZBAT
NETBATCH SERVER - T9190C23 …
14} INFO EXECUTOR QAT-5, CPU
EXECUTOR ATTRIBUTES for QAT-5
cpu: 5
15} ALTER EXECUTOR QAT-5, CPU 8
2074-E CPU must be 0 to 7
16} ALTER EXECUTOR QAT-5, CPU 7
Executor QAT-5 altered
```

DEFAULT-CLASS Scheduler Attribute

The DEFAULT-CLASS scheduler attribute specifies the class of a job submitted without the CLASS job attribute. It also specifies the class of an executor added without the CLASS executor attribute. For more information, see CLASS Job Attribute on page 7-32 and CLASS Executor Attribute on page 7-30.

```
D[EFAULT]-C[LASS] class-name
```

class-name

is the name of a class.

Considerations

- A scheduler adopts the attribute DEFAULT-CLASS DEFAULT by default when cold started.
- The class specified by the DEFAULT-CLASS attribute is unavailable for use by jobs or assignment to executors until added to the scheduler. To add the class to the scheduler, use the ADD CLASS command.

Note. This consideration applies in particular to class DEFAULT, which is the default class name after a scheduler cold start.

Examples

- This example shows a cold start of a scheduler. The scheduler's DEFAULT-CLASS attribute after startup specifies the nonexistent class DEFAULT. The example shows the ADD CLASS command adding this class to the scheduler.

```
> LOGON SUPER.FPP, psswrd
> NETBATCH / NAME $ZBAT, NOWAIT / $DATA7.ZBAT !
> BATCHCOM $ZBAT; ADD SCHEDULER
Scheduler added
```

```
> BATCHCOM $ZBAT; START SCHEDULER
Scheduler started
> BATCHCOM $ZBAT; INFO SCHEDULER, DEFAULT-CLASS
SCHEDULER ATTRIBUTES
default-class: DEFAULT
> BATCHCOM $ZBAT; INFO CLASS DEFAULT
2105-E CLASS DEFAULT does not exist
> BATCHCOM $ZBAT; ADD CLASS DEFAULT, INITIATION OFF
Class DEFAULT added
> BATCHCOM $ZBAT; INFO CLASS DEFAULT
CLASS ATTRIBUTE for DEFAULT
initiation: Off
```

- This example shows a job adopting its CLASS attribute from the scheduler's DEFAULT-CLASS attribute:

```
9} INFO SCHEDULER, DEFAULT-CLASS
SCHEDULER ATTRIBUTES
default-class: ENFORM-REPORTS
10} SUBMIT JOB RECEIPTS, EXECUTOR-PROGRAM ENFORM, AFTER
01JUL93
Job RECEIPTS job number 299 submitted
11} INFO JOB RECEIPTS, CLASS
JOB ATTRIBUTES for RECEIPTS
jobnumber: 299
class: ENFORM-REPORTS
```

- This example shows a SUBMIT JOB command specifying a job's CLASS attribute, thus overriding the scheduler's DEFAULT-CLASS attribute:

```
20} INFO SCHEDULER, DEFAULT-CLASS
SCHEDULER ATTRIBUTES
default-class: ADMINISTRATION
21} SUBMIT JOB PERSONNEL, CLASS SALES, HOLD ON
Job PERSONNEL job number 301 submitted
22} INFO JOB PERSONNEL, CLASS
JOB ATTRIBUTES for PERSONNEL
jobnumber: 301
class: SALES
```

- This example shows an executor adopting its CLASS attribute from the scheduler's DEFAULT-CLASS attribute:

```
26} INFO SCHEDULER, DEFAULT-CLASS
SCHEDULER ATTRIBUTES
default-class: DEFAULT
27} ADD EXECUTOR EXEC-3, CPU 3
Executor EXEC-3 added
28} INFO EXECUTOR EXEC-3, CLASS
EXECUTOR ATTRIBUTES for EXEC-3
classes: DEFAULT
```

- This example shows an ADD EXECUTOR command specifying an executor's CLASS attribute, thus overriding the scheduler's DEFAULT-CLASS attribute:

```
35} INFO SCHEDULER, DEFAULT-CLASS
SCHEDULER ATTRIBUTES
default-class: BUILDS
36} ADD EXECUTOR DEVELOPMENT, CPU 2, CLASS COMPILES
Executor DEVELOPMENT added
37} INFO EXECUTOR DEVELOPMENT, CLASS
EXECUTOR ATTRIBUTES for DEVELOPMENT
classes: COMPILES
```

DEFAULT-EXECUTOR-PROGRAM Scheduler Attribute

The DEFAULT-EXECUTOR-PROGRAM scheduler attribute specifies the executor program of a job submitted without the EXECUTOR-PROGRAM attribute. For more information, see EXECUTOR-PROGRAM Job Attribute on page 7-58.

```
D[EFAULT]-E[XECUTOR]-[PROGRAM] file-name
```

file-name

is the name of a program file.

Consideration

A scheduler adopts the attribute DEFAULT-EXECUTOR-PROGRAM $SYSTEM.SYSTEM.TACL by default when cold started.

Examples

- This example shows a job adopting its EXECUTOR-PROGRAM attribute from the scheduler's DEFAULT-EXECUTOR-PROGRAM attribute:

```
5} INFO SCHEDULER, DEFAULT-EXECUTOR-PROGRAM
SCHEDULER ATTRIBUTES
default-executor-program: $SYSTEM.SYSTEM.TACL
6} SUBMIT JOB DEP
Job DEP job number 321 submitted
7} INFO JOB DEP, EXECUTOR-PROGRAM
JOB ATTRIBUTES for DEP
jobnumber: 321
executor-program: $SYSTEM.SYSTEM.TACL
```

- This example shows a SUBMIT JOB command specifying a job's EXECUTOR-PROGRAM attribute, thus overriding the scheduler's DEFAULT-EXECUTOR-PROGRAM attribute:

```
11} INFO SCHEDULER, DEFAULT-EXECUTOR-PROGRAM
SCHEDULER ATTRIBUTES
default-executor-program: $SYSTEM.SYSTEM.FUP
12} SUBMIT JOB BC, EXECUTOR-PROGRAM $QAT2.T9190D20.BATCHCOM
```

```
Job BC job number 322 submitted
13} INFO JOB BC, EXECUTOR-PROGRAM
JOB ATTRIBUTES for BC
jobnumber: 322
executor-program: $QAT2.T9190D20.BATCHCOM
```

DEFAULT-HIGHPIN Scheduler Attribute

The DEFAULT-HIGHPIN scheduler attribute (available in D20 or later versions of the NetBatch product) specifies the HIGHPIN attribute of a job submitted without that attribute. For more information, see HIGHPIN Job Attribute on page 7-61.

```
D[EFAULT]-H[IGHPIN] { OF[F] | ON }
```

Consideration

A scheduler adopts the attribute DEFAULT-HIGHPIN OFF by default when cold started.

Examples

- This example shows a job adopting its HIGHPIN attribute from the scheduler's DEFAULT-HIGHPIN attribute:

```
12} INFO SCHEDULER, DEFAULT-HIGHPIN
SCHEDULER ATTRIBUTES
default-highpin: Off
13} SUBMIT JOB DHP-1
Job DHP-1 job number 399 submitted
14} INFO JOB DHP-1, HIGHPIN
JOB ATTRIBUTES for DHP-1
jobnumber: 399
highpin: Off
```

- This example shows a SUBMIT JOB command specifying a job's HIGHPIN attribute, thus overriding the scheduler's DEFAULT-HIGHPIN attribute:

```
16} INFO SCHEDULER, DEFAULT-HIGHPIN
SCHEDULER ATTRIBUTES
default-highpin: On
17} SUBMIT JOB DHP-2, HIGHPIN OFF
Job DHP-2 job number 400 submitted
18} INFO JOB DHP-2, HIGHPIN
JOB ATTRIBUTES for DHP-2
jobnumber: 400
highpin: Off
```

DEFAULT-MAXPRINTLINES Scheduler Attribute

The DEFAULT-MAXPRINTLINES scheduler attribute specifies the maximum number of output-file print lines of a job submitted without the MAXPRINTLINES attribute. For more information, see MAXPRINTLINES Job Attribute on page 7-82.

```
D[EFAULT]-MAXPRINTL[INES] { number | NON[E]
```

number

is a number in the range 120 through 65534 specifying the maximum number of print lines.

NONE

specifies no maximum.

Considerations

- A scheduler adopts the attribute DEFAULT-MAXPRINTLINES NONE by default when cold started.
- To avoid having conflicting DEFAULT-MAXPRINTLINES and DEFAULT-MAXPRINTPAGES attributes, do one of:
  - Set the value of each attribute to NONE.
  - Set the value of one attribute to NONE and the value of the other to number.

Examples

- This example shows a job adopting its MAXPRINTLINES and MAXPRINTPAGES attributes from the scheduler's DEFAULT-MAXPRINTLINES and DEFAULT-MAXPRINTPAGES attributes:

```
7} INFO SCHEDULER, DEFAULT-MAXPRINTLINES, DEFAULT-
MAXPRINTPAGES
SCHEDULER ATTRIBUTES
default-maxprintlines: None
default-maxprintpages: 25
8} SUBMIT JOB MPL-1
Job MPL-1 job number 272 submitted
9} INFO JOB MPL-1, MAXPRINTLINES, MAXPRINTPAGES
JOB ATTRIBUTES for MPL-1
jobnumber: 272
maxprintlines: None
maxprintpages: 25
```

- This example shows a SUBMIT JOB command specifying a job's MAXPRINTLINES and MAXPRINTPAGES attributes, thus overriding the

scheduler's DEFAULT-MAXPRINTLINES and DEFAULT-MAXPRINTPAGES attributes:

```
11} INFO SCHEDULER, DEFAULT-MAXPRINTLINES, DEFAULT-
MAXPRINTPAGES
SCHEDULER ATTRIBUTES
default-maxprintlines: 600
default-maxprintpages: None
12} SUBMIT JOB MPL-2, MAXPRINTLINES NONE, MAXPRINTPAGES 10
Job MPL-2 job number 273 submitted
13} INFO JOB MPL-2, MAXPRINTLINES, MAXPRINTPAGES
JOB ATTRIBUTES for MPL-2
jobnumber: 273
maxprintlines: None
maxprintpages: 10
```

DEFAULT-MAXPRINTPAGES Scheduler Attribute

The DEFAULT-MAXPRINTPAGES scheduler attribute specifies the maximum number of output-file print pages of a job submitted without the MAXPRINTPAGES attribute. For more information, see MAXPRINTPAGES Job Attribute on page 7-83.

```
D[EFAULT]-MAXPRINTP[AGES] { number | NON[E]
```

number

is a number in the range 2 through 65534 specifying the maximum number of print pages.

NONE

specifies no maximum.

Considerations

- A scheduler adopts the attribute DEFAULT-MAXPRINTPAGES NONE by default when cold started.
- To avoid having conflicting DEFAULT-MAXPRINTLINES and DEFAULT-MAXPRINTPAGES attributes:
  - Set the value of each attribute to NONE.
  - Set the value of one attribute to NONE and the value of the other to number.

Examples

- This example shows a job adopting its MAXPRINTLINES and MAXPRINTPAGES attributes from the scheduler's DEFAULT-MAXPRINTLINES and DEFAULT-MAXPRINTPAGES attributes:

```
15} INFO SCHEDULER, DEFAULT-MAXPRINTLINES, DEFAULT-
MAXPRINTPAGES
SCHEDULER ATTRIBUTES
default-maxprintlines: 1200
default-maxprintpages: None
16} SUBMIT JOB MPP-1
Job MPP-1 job number 274 submitted
17} INFO JOB MPP-1, MAXPRINTLINES, MAXPRINTPAGES
JOB ATTRIBUTES for MPP-1
jobnumber: 274
maxprintlines: 1200
maxprintpages: None
```

- This example shows a SUBMIT JOB command specifying a job's MAXPRINTLINES and MAXPRINTPAGES attributes, thus overriding the scheduler's DEFAULT-MAXPRINTLINES and DEFAULT-MAXPRINTPAGES attributes:

```
19} INFO SCHEDULER, DEFAULT-MAXPRINTLINES, DEFAULT-
MAXPRINTPAGES
SCHEDULER ATTRIBUTES
default-maxprintlines: None
default-maxprintpages: 20
20} SUBMIT JOB MPP-2, MAXPRINTLINES 1800, MAXPRINTPAGES NONE
Job MPP-2 job number 275 submitted
21} INFO JOB MPP-2, MAXPRINTLINES, MAXPRINTPAGES
JOB ATTRIBUTES for MPP-2
jobnumber: 275
maxprintlines: 1800
maxprintpages: None
```

DEFAULT-OUT Scheduler Attribute

The DEFAULT-OUT scheduler attribute specifies the output file of a job submitted without the OUT attribute. For more information, see OUT Job Attribute on page 7-87.

```
D[EFAULT]-O[UT] file-name
```

file-name

is the name of an output file. The file can be a device, a process, or a disk file.

Consideration

A scheduler adopts the attribute DEFAULT-OUT $S.#BATCH by default when cold started.

Examples

- This example shows a job adopting its OUT attribute from the scheduler's DEFAULT-OUT attribute:

```
9} INFO SCHEDULER, DEFAULT-OUT
SCHEDULER ATTRIBUTES
default-out: \MELBDEV.$S.#BATCH
10} SUBMIT JOB DO1
Job DO1 Jobnumber 1 submitted
11} INFO JOB DO1, OUT
JOB ATTRIBUTES for DO1
jobnumber: 1
out: \MELBDEV.$S.#BATCH
```

- This example shows a SUBMIT JOB command specifying a job's OUT attribute, thus overriding the scheduler's DEFAULT-OUT attribute:

```
10} INFO SCHEDULER, DEFAULT-OUT
SCHEDULER ATTRIBUTES
default-out: \MELBDEV.$T4.#A
11} SUBMIT JOB D02, OUT $DATA7.NB.D02OUT
Job D02 job number 2 submitted
12} INFO JOB D02, OUT
JOB ATTRIBUTES for D02
jobnumber: 2
out: \MELBDEV.$DATA7.NB.D02OUT
```

DEFAULT-PRI Scheduler Attribute

The DEFAULT-PRI scheduler attribute specifies the execution priority of the executor-program process of a job submitted without the PRI attribute. For more information, see PRI Job Attribute on page 7-92.

```
D[EFAULT]-P[RI] number
```

number

is a number in the range 1 through 199 (1 is lowest) specifying the execution priority of a job's executor-program process.

Consideration

A scheduler adopts the attribute DEFAULT-PRI 120 by default when cold started.

Examples

- This example shows a job adopting its PRI attribute from the scheduler's DEFAULT-PRI attribute:

```
3} INFO SCHEDULER, DEFAULT-PRI
SCHEDULER ATTRIBUTES
```

```
default-pri: 120
4} SUBMIT JOB PRI-1
Job PRI-1 job number 276 submitted
5} INFO JOB PRI-1, PRI
JOB ATTRIBUTES for PRI-1
jobnumber: 276
pri: 120
```

- This example shows a SUBMIT JOB command specifying a job's PRI attribute, thus overriding the scheduler's DEFAULT-PRI attribute:

```
11} INFO SCHEDULER, DEFAULT-PRI
SCHEDULER ATTRIBUTES
default-pri: 99
12} SUBMIT JOB PRI-2, PRI 149
Job PRI-2 job number 278 submitted
13} INFO JOB PRI-2, PRI
JOB ATTRIBUTES for PRI-2
jobnumber: 278
pri: 149
```

DEFAULT-SELPRI Scheduler Attribute

The DEFAULT-SELPRI scheduler attribute specifies the selection priority of a job submitted without the SELPRI attribute. For more information, see SELPRI Job Attribute on page 7-101.

```
D[EFAULT]-SE[LPRI] number
```

number

is a number in the range 0 through 7 (0 is lowest) specifying the selection priority.

Consideration

A scheduler adopts the attribute DEFAULT-SELPRI 3 by default when cold started.

Examples

- This example shows a job adopting its SELPRI attribute from the scheduler's DEFAULT-SELPRI attribute:

```
12} INFO SCHEDULER, DEFAULT-SELPRI
SCHEDULER ATTRIBUTES
default-selpri: 3
13} SUBMIT JOB A, HOLD ON
Job A job number 315 submitted
14} INFO JOB A, SELPRI
JOB ATTRIBUTES for A
jobnumber: 315
selpri: 3
```

- This example shows SUBMIT JOB command specifying a job's SELPRI attribute, thus overriding the scheduler's DEFAULT-SELPRI attribute:

```
17} INFO SCHEDULER, DEFAULT-SELPRI
SCHEDULER ATTRIBUTES
default-selpri: 5
18} SUBMIT JOB B, HOLD ON, SELPRI 4
Job B job number 316 submitted
19} INFO JOB B, SELPRI
JOB ATTRIBUTES for B
jobnumber: 316
selpri: 4
```

DEFAULT-STALL Scheduler Attribute

The DEFAULT-STALL scheduler attribute specifies the STALL attribute of a job submitted without that attribute. For more information, see STALL Job Attribute on page 7-102.

```
D[EFAULT]-ST[ALL] { OF[F] | ON }
```

Consideration

A scheduler adopts the attribute DEFAULT-STALL OFF by default when cold started.

Examples

- This example shows a job adopting its STALL attribute from the scheduler's DEFAULT-STALL attribute:

```
7} INFO SCHEDULER, DEFAULT-STALL
SCHEDULER ATTRIBUTES
default-stall: On
8} SUBMIT JOB DS1, EVERY 7 DAYS, IFFAILS ON, RESTART OFF,
STOP-ON-ABEND ON
Job DS1 job number 2 submitted
9} INFO JOB DS1, STALL
JOB ATTRIBUTES for DS1
jobnumber: 2
stall: On
```

- This example shows a SUBMIT JOB command specifying a job's STALL attribute, thus overriding the scheduler's DEFAULT-STALL attribute:

```
11} INFO SCHEDULER, DEFAULT-STALL
SCHEDULER ATTRIBUTES
default-stall: On
12} SUBMIT JOB DS2, RESTART ON, STOP-ON-ABEND OFF, STALL OFF
Job DS2 job number 3 submitted
13} INFO JOB DS2, STALL
JOB ATTRIBUTES for DS2
jobnumber: 3
stall: Off
```

DEFAULT-STOP-ON-ABEND Scheduler Attribute

The DEFAULT-STOP-ON-ABEND scheduler attribute specifies the STOP-ON-ABEND attribute of a job submitted without that attribute. For more information, see STOP-ON-ABEND Job Attribute on page 7-105.

```
D[EFAULT]-[STOP]-[ON]-[ABEND] { OF[F] | ON }
```

Consideration

A scheduler adopts the attribute DEFAULT-STOP-ON-ABEND OFF by default when cold started.

Examples

- This example shows a job adopting its STOP-ON-ABEND attribute from the scheduler's DEFAULT-STOP-ON-ABEND attribute:

```
2} INFO SCHEDULER, DEFAULT-STOP-ON-ABEND
SCHEDULER ATTRIBUTES
default-stop-on-abend: Off
3} SUBMIT JOB DSOA-1, RESTART OFF, STALL OFF
Job DSOA-1 job number 397 submitted
4} INFO JOB DSOA-1, STOP-ON-ABEND
JOB ATTRIBUTES for DSOA-1
jobnumber: 397
stop-on-abend: Off
```

- This example shows a SUBMIT JOB command specifying a job's STOP-ON-ABEND attribute, thus overriding the scheduler's DEFAULT-STOP-ON-ABEND attribute:

```
6} INFO SCHEDULER, DEFAULT-STOP-ON-ABEND
SCHEDULER ATTRIBUTES
default-stop-on-abend: On
7} SUBMIT JOB DSOA-2, EVERY 2 DAYS, IFFAILS ON, RESTART ON,
STALL ON, STOP-ON-ABEND OFF
Job DSOA-2 job number 398 submitted
8} INFO JOB DSOA-2, STOP-ON-ABEND
JOB ATTRIBUTES for DSOA-2
jobnumber: 398
stop-on-abend: Off
```

DEFINE Attachment-Set Attribute

The DEFINE attachment-set attribute specifies the name and attributes of a DEFINE. (A DEFINE is a named set of attributes and associated values.) The scheduler passes the DEFINE to the executor-program process of a job using the attachment set when the job starts.

```
( DEFI[NE] DEFINE-name-1 , [ LIK[E] DEFINE-name-2 , ]
[ CLASS DEFINE-class , ]
DEFINE-attribute [ , DEFINE-attribute ]… )
```

DEFINE-name-1

is the name of a DEFINE. The name can contain from 2 through 24 characters. The first character must be an equals sign (=) and the second a letter. The remaining characters can be letters, numbers, hyphens, underscores, or circumflexes. The name cannot contain spaces.

Note. When you specify DEFINE names, consider these naming restrictions:

- To avoid confusion, do not use the attribute keywords ASSIGN, DEFINE, and PARAM as DEFINE names.
- HP reserves DEFINE names whose second character is an underscore for internal use in products such as the TACL command interpreter.
- HP reserves DEFINE names beginning with =_ZBAT for NetBatch product use.

LIKE

specifies, for ADD, ALTER, and SET attachment-set commands, that all attributes of DEFINE *DEFINE-name-1* are to match those of DEFINE *DEFINE-name-2*. LIKE overrides attachment-set attributes in the working-attributes set and attachment-set defaults (see Table 6-3 on page 6-35). *DEFINE-attribute* overrides *DEFINE-name-2* attributes of the same type.

DEFINE-name-2

is the name of an existing DEFINE whose attributes you want DEFINE *DEFINE-name-1* to match.

DEFINE-class

specifies the CLASS attribute of the DEFINE (that is, the DEFINE's type). The options are:

CATALOG	MAP	SORT	SUBSORT
DEFAULT	SEARCH	SPOOL	TAPE

The default is CLASS MAP. Table 7-1 on page 7-48 describes each DEFINE type.

Table 7-1. DEFINE Types (Classes)

Type	Description
CATALOG	Specifies the location of a SQL/MP catalog. You can enter the logical name of the catalog DEFINE instead of a catalog name in, for example, CATALOG clauses in NonStop SQL/MP data definition language (DDL) statements.
DEFAULTS	Holds the standard default values of a process such as the default volume.
MAP	Redirects or substitutes a file. You can enter the logical name of a map DEFINE instead of a physical file name in a command or procedure call.
SEARCH	Specifies a search list of subvolumes for a program.
SORT	Sets parameters for a FASTSORT process.
SPOOL	Passes information to the spooler collector process. The attributes of a spool DEFINE specify parameters such as the spooler location and batch name.
SUBSORT	Sets parameters for a parallel sort run under the FASTSORT subsystem.
TAPE	Passes information to a tape process during a labeled-tape operation. Tape DEFINE attributes specify parameters such as the tape device name and the record format.

DEFINE-attribute

is a DEFINE attribute whose value depends on the DEFINE type as specified by CLASS *DEFINE-class*. Table 7-2 lists the DEFINE types and their attributes. For detailed descriptions of the attributes, see the description of the SET DEFINE command in the *TACL Reference Manual*.

Table 7-2. DEFINE Attributes (page 1 of 3)

Type	Attributes
CATALOG	SUBVOL *subvolume*
DEFAULTS	CATALOG *subvolume* SWAP *volume* VOLUME *subvolume*
MAP	FILE *file-name*
SEARCH	{ SUBVOL *n* SUBVOL *n*,RELSUBVOL *n* } *subvolume* [, { SUBVOL *n* RELSUBVOL *n* } *subvolume*]…

Table 7-2. DEFINE Attributes (page 2 of 3)

Type	Attributes
SORT	BLOCK *size* CPU *cpu-number* CPUS { (*cpu-number* [, *cpu-number*]…) ALL) } MODE { AUTOMATIC MINSPACE MINTIME } NOTCPUS (*cpu-number* [, *cpu-number*]…) PRI *priority* PROGRAM *file-name* SCRATCH *file-name* SEGMENT *size* SUBSORTS (*DEFINE-name* [, *DEFINE-name*]…) SWAP *file-name*

Table 7-2. DEFINE Attributes (page 3 of 3)

Type	Attributes
SPOOL	BATCHNAME *batch-name* COPIES *num* FORM *form-name* HOLD { OFF | ON } HOLDAFTER { OFF | ON } LOC [*node*.] $*collector* [.#*group-name* [.*dest*]] MAXPRINTLINES *num* MAXPRINTPAGES *num* OWNER { *group-name.user-name* " *group-ID*, *user-ID* " } PAGESIZE *num* REPORT *report-name* SELPRI *num*
SUBSORT	CPU *cpu-number* PRI *priority* PROGRAM *file-name* SCRATCH *file-name* SEGMENT *size* SWAP *file-name*
TAPE	BLOCKLEN *block-length* DENSITY { 800 | 1600 | 6250 } DEVICE $*device-name* EBCDIC { IN | OUT |OFF | ON } EXPIRATION *date* FILEID *file-name* FILESECT *volume-order* FILESEQ *file-order* GEN *gen-num* LABELS { ANSI IBM OMITTED BYPASS BACKUP IBMBACKUP } MOUNTMSG " *text* " OWNER *owner-ID* RECFORMBBC { F| U } RECLEN *record-length* REELS *volumes* RETENTION *days* SYSTEM *node* TAPEMODE { STARTSTOP| STREAM } USE { IN | OUT | EXTEND | OPENFLAG } VERSION *num* VOLUMEBBC { *volume-ID* | SCRATCH }

Consideration

For more information on DEFINEs, see the *TACL Reference Manual* and the *TACL Programming Guide*.

Examples

- This example shows the scheduler passing DEFINEs from a job's attachment set to the job's TACL executor-program process:

```
> LOGON NB.USER, psswrd
> FUP COPY INFILE
INFO DEFINE **, DETAIL
> BATCHCOM $ZBAT; ADD ATTACHMENT-SET D, (DEFINE =_DEFAULTS,
CLASS DEFAULTS, VOLUME $A.NB), (DEFINE =OUT, CLASS SPOOL,
BATCHNAME MYJOBS, COPIES 2, HOLDAFTER ON, LOC
\MELBQAT.$S, OWNER "133,2")
Attachment-set (NB.USER)D added
> BATCHCOM $ZBAT; SUBMIT JOB W, EXECUTOR-PROGRAM TACL, IN
INFILE, OUT OUTFILE, ATTACHMENT-SET D
Job W job number 410 submitted
> FUP COPY OUTFILE
INFO DEFINE **, DETAIL
Define Name =OUT
CLASS SPOOL
LOC \MELBQAT.$S
COPIES 2
HOLDAFTER ON
OWNER 133,2
BATCHNAME MYJOBS
Define Name =_DEFAULTS
CLASS DEFAULTS
VOLUME $A.NB
```

- This example shows various attachment-set commands adding, altering, and displaying an attachment set's DEFINE attributes. The TACL environment is the source of DEFINEs =_DEFAULTS and =TAPE1.

```
> LOGON FPP.MANAGER, psswrd
> ALTER DEFINE =_DEFAULTS, VOLUME $DATA7.DEVBAKUP
> ADD DEFINE =TAPE1, CLASS TAPE, VOLUME 30, LABELS ANSI,
FILEID BACKUPS, DEVICE \MELBDEV.$TAPE
> INFO DEFINE **, DETAIL
Define Name =TAPE1
CLASS TAPE
VOLUME 30
LABELS ANSI
FILEID BACKUPS
DEVICE \MELBDEV.$TAPE
Define Name =_DEFAULTS
CLASS DEFAULTS
VOLUME $DATA7.DEVBAKUP
> BATCHCOM $ZBAT
1} SHOW ATTACHMENT-SET DEFINE
```

```
ATTACHMENT-SET ATTRIBUTES
attachments: DEFINE =TAPE1, CLASS TAPE, VOLUME "30 ",
LABELS ANSI, FILEID BACKUPS, DEVICE
\MELBDEV.$TAPE
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$DATA7.DEVBAKUP
2} SET ATTACHMENT-SET (DEFINE =TAPE2, LIKE =TAPE1, MOUNTMSG
"DAILY BACKUPS ONLY", DEVICE \MELBQAT.$TAPE)
3} ADD ATTACHMENT-SET BKUPS, (DEFINE =TAPE3, LIKE =TAPE2,
DEVICE \MELBORN.$TAPE)
Attachment-set (FPP.MANAGER)BKUPS added
4} INFO ATTACHMENT-SET BKUPS, DEFINE *
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)BKUPS
attachments: DEFINE =TAPE1, CLASS TAPE, VOLUME "30 ",
LABELS ANSI, FILEID BACKUPS, DEVICE
\MELBDEV.$TAPE
DEFINE =TAPE2, CLASS TAPE, VOLUME "30 ",
LABELS ANSI, FILEID BACKUPS, DEVICE
\MELBQAT.$TAPE, MOUNTMSG "DAILY
BACKUPS ONLY"
DEFINE =TAPE3, CLASS TAPE, VOLUME "30 ",
LABELS ANSI, FILEID BACKUPS, DEVICE
\MELBORN.$TAPE, MOUNTMSG "DAILY
BACKUPS ONLY"
DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$DATA7.DEVBAKUP
5} ALTER ATTACHMENT-SET BKUPS, (DEFINE =_DEFAULTS, VOLUME
$DATA6.NBFILES)
Attachment-set (FPP.MANAGER)BKUPS altered
6} INFO ATTACHMENT-SET BKUPS, DEFINE =_DEFAULTS
ATTACHMENT-SET ATTRIBUTES for (FPP.MANAGER)BKUPS
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$DATA6.NBFILES
```

DESCRIPTION Job Attribute

The DESCRIPTION job attribute contains text that describes a job. The text appears in the scheduler's log file each time the job runs.

```
DES[CRIPTION] " [ string ] "
```

string

is a string of 1 through 1000 ASCII characters describing the job.

If you specify DESCRIPTION without *string* in an ALTER JOB command, the scheduler removes the DESCRIPTION attribute from the job. Specifying DESCRIPTION without *string* in a SET JOB command makes BATCHCOM remove the DESCRIPTION attribute from the working-attributes set.

Considerations

- A job submitted without the DESCRIPTION attribute has no descriptive text.

- BATCHCOM breaks lines when their length exceeds 70 characters. To force a line break, enter a percent (%) character where you want the break to occur.

  Characters preceded by a backslash (\) are treated as literal characters.

Example

- This example shows submission of a job with the DESCRIPTION attribute and the effect of the percent (%) and backslash (\) characters:

```
19} SUBMIT JOB DESCRIPTION-TEST, DESCRIPTION "This is a
sample job description t hat illustrates use of the
DESCRIPTION job attribute. You might be interested to see
where BATCHCOM breaks lines and the effects of the
percent (\%) %and backs
lash (\\) characters."
Job DESCRIPTION-TEST Jobnumber 2 submitted
20} INFO JOB 2, DESCRIPTION
JOB ATTRIBUTES for DESCRIPTION-TEST
jobnumber: 2
description:
This is a sample job description that illustrates use of the
DESCRIPTION job attribute. You might be interested to see
where BATCHCOM breaks lines and the effects of the percent
(%) and backslash (\) characters.
```

- The log-file entries relating to the job submitted in the previous example include:

```
19SEP94 16:31:00 BEGIN JOB (SUPER.SUPER)DESCRIPTION-TEST:1
E_E1 L_26 J_2 P_TACL
\MELBDEV.$X594:16602925 U_255,255
This is a sample job description that illustrates use of the
DESCRIPTION job attribute. You might be interested to see
where BATCHCOM breaks lines and the effects of the percent
(%) and backslash (\) characters. .
.
```

EMS Scheduler Attribute

The EMS scheduler attribute enables or disables event-message generation while the scheduler is running. For information about NetBatch event messages, see the *NetBatch Management Programming Manual*.

```
EM[S] { ER[RORS]
OF[F]
ON)
```

ERRORS

enables generation of all scheduler event messages except 102 (ZBAT-EVT-JOB-START) and 202 (ZBAT-EVT-JOB-NORMAL-STOP).

OFF

disables generation of all scheduler event messages.

ON

enables generation of all scheduler event messages.

Considerations

- A scheduler run without the EMS parameter in the RUN NETBATCH command adopts the attribute EMS OFF by default when cold started.
- A RUN NETBATCH command that includes the EMS parameter sets the EMS attribute to ON.

Example

In this example, the EMS parameter in the NETBATCH command sets the EMS attribute to ON, and an ALTER SCHEDULER command sets the attribute to ERRORS:

```
> NETBATCH /NAME $ZBAT, NOWAIT/ $DATA7.ZBAT ! EMS
> BATCHCOM $ZBAT
BATCHCOM - T9190D30 - (31OCT94^01JUN94)
(C)1986 Tandem (C)2004 Hewlett Packard Development Company, L.P.
NETBATCH SERVER - T9190D30 …
1} ADD SCHEDULER; START SCHEDULER
Scheduler added
Scheduler started
3} INFO SCHEDULER, EMS
SCHEDULER ATTRIBUTES
ems: On
4} ALTER SCHEDULER, EMS ERRORS
Scheduler altered
5} INFO SCHEDULER, EMS
SCHEDULER ATTRIBUTES
ems: Errors
```

EVERY Job Attribute

The EVERY job attribute specifies that job execution occur at regular, specified intervals.

```
EV[ERY] [ weeks WEEK[S]
days D[AYS]
hours [ : mins ] [HOURS]
hours H[OURS] [ mins MIN[UTES] ]
crontab-entry ]
```

weeks

is an integer in the range 1 through 52 specifying the interval in seven-day weeks.

days

is an integer in the range 1 through 365 specifying the interval in days.

hours

is an integer in the range 0 through 168 specifying the interval in hours.

mins

is an integer in the range 0 through 59 specifying the interval in minutes.

crontab-entry

specifies the interval in the form of this five-field entry:

minute hour day-of-month month day-of-week

minute

is an integer in the range 0 through 59 specifying the minute of the hour.

hour

is an integer in the range 0 through 23 specifying the hour of the day.

day-of-month

is an integer in the range 1 through 31 specifying the day of the month.

month

is an integer in the range 1 through 12 specifying the month of the year.

day-of-week

is an integer in the range 0 through 6 (for Sunday through Saturday) specifying the day of the week.

Each *crontab-entry* field can contain:

- A number in the specified range
- Two numbers separated by a dash to indicate an inclusive range
- A list of numbers separated by commas, which selects all numbers in the list
- An asterisk (*), meaning all legal values

The specification of days can be made by two fields (*day-of-month* and *day-of-week*). If you specify both as a list of elements, both are adhered to. For example, this entry specifies midnight on the first and fifteenth days of each month, as well as every Monday:

```
0 0 1,15 * 1
```

If you use only one of the two fields to specify days, you must put an asterisk in the other field.

Considerations

- A job with the EVERY attribute can run immediately on submission unless delayed by another attribute. When execution finishes, the scheduler reschedules the job to run when the EVERY interval expires. The scheduler calculates the next run time by adding the interval to the original submission time. The rescheduled job retains all its original attributes, including the job number.
- To remove the EVERY attribute from a job, use the ALTER JOB command to specify EVERY without any qualifiers; for example, ALTER JOB 19, EVERY.
- The NetBatch expression for a job with the CALENDAR or EVERY attribute is “recurrent job.” These attributes cause the job to run repeatedly (that is, recur) at a specified interval. Conversely, a nonrecurrent job is a job whose attributes do not include CALENDAR or EVERY.
- A job with the EVERY attribute accumulates a run backlog if it is held (HOLD ON) or suspended for longer than the EVERY interval and the scheduler has the attribute CATCHUP ON. When you take the job off hold or reactivate it, it runs repeatedly until the backlog clears. The job also accumulates a run backlog if it runs for longer than the EVERY interval. The scheduler treats a job with a HOLD ON set as a failed job. When a job is HOLD OFF, the scheduler correctly reschedules the job. If it is an EVERY JOB, it immediately runs once. For more information about run backlogs (including how to prevent them from running), see ACTIVATE JOB Command on page 6-32.
- The CALENDAR and EVERY job attributes are mutually exclusive. This restriction means a job can have one of the attributes, but not both. It also means that CALENDAR overrides EVERY when assigned to a job whose attributes include EVERY, and EVERY overrides CALENDAR when assigned to a job whose attributes include CALENDAR.

- For information about how the scheduler treats a recurrent job that fails, see IFFAILS Job Attribute on page 7-64, RESTART Job Attribute on page 7-95, STALL Job Attribute on page 7-102, STOP-ON-ABEND Job Attribute on page 7-105.
- For information about recurrent jobs that are also dependent jobs, see WAITON Job Attribute on page 7-119.

Examples

- This example shows the submission of a job whose EVERY attribute specifies execution at five-minute intervals. The job runs when submitted, finishes executing, then recurs with a next run time five minutes later than the original submission time.

```
> TIME
October 3, 1994 16:41:58
> BATCHCOM $ZBAT; SUBMIT JOB, EXECUTOR-PROGRAM DELAY, STARTUP
"1 MINS", EVERY 0:05 HOURS
Job ZBAT-0044 Jobnumber 44 submitted
> BATCHCOM $ZBAT; INFO JOB 44, EVERY
JOB ATTRIBUTES for ZBAT-0044
jobnumber: 44
every: 0:05 HOURS
next-runtime: 03OCT94 16:42:40
> BATCHCOM $ZBAT; STATUS JOB 44
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
44 ZBAT-0044 205,70 341 EXECUTING DEFAULT
> BATCHCOM $ZBAT; STATUS JOB 44
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
44 ZBAT-0044 205,70 341 16:47:40 DEFAULT
```

- This example shows the submission of a job that is to run at 5:15 a.m. and at 5:45 a.m., Monday through Friday:

```
12} SUBMIT JOB, EVERY 15,45 5 * * MON-FRI
Job ZBAT-0041 Jobnumber 41 submitted
13} INFO JOB 41, EVERY
JOB ATTRIBUTES for ZBAT-0041
jobnumber: 41
every:
minutes: 15,45
hours: 5
days: *
months: *
weekday: 1-5
```

- This example shows the submission of a job that is to run at noon on February 14:

```
39} ALTER JOB 42, EVERY 0 12 14 FEB *, INFO JOB 42, EVERY
Job ZBAT-0042 Jobnumber 42 altered
JOB ATTRIBUTES for ZBAT-0042
jobnumber: 42
every:
```

```
minutes: 0
hours: 12
days: 14
months: 2
weekday: *
```

EXECUTOR-PROGRAM Job Attribute

The EXECUTOR-PROGRAM job attribute specifies the program file of the program the scheduler starts as the initial process of a job.

```
E[XECUTOR]-P[ROGRAM] file-name
```

file-name

is the name of a program file.

BATCHCOM expands a partially qualified executor-program file name by using the defaults specified in the last VOLUME command. If there was no such command, the defaults current at the start of the session apply. If the resulting expanded name specifies a nonexistent file, BATCHCOM uses $SYSTEM.SYSTEM for expansion purposes. For example, file ID FUP expands to $SYSTEM.SYSTEM.FUP, not $DATA7.USER.FUP, because FUP does not exist in subvolume $DATA7.USER:

```
> FILES
No files match \MELBDEV.$DATA7.USER.*
> BATCHCOM $ZBAT
1} VOLUME
VOLUME $DATA7.USER, "NCNC"
2} SET JOB EXECUTOR-PROGRAM FUP
3} SHOW JOB EXECUTOR-PROGRAM
JOB ATTRIBUTES
executor-program: $SYSTEM.SYSTEM.FUP
```

Considerations

- A job submitted without the EXECUTOR-PROGRAM attribute adopts the DEFAULT-EXECUTOR-PROGRAM scheduler attribute. To display the value of this attribute, use the INFO SCHEDULER command.
- A job with a remote executor program runs in the CPU specified by its executor, but on the remote node. If the specified CPU is unavailable on the remote node, the scheduler runs the job in the node's highest-numbered CPU.
- The home terminal of a job's executor-program process is the scheduler unless specified otherwise by the TERM attribute. However, the scheduler plays no part in determining the home terminal of processes created by that process. (The program performs this function independently of the scheduler.)

Example

This example shows execution of a TACL job that starts three FUP processes. (The job inherits its EXECUTOR-PROGRAM attribute from the scheduler.) The example also shows the home terminals of the job's processes. The TACL executor-program process $Y561 uses the scheduler process $ZBAT as its home terminal. FUP processes $EP1 and $EP2 created by the TACL process use the home terminals specified by their RUN commands. FUP process $EP3 uses the home terminal of the TACL process. (This setting is the TACL default for processes whose RUN commands do not include the TERM run option. For more information, see the *TACL Reference Manual*.)

```
> STATUS *, TERM
Process Pri … Program file Hometerm
$C4 1,31 150 … TACL $T4.#A
$ZBAT 1,51 149 … NETBATCH $T4.#A
$C4 B 2,26 150 … TACL $T4.#A
$ZBAT B 2,52 149 … NETBATCH $T4.#A
> FUP COPY EP
FUP /OUT $NULL, NAME $EP1, NOWAIT, TERM \QA.$TRM2.#A/ FILES *
FUP /OUT $NULL, NAME $EP2, NOWAIT, TERM $NULL/ FILES *
FUP /OUT $NULL, NAME $EP3/ FILES *
> BATCHCOM $ZBAT; INFO SCHEDULER, DEFAULT-EXECUTOR-PROGRAM
SCHEDULER ATTRIBUTES
default-executor-program: $SYSTEM.SYSTEM.TACL
> BATCHCOM $ZBAT; SUBMIT JOB X, IN EP
Job X job number 323 submitted
> BATCHCOM $ZBAT; INFO JOB X, EXECUTOR-PROGRAM
JOB ATTRIBUTES for X
jobnumber: 323
executor-program: $SYSTEM.SYSTEM.TACL
> STATUS *, USER
Process Pri … Program file Hometerm
$EP3 0,53 149 … FUP $ZBAT
$EP2 0,54 149 … FUP $NULL
$EP1 0,63 149 … FUP \QA.$TRM2.#A
$Y561 0,65 150 … TACL $ZBAT
$C4 1,31 150 … TACL $T4.#A
$C4 B 2,26 150 … TACL $T4.#A
```

EXTSWAP Job Attribute

The EXTSWAP job attribute specifies the name of the swap file for the default extended data segment of a job's executor-program process. The file provides space for memory swaps of the default extended data segment during process execution.

```
EXT[SWAP]BBC{( AALVS1($volume-name, file-name) )
```

volume-name

is the name of the volume where the executor-program process is to create a temporary swap file. When the process terminates, the operating system automatically purges the file.

file-name

is the name of a swap file for the default extended data segment. The file must be on the same node as the executor-program process and can only be used by one process at a time. The file is not purged when the executor-program process terminates.

BATCHCOM expands a partially qualified file name by using the defaults specified in the last VOLUME command. If there was no such command, the defaults current at the start of the session apply.

Considerations

- A job submitted without the EXTSWAP attribute creates its extended swap file on the volume specified by the SWAP attribute of DEFINE =_DEFAULTS. If the DEFINE does not have the SWAP attribute, the operating system selects a volume for the file.
- An executor-program process does not use the specified swap file unless the process lets the operating system allocate space for memory swaps of the default extended data segment.
- The scheduler places the value of the EXTSWAP attribute in the *ext-swap-file* parameter of the PROCESS_CREATE_ procedure. For more information about the procedure, see the *Guardian Procedure Calls Reference Manual.*

Example

This example shows the EXTSWAP attribute specifying the extended swap file for a user-written program:

```
> BATCHCOM $ZBAT; SUBMIT JOB EXTSWAP-JOB, EXECUTOR-PROGRAM
$DATA7.OBJFILES.OBJECT, EXTSWAP $TRASH.ZBAT.EXTSWAP
Job EXTSWAP-JOB Jobnumber 5 submitted
> BATCHCOM $ZBAT; INFO JOB 5, EXTSWAP
JOB ATTRIBUTES for EXTSWAP-JOB
```

```
jobnumber: 5
extswap: \MELBDEV.$TRASH.ZBAT.EXTSWAP
> STATUS *, GMOMJOBID $ZBAT.5, DETAIL .
.
Current Extended Swap File Name: $TRASH.ZBAT.EXTSWAP .
.
GMOMJOBID: $ZBAT.5
```

HIGHPIN Job Attribute

The HIGHPIN job attribute determines whether a job's executor-program process runs at a low PIN or at a high PIN. The attribute is available only in D20 or later versions of the NetBatch product.

```
HIG[HPIN] { OF[F] | ON }
```

`OFF`

causes the executor-program process to run at a low PIN (a PIN in the range 0 through 254).

`ON`

causes the executor-program process to run at a high PIN (a PIN in the range 256 through 65535) when both these conditions exist:

- A high PIN is available in the CPU of the job's executor.
- The executor program has high PIN capabilities. That is, the program has been specifically written or compiled to run at a high PIN. (TACL, compiler, and BINDER options exist to control PIN assignment and to indicate server support of high PIN requester processes.)

Considerations

- A job submitted without the HIGHPIN attribute adopts the DEFAULT-HIGHPIN scheduler attribute. To display the value of this attribute, use the INFO SCHEDULER command.
- The operating system generates a name for the executor-program process of a job submitted without the NAME attribute. A low PIN process is given a four-character name and a high PIN process is given a five-character name.
- The scheduler plays no part in determining the PIN of processes created by an executor program (the operating system performs this function).
- For information about NetBatch D-series software versus C-series software compatibility, see Section 1, NetBatch Introduction.

Example

This example shows the HIGHPIN job attribute making the scheduler run a job's executor-program process at a high PIN (PIN 267):

```
> BATCHCOM $ZBAT; SUBMIT JOB FILES, IN FILES, EXECUTOR-
PROGRAM TACL, HIGHPIN ON
Job FILES job number 399 submitted
> STATUS *, GMOMJOBID $ZBAT.399
Process … Program file Hometerm
$Z0279 1,267 … $SYSTEM.SYS00.TACL $ZBAT
```

HOLD Job Attribute

The HOLD job attribute determines whether a job is available for execution.

```
HOLD { OF[F] | ON }
```

`OFF`

makes the job available for execution.

`ON`

prevents the scheduler from executing the job. A job with the HOLD ON attribute is on hold and has a state of SPECIAL-1.

Considerations

- The HOLD attribute prevents a job from running. The HOLDAFTER attribute allows the job to run, but puts it on hold after it has run.
- The scheduler treats a job submitted without the HOLD attribute like a job with the attribute HOLD OFF.
- Assigning the attribute HOLD ON to an executing, over-limit, or suspended nonrecurrent job has no effect on the job. (A nonrecurrent job is one whose attributes do not include CALENDAR or EVERY.)
- Assigning the attribute HOLD ON to an executing, over-limit, or suspended recurrent job does not affect the job until it recurs. (A recurrent job has the attribute CALENDAR or EVERY.) The state of the job when it does recur is SPECIAL-1, not TIME.
- These considerations apply to recurrent jobs:
  - A job on hold whose attributes include EVERY accumulates a run backlog if held for longer than the interval the attribute specifies and the scheduler has the attribute CATCHUP ON. When you change the job's HOLD attribute to OFF, the job runs repeatedly until the backlog clears.

- A job on hold whose attributes include CALENDAR does not accumulate a run backlog while held. When you change the job's HOLD attribute to OFF, the job runs again at the next future CALENDAR time.

  For more information about run backlogs, see ACTIVATE JOB Command on page 6-32.

- To make a job in the SPECIAL-1 state eligible for execution, alter its HOLD attribute from ON to OFF.

Example

This example shows the effect of the HOLD attribute:

```
12} SUBMIT JOB X, HOLD ON
Job X job number 406 submitted
13} STATUS JOB X
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
406 X 205,70 1:Hold DEFAULT
14} INFO JOB X, HOLD
JOB ATTRIBUTES for X
jobnumber: 406
hold: On
15} ALTER JOB X, HOLD OFF
Job X job number 406 altered
16} STATUS JOB X
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
406 X 205,70 269 EXECUTING DEFAULT
```

HOLDAFTER Job Attribute

The HOLDAFTER job attribute determines whether the scheduler puts a job on hold when it finishes executing.

```
HOLDA[FTER] { OF[F] | ON }
```

`OFF`

prevents the scheduler from putting the job on hold when it finishes executing.

`ON`

causes the scheduler to put the job on hold when it finishes executing. A job on hold has a state of SPECIAL-1 and the attribute HOLD ON.

Considerations

- The HOLDAFTER attribute allows a job to run but puts it on hold after it has run. The HOLD attribute prevents the job from running.

- The scheduler treats a job submitted without the HOLDAFTER attribute like a job with the attribute HOLDAFTER OFF.
- The scheduler puts a recurring job with the HOLDAFTER ON attribute in the SPECIAL-1 state, not the TIME state, after execution finishes. (A recurring job has the CALENDAR or EVERY attribute.)
- To make a job in the SPECIAL-1 state eligible for execution, alter its HOLD attribute from ON to OFF.

Example

This example shows the scheduler putting a job with the HOLDAFTER ON attribute on hold when the job finishes executing:

```
28} SUBMIT JOB X, HOLDAFTER ON
Job X job number 401 submitted
29} INFO JOB X, HOLD, HOLDAFTER
JOB ATTRIBUTES for X
jobnumber: 401
hold-after: On
30} STATUS JOB X
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
401 X 205,70 226 EXECUTING DEFAULT
31} STATUS JOB X
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
401 X 205,70 226 1:Hold DEFAULT
32} INFO JOB X, HOLD, HOLDAFTER
JOB ATTRIBUTES for X
jobnumber: 401
hold: On
hold-after: On
```

IFFAILS Job Attribute

The IFFAILS job attribute determines whether the scheduler reschedules a recurrent job that fails during execution. (A recurrent job is a job with the CALENDAR or EVERY attribute.) The IFFAILS attribute works in combination with the job's RESTART, STALL, and STOP-ON-ABEND attributes (see Table 4-5, Job States on Process Failure, on page 4-28).

```
IF[FAILS] { OF[F] | ON }
```

`OFF`

causes the scheduler to treat the job according to:

- The job's RESTART, STALL, and STOP-ON-ABEND attributes
- Whether the job is recurrent or nonrecurrent

- The cause of the failure (for example, the job's executor-program process abends)

For more information, see Table 4-5, Job States on Process Failure, on page 4-28, RESTART Job Attribute on page 7-95, STALL Job Attribute on page 7-102, and STOP-ON-ABEND Job Attribute on page 7-105.

`ON`

causes the scheduler to reschedule the job if it fails and is recurrent, depending on the values of the job's RESTART, STALL, and STOP-ON-ABEND attributes. For more information, see Table 4-5, Job States on Process Failure, on page 4-28.

Considerations

- The scheduler treats a job submitted without the IFFAILS attribute like a job with the attribute IFFAILS OFF.
- The IFFAILS attribute has no effect on nonrecurrent jobs (jobs whose attributes do not include CALENDAR or EVERY).
- For help when setting the IFFAILS, RESTART, STALL, and STOP-ON-ABEND attributes, see the decision chart in Section 4, Job Planning, Submission, and Management.

Example

This example shows the effect of the IFFAILS attribute on two recurrent jobs whose executor programs abend. The jobs have the same RESTART, STALL, and STOP-ON-ABEND attributes but different IFFAILS attributes. Job X has the IFFAILS OFF attribute, so the scheduler puts the job in the SPECIAL-6 state after failure. Job Y has the IFFAILS ON attribute, which causes the scheduler to put the job in the TIME state after failure.

```
> TIME
November 5, 1993 15:20:43
> FUP COPY DELAY05
DELAY 5 MINS
> BATCHCOM $ZBAT; SUBMIT JOB X, IN DELAY05, EVERY 1 DAYS,
IFFAILS OFF, RESTART OFF, STALL OFF, STOP-ON-ABEND ON
Job X job number 432 submitted
> BATCHCOM $ZBAT; SUBMIT JOB Y, LIKE X, IFFAILS ON
Job Y job number 433 submitted
> BATCHCOM $ZBAT; STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
432 X 205,70 399 EXECUTING DEFAULT
433 Y 205,70 401 EXECUTING DEFAULT
> STATUS *, GMOMJOBID $ZBAT.432
Process … Program file Hometerm
… $SYSTEM.SYS00.DELAY $ZBAT
```

```
$Z585 … $SYSTEM.SYS00.TACL $ZBAT
> STATUS *, GMOMJOBID $ZBAT.433
Process … Program file Hometerm
… $SYSTEM.SYS00.DELAY $ZBAT
$Z586 … $SYSTEM.SYS00.TACL $ZBAT
> STOP $Z585
> STOP $Z586
> BATCHCOM $ZBAT; STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
432 X 205,70 399 6:Fail RsOffDEFAULT
433 Y 205,70 401 06NOV93 DEFAULT
```

IN Job Attribute

The IN job attribute specifies the name of a job's input file.

```
IN [ file-name ]
```

file-name

is the name of an input file. The file can be a device, a process, or a disk file. If you omit *file-name*, the scheduler passes spaces as the input-file name. (The effect on the job of having spaces passed as the input-file name depends on the job's executor program.) Specifying a nonexistent file causes the scheduler to generate this warning message but does not prevent the scheduler from executing the job:

```
0513-W IN file does not exist; create and secure as required
```

BATCHCOM expands a partially qualified file name by using the defaults specified in the last VOLUME command. If there was no such command, the defaults current at the start of the session apply.

You can create a job's input file any time after submission and before execution if either of these conditions exists:

- The job is in the READY state.
- The job has the HOLD or WAITON attribute, or a time attribute (AFTER, AT, CALENDAR, EVERY, or WAIT).

Specifying a nonexistent disk file as an input file prevents all users except the job owner from accessing the job.

Considerations

- Input-file commands must be compatible with a job's executor program. Otherwise the job abends.

- Specifying an input file to which other users have write access causes the scheduler to generate this warning message:

  ```
  0512-W Other users can WRITE or PURGE the IN file;
  resecure if required
  ```

  Users with write access to your job input file can alter any job attribute or delete a job using the file. Using the input file as a medium, these users also can assume your level of security. As a result, they could modify the input file to purge your files, change your password, and so on. To avoid compromising system security, secure input files by means of the Safeguard distributed security management facility or the Guardian standard security system.

  For information about Safeguard security, see the *Safeguard Reference Manual.* For information about Guardian security, see the *Guardian User's Guide*.

- To remove the IN attribute from a job, use the ALTER JOB command's REMOVE IN qualifier. For more information, see ALTER JOB Command on page 6-62.

- A job without the IN attribute uses the scheduler as its input file. If the job reads the file, it encounters file-system error 2 (operation not allowed). If the job writes the file, it writes to the scheduler's log file.

Example

This example shows the use of the IN attribute:

```
> FUP COPY INFILE
FILES $DATA7.ZBAT
> BATCHCOM $ZBAT; SUBMIT JOB X, EXECUTOR-PROGRAM TACL, IN
INFILE, OUT OUTFILE
Job X job number 424 submitted
> BATCHCOM $ZBAT; INFO JOB X, IN
JOB ATTRIBUTES for X
jobnumber: 424
in: $DATA7.NB.INFILE
> FUP COPY OUTFILE
TACL (T9205Dxx - DDMMMYY), Operating System Dxx .
.
FILES $DATA7.ZBAT
$DATA7.ZBAT
ATTACH ATTACH0 BATCHCTL CHKQUE CHKQUE0 EXECUTO0
EXECUTOR JOB JOBCLAS0 JOBCLASS LOGAAA NBATTX
NBATTX0
```

INITIATION Class Attribute

The INITIATION class attribute specifies whether jobs from the class are available for execution by the scheduler.

```
INI[TIATION] { OF[F] | ON }
```

`OFF`

specifies jobs from the class are unavailable for execution.

`ON`

specifies jobs from the class are available for execution.

Consideration

The scheduler assigns the attribute INITIATION ON to a class added without the INITIATION attribute.

Examples

- This example shows the ADD CLASS command adding a class to a scheduler. Note the adoption by the class of the default attribute INITIATION OFF.

```
67} CHANGEUSER 255,205 psswrd
67} ADD CLASS CL1
Class CL1 added
68} INFO CLASS CL1
CLASS ATTRIBUTE for CL1
initiation: On
```

- This example shows a class's INITIATION OFF attribute preventing the scheduler from executing jobs from the class. The example also shows the effect of altering the attribute to INITIATION ON.

```
18} STATUS EXECUTOR *
EXECUTOR CPU STATE JOB CLASS
------------------------ --- ------ ---- -----
E1 0 ON
E2 1 ON
19} INFO EXECUTOR *, CLASS
EXECUTOR ATTRIBUTES for E1
classes: DEFAULT
EXECUTOR ATTRIBUTES for E2
classes: DEFAULT
20} INFO CLASS DEFAULT
CLASS ATTRIBUTE for DEFAULT
initiation: Off
21} SUBMIT JOB JOB-A, CLASS DEFAULT
Job JOB-A job number 390 submitted
22} SUBMIT JOB JOB-B, CLASS DEFAULT
Job JOB-B job number 391 submitted
```

```
23} STATUS JOB *
JOB JOBNAME USERID LOG STATE
CLASSNAME
---- ---------------------- ------- ---- ---------- -----
390 JOB-A 255,205 READY DEFAULT
391 JOB-B 255,205 READY DEFAULT
24} ALTER CLASS DEFAULT, INITIATION ON
Class DEFAULT altered
25} STATUS JOB *
JOB JOBNAME USERID LOG STATE
CLASSNAME
---- ---------------------- ------- ---- ---------- -----
390 JOB-A 255,205 141 EXECUTING DEFAULT
391 JOB-B 255,205 142 EXECUTING DEFAULT
26} STATUS EXECUTOR *
EXECUTOR CPU STATE JOB CLASS
---------------------- --- ------ ---- ---------
E1 0 ACTIVE 390 DEFAULT
E2 1 ACTIVE 391 DEFAULT
```

INITIATION Scheduler Attribute

The INITIATION scheduler attribute enables or disables job startup by the scheduler. The attribute lets you prevent jobs from starting after a scheduler warm start and prevent jobs from starting in a running scheduler. The attribute does not prevent users from submitting jobs and does not affect executing or over-limit jobs.

```
INI[TIATION] { OF[F] | ON }
```

`OFF`

prevents the scheduler from starting submitted jobs.

`ON`

lets the scheduler start submitted jobs.

Considerations

- A scheduler adopts the attribute INITIATION ON by default when cold started.
- To prevent jobs from starting after a scheduler warm start, set the scheduler's INITIATION attribute to OFF before entering the START SCHEDULER command. When the scheduler starts and after reviewing your jobs, set the attribute to ON.

Example

This example shows the effect of the INITIATION scheduler attribute. When set to OFF, the attribute disables job startup. When set to ON, the attribute enables job startup.

```
40} STATUS JOB (ZBAT*)
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
7 ZBAT-0007 255,255 READY DEFAULT
8 ZBAT-0008 255,255 READY ADMIN
9 ZBAT-0009 255,255 READY SALES
10 ZBAT-0010 255,255 READY ACCOUNTS
41} INFO SCHEDULER, INITIATION
SCHEDULER ATTRIBUTES
initiation: Off
42} ALTER SCHEDULER, INITIATION ON
Scheduler altered
43} STATUS JOB (ZBAT*)
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
7 ZBAT-0007 255,255 127 EXECUTING DEFAULT
8 ZBAT-0008 255,255 128 EXECUTING ADMIN
9 ZBAT-0009 255,255 129 EXECUTING SALES
10 ZBAT-0010 255,255 130 EXECUTING ACCOUNTS
```

JOB-LOG Job Attribute

The JOB-LOG job attribute directs log-file events for a job to a specified file.

```
J[OB]-L[OG] [ log-file ]
```

log-file

is the name of a job log file. The file can be a device, a process, or any type of unstructured disk file except an EDIT file. For a disk file, these conditions apply:

- If the file exists and is not an EDIT file, the scheduler opens it and appends events. If the file is an EDIT file, the command fails. If the file does not exist, the scheduler creates it with a file code of 847.
- Only one process at a time can write to a disk log file.

BATCHCOM expands a partially qualified file name by using the defaults specified in the last VOLUME command. If there was no such command, the defaults current at the start of the session apply.

Specifying JOB-LOG without *log-file* prevents the creation of a job log file. Specifying a spooler location in *log-file* for a job whose OUT attribute specifies a spooler collector process separates the log file from other spooler output of the job. (JOB-LOG does this by making the spooler assign a different batch number to the log file.) The location can be the same as or different from that specified by the OUT attribute—the effect is the same.

For a job with the ATTACHMENT-SET attribute, *log-file* can specify a DEFINE from the attachment set.

Considerations

- The scheduler creates a spooler-job log file for a job without the JOB-LOG attribute if the job's OUT attribute specifies a spooler collector process. The scheduler does not create a log file if the OUT attribute specifies a device, a disk file, or a nonspooler collector process.
- To remove the JOB-LOG attribute from a job, use the ALTER JOB command's REMOVE JOB-LOG qualifier. For more information, see ALTER JOB Command on page 6-62.

Examples

- This example shows the submission and execution of a job whose OUT attribute specifies a nonspooler process. The JOB-LOG attribute directs the job's log-file output to a terminal. Without the attribute, the job would not have a log file.

```
> BATCHCOM $ZBAT; SUBMIT JOB, EXECUTOR-PROGRAM TACL, IN
INFILE,
OUT $NULL, JOB-LOG [#MYTERM]
Job ZBAT-0016 Jobnumber 16 submitted
> BATCHCOM $ZBAT; INFO JOB 16, OUT, JOB-LOG
JOB ATTRIBUTES for ZBAT-0016
jobnumber: 16
out: \MELBDEV.$NULL
job-log: \MELBDEV.$ZTN0.#PTY4
> PAUSE
21SEP94 11:17:20 BEGIN JOB (SUPER.SUPER)ZBAT-0016:1 E_E1 … .
.
```

- This example shows two similar jobs that send their output to a spooler. The first job does not have the JOB-LOG attribute, so it has a log file and an executor-program output file created in the spooler (spooler jobs 80 and 81 respectively). The second job uses the JOB-LOG attribute to prevent log-file creation, so it has an executor-program output file only (spooler job 82).

```
4} SUBMIT JOB, EXECUTOR-PROGRAM TACL, IN INFILE, OUT $S.#RUN1
Job ZBAT-0017 Jobnumber 17 submitted
5} INFO JOB 17, OUT, JOB-LOG
JOB ATTRIBUTES for ZBAT-0017
jobnumber: 17
out: \MELBDEV.$S.#RUN1
6} SUBMIT JOB, EXECUTOR-PROGRAM TACL, IN INFILE, OUT
$S.#RUN2,
JOB-LOG
Job ZBAT-0018 Jobnumber 18 submitted
7} INFO JOB 18, OUT, JOB-LOG
JOB ATTRIBUTES for ZBAT-0018
jobnumber: 18
out: \MELBDEV.$S.#RUN2
```

```
job-log:
8} RUN PERUSE
PERUSE - T9101D20 - (01JUN93) SYSTEM \MELBDEV
SPOOLER SUPERVISOR IS \MELBDEV.$SPLS
JOB BATCH STATE PAGES COPIES PRI HOLD LOCATION … .
.
80 48 READY 2 1 4 #RUN1 …
81 48 READY 1 1 4 #RUN1 …
82 49 READY 1 1 4 #RUN2 … .
.
```

- This example shows two similar jobs that send their output to a spooler. The first job does not have the JOB-LOG attribute, so the spooler batches together the job's log file and executor-program output file. The second job uses the JOB-LOG attribute to separate the log file from other output, so the spooler assigns different batch numbers to the log and executor-program output files.

```
10} SUBMIT JOB, EXECUTOR-PROGRAM TACL, IN INFILE, OUT
$S.#RUN3
Job ZBAT-0019 Jobnumber 19 submitted
11} INFO JOB 19, OUT, JOB-LOG
JOB ATTRIBUTES for ZBAT-0019
jobnumber: 19
out: \MELBDEV.$S.#RUN3
12} SUBMIT JOB, EXECUTOR-PROGRAM TACL, IN INFILE, OUT
$S.#RUN4,
JOB-LOG $S.#RUN4LOG
Job ZBAT-0020 Jobnumber 20 submitted
13} INFO JOB 20, OUT, JOB-LOG
JOB ATTRIBUTES for ZBAT-0020
jobnumber: 20
out: \MELBDEV.$S.#RUN4
job-log: \MELBDEV.$S.#RUN4LOG
14} RUN PERUSE
PERUSE - T9101D20 - (01JUN93) SYSTEM \MELBDEV
SPOOLER SUPERVISOR IS \MELBDEV.$SPLS
JOB BATCH STATE PAGES COPIES PRI HOLD LOCATION … .
.
85 52 READY 2 1 4 #RUN3 …
86 52 READY 1 1 4 #RUN3 …
87 53 READY 1 1 4 #RUN4LOG …
88 54 READY 1 1 4 #RUN4 … .
.
```

JOBID-ZERO Job Attribute

The JOBID-ZERO job attribute determines whether the scheduler assigns a GMOMJOBID to a job's executor-program process. The job executes under the scheduler's control if assigned a GMOMJOBID and outside that control if not.

```
J[OBID]-Z[ERO] { OF[F] | ON }
```

`OFF`

makes the scheduler assign a GMOMJOBID to the job's executor-program process. As a result, the job executes under the scheduler's control.

`ON`

prevents the scheduler from assigning a GMOMJOBID to the job's executor-program process. As a result, the job executes outside the scheduler's control.

These considerations also apply:

- The scheduler does not create a log file for the job.
- Processes of the job cannot access the scheduler as a home terminal. For this reason, use the TERM attribute to specify a home terminal for the job. For more information, see TERM Job Attribute on page 7-114.
- The job is complete after the scheduler sends the startup message.

Considerations

- The scheduler assigns the attribute JOBID-ZERO OFF to a job submitted without the JOBID-ZERO attribute.
- The scheduler places 0 in the *jobid* parameter of the PROCESS_CREATE_ procedure when JOBID-ZERO is set to ON and places the job number in the parameter when JOBID-ZERO is set to OFF. For more information about the procedure, see the *Guardian Procedure Calls Reference Manual*.

Example

This example shows the effect of the JOBID-ZERO attribute. The scheduler assigns a GMOMJOBID to the job with the attribute set to OFF, but not to the job with the attribute set to ON. Note the use of the TERM attribute to direct home-terminal output away from the scheduler.

```
> BATCHCOM $ZBAT; SUBMIT JOB GMOMJOBID, EXECUTOR-PROGRAM
DELAY,
STARTUP "1 MINS", NAME $YES, JOBID-ZERO OFF
Job GMOMJOBID Jobnumber 1 submitted
> STATUS $YES, DETAIL .
.
```

```
Myterm: $ZBAT .
.
GMOMJOBID: $ZBAT.1
> BATCHCOM $ZBAT; SUBMIT JOB NO-GMOMJOBID, EXECUTOR-PROGRAM
DELAY, STARTUP "1 MINS", NAME $NO, JOBID-ZERO ON, TERM
\MELBDEV.$ZTN0.#PTY6
Job NO-GMOMJOBID Jobnumber 2 submitted
> STATUS $NO, DETAIL .
.
Myterm: $ZTN0.#PTY6 .
.
GMOMJOBID:
```

LIB Job Attribute

The LIB job attribute specifies the name of the user library file for a job's executor program.

```
LIB [ file-name ]
```

file-name

specifies the name of the user library file. The file must be on the same node as the executor-program process.

BATCHCOM expands a partially qualified file name by using the defaults specified in the last VOLUME command. If there was no such command, the defaults current at the start of the session apply.

If you specify LIB without `file-name` in an ALTER JOB command, the scheduler removes the LIB attribute from the job. Specifying LIB without `file-name` in a SET JOB command makes BATCHCOM remove the LIB attribute from the working-attributes set.

Considerations

- The executor program of a job submitted without the LIB attribute uses the library file specified when the program last ran. The program runs without a library file if that is how it ran previously. The current library file applies if the program is already running.
- The owner of a job with the LIB attribute must have write access to the job's executor program for the job to start. Without write access, the job fails on startup and goes into the SPECIAL-3 state.
- Any number of processes can share a library file, but each must have the same user-library configuration to avoid library-conflict errors.
- To remove the library file from an executor program, use the ALTER JOB command's REMOVE LIB qualifier. (Specifying LIB without a file name in a TACL

RUN command has the same effect.) For more information, see ALTER JOB Command on page 6-62.

- The scheduler places the value of the LIB attribute in the *library-file* parameter of the PROCESS_CREATE_ procedure. For more information about the procedure, see the *Guardian Procedure Calls Reference Manual*.

Example

This example shows the LIB attribute specifying the library file for a user-written program:

```
> BATCHCOM $ZBAT; SUBMIT JOB SCHDCOLD, EXECUTOR-PROGRAM
$TRASH.OBJS.OBJ9409, LIB $TRASH.LIBS.OBJLIB, IN
$TRASH.INFILES.SCHDCOLD
Job SCHDCOLD Jobnumber 33 submitted
> STATUS *, GMOMJOBID $ZBAT.33, DETAIL .
.
Library File Name: $TRASH.LIBS.OBJLIB .
.
GMOMJOBID: $ZBAT.33
```

LIMIT Job Attribute

The LIMIT job attribute specifies an execution time limit for a job. On expiration of the limit, the job continues to execute, but its state changes from EXECUTING to OVER LIMIT.

```
LIM[IT] hours [ : mins ]
```

hours

is a number in the range 0 through 999 specifying the hours limit.

mins

is a number in the range 0 through 59 specifying the minutes limit. The default is 0.

Considerations

- The scheduler treats a job submitted without the LIMIT attribute like a job with the attribute LIMIT 0:0.
- To remove the LIMIT attribute from a job, specify LIMIT 0:0 in an ALTER JOB command.
- A log-file event in this form appears in the scheduler and job log files when a job exceeds its execution time limit:

  ddmmmyy hh: *mm*: *ss* LIMIT Exceeded J_ *job-number*

- A scheduler with EMS event-message generation enabled generates event message 204 (ZBAT-EVT-JOB-OVER-LIMIT) when a job exceeds its execution time limit.

  For message details, see the *NetBatch Management Programming Manual.*

Example

This example shows the submission and execution of a job with the LIMIT attribute. The example shows the job's state change when the specified limit expires and the log-file events that record the job's execution history. Job execution continues after expiration of the limit.

```
10} SUBMIT JOB LIMIT-TEST, EXECUTOR-PROGRAM DELAY, STARTUP "2
MINS", LIMIT 0:01
Job LIMIT-TEST Jobnumber 14 submitted
11} STATUS JOB 14
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
14 LIMIT-TEST 205,70 4334 EXECUTING DEFAULT
12} STATUS JOB 14
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
14 LIMIT-TEST 205,70 4334 OVER LIMIT DEFAULT
13} STATUS-HISTORY 14
2099-E JOB does not exist
16SEP94 11:48:25 BEGIN JOB (FPP.USER)LIMIT-TEST:1 E_E1
L_4334 J_14 P_DELAY
\MELBDEV.$X158:16504109 U_205,70
16SEP94 11:48:26 LIST JOB LIMIT-TEST EXECUTING J_14
16SEP94 11:48:26 START EXECUTOR-PROGRAM U_205,70 J_14
P_DELAY \MELBDEV.$X158:16504109
16SEP94 11:49:26 LIMIT Exceeded J_14
16SEP94 11:50:26 STOP CC_0 EXECUTOR-PROGRAM J_14
\MELBDEV.$X158:16504109
16SEP94 11:50:27 FINISH JOB LIMIT-TEST T_0:0:0:13 J_14
P_DELAY
Output 6 Lines
```

LOCALNAMES Scheduler Attribute

The LOCALNAMES scheduler attribute makes the scheduler treat jobs submitted from licensed requesters on specified nodes as local jobs, not as remote jobs. (An example of such a requester is NetBatch-Plus.) The remotely submitted jobs gain, through the scheduler, the same access privileges on the scheduler's node as locally submitted jobs.

```
LO[CALNAMES]BBC[( AALVS1(\remote-node,( \remote-node [ ,
\remote-node ]… )))
```

remote-node

is the name of a node remote to the scheduler. LOCALNAMES can specify up to 30 remote nodes.

Specifying LOCALNAMES without `remote-node` in an ALTER SCHEDULER command removes the LOCALNAMES attribute from the scheduler.

△ **Caution.** Before specifying the LOCALNAMES attribute, consider these security-related issues:

- You must license remote-node requester programs before using them to submit "local" jobs.
- A remote-node user with access by way of a licensed requester to the scheduler's node gains local access to that node through the scheduler. For example, the remote-node super ID (255,255) could have the same access privileges on the scheduler's node as the local super ID.
- The file security of remote-node requester programs must restrict write and purge access to the programs to local users. (The recommended minimum file security is "A-A-".) This requirement ensures that requests can be made from remote-node only. To prevent unauthorized access to the scheduler's node, make sure the super ID owns the programs and secures them appropriately.

Considerations

- The LOCALNAMES attribute can be set only by the local super ID (255,255).
- A RUN NETBATCH cold-start command that includes the *remote-node* parameter sets the LOCALNAMES attribute to the specified nodes. For example:

```
> NETBATCH /NAME $ZBAT, NOWAIT/ $DATA7.ZBAT ! \DEV \PROD
> BATCHCOM $ZBAT; INFO SCHEDULER, LOCALNAMES
SCHEDULER ATTRIBUTES
localnames: \DEV
\PROD
```

- If the command does not include the *remote-node* parameter, the scheduler starts without the LOCALNAMES attribute; for example:

```
> NETBATCH /NAME $ZBAT, NOWAIT/ $DATA7.ZBAT !
> BATCHCOM $ZBAT; INFO SCHEDULER, LOCALNAMES
SCHEDULER ATTRIBUTES
```

- A scheduler without the LOCALNAMES attribute treats remotely submitted jobs as remote jobs subject to normal NonStop system remote-access restrictions.
- You cannot selectively add a node to or delete a node from the LOCALNAMES node list. To add or delete a node, you must reenter the complete set of nodes in an ALTER SCHEDULER command.
- To understand the effect of the LOCALNAMES attribute, consider the example of a job submitted to a local scheduler by a remote NetBatch-Plus requester. The job fails during startup (SPECIAL-3 state) if its executor program is local to the scheduler, the program's security is "AAAA," and the scheduler does not have the LOCALNAMES attribute. Conversely, the job starts successfully if the LOCALNAMES attribute specifies the NetBatch-Plus requester's node.

Examples

- This example shows the failure of a remote job submitted to a local scheduler that does not have the LOCALNAMES attribute. The scheduler could not create the job's executor-program process because the remote user did not have access to the executor program.

```
> FILEINFO \REMOTE.$TRASH.OBJECTS.BATCHCOM
\REMOTE.$TRASH.OBJECTS
Code … Owner RWEP …
BATCHCOM 100L … 255,255 "A-A-" …
> FILEINFO \LOCAL.$DATA7.PURGE.*
\LOCAL.$DATA7.PURGE
Code … Owner RWEP …
INFILE 101 … 255,255 "AAAA" …
TACL 100L … 255,255 "AAAA" …
> \REMOTE.$TRASH.OBJECTS.BATCHCOM
BATCHCOM - T9190D30 - (31OCT94^01JUN94)
(C)1986 Tandem (C)2004 Hewlett Packard Development Company,
L.P.
1} OPEN \LOCAL.$ZBAT
NETBATCH SERVER - T9190D30 …
2} INFO SCHEDULER, LOCALNAMES
SCHEDULER ATTRIBUTES
3} SUBMIT JOB, EXECUTOR-PROGRAM \LOCAL.$DATA7.PURGE.TACL, IN
\LOCAL.$DATA7.PURGE.INFILE
Job ZBAT-0004 Jobnumber 4 submitted
0527-W You have no READ access to IN file
4} STATUS JOB 4
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
```

```
---- ---------------- ------- ---- ----------- ---------
4 ZBAT-0004 255,255 122 3:NEWP. err DEFAULT
```

- This example shows the same command sequence as that in the previous example, but the job starts successfully because the scheduler has the LOCALNAMES attribute:

```
> FILEINFO \REMOTE.$TRASH.OBJECTS.BATCHCOM
\REMOTE.$TRASH.OBJECTS
Code … Owner RWEP …
BATCHCOM 100L … 255,255 "A-A-" …
> FILEINFO \LOCAL.$DATA7.PURGE.*
\LOCAL.$DATA7.PURGE
Code … Owner RWEP …
INFILE 101 … 255,255 "AAAA" …
TACL 100L … 255,255 "AAAA" …
> \REMOTE.$TRASH.OBJECTS.BATCHCOM
BATCHCOM - T9190D30 - (31OCT94^01JUN94)
(C)1986 Tandem (C)2004 Hewlett Packard Development Company,
L.P.
1} OPEN \LOCAL.$ZBAT
NETBATCH SERVER - T9190D30 …
2} INFO SCHEDULER, LOCALNAMES
SCHEDULER ATTRIBUTES
localnames: \REMOTE
3} SUBMIT JOB, EXECUTOR-PROGRAM \LOCAL.$DATA7.PURGE.TACL, IN
\LOCAL.$DATA7.PURGE.INFILE
Job ZBAT-0005 Jobnumber 5 submitted
0512-W Other users can WRITE or PURGE the IN file; resecure
if
required
4} STATUS JOB 5
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
5 ZBAT-0005 255,255 123 EXECUTING DEFAULT
```

MAX-CONCURRENT-JOBS Scheduler Attribute

The MAX-CONCURRENT-JOBS scheduler attribute specifies the maximum number of concurrent jobs and temporary executors for the scheduler.

```
M[AX]-[CONCURRENT]-[JOBS] max-concurrent-jobs
[ , max-temporary-executors ]
```

max-concurrent-jobs

is a number in the range 0 through 500 specifying the maximum number of jobs that can run concurrently.

max-temporary-executors

is a number in the range 0 through 500 specifying the maximum number of temporary executors that the scheduler can create. (The scheduler creates

temporary executors for jobs with the AT attribute and for jobs affected by the RUNNOW JOB command. For more information, see AT Job Attribute on page 7-15 and RUNNOW JOB Command on page 6-128.)

Considerations

- A scheduler adopts the attribute MAX-CONCURRENT-JOBS 500,500 by default when cold started.
- The scheduler can handle up to 500 concurrent jobs, depending on how many concurrent openers it has and on the openers limits imposed by other subsystems such as spooler subsystems. The scheduler can manage a maximum of 2000 concurrent opens from executor-program processes, child processes of executor-program processes, and BATCHCOM, NetBatch-Plus, or user-written requester processes.

Example

This example shows the effect of the attribute MAX-CONCURRENT-JOBS 2,2. The attribute prevents the scheduler from running more than two jobs at once and from creating more than two temporary executors.

```
79} STATUS JOB *
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
11 ZBAT-0011 255,255 READY DEFAULT
12 ZBAT-0012 255,255 READY DEFAULT
13 ZBAT-0013 255,255 READY DEFAULT
14 ZBAT-0014 255,255 READY DEFAULT
80} STATUS EXECUTOR *
2117-I No EXECUTOR selected
81} INFO SCHEDULER, MAX-CONCURRENT-JOBS
SCHEDULER ATTRIBUTES
max-concurrent-jobs: 2,2
82} RUNNOW JOB (ZBAT*)
Job ZBAT-0011 Jobnumber 11 runnow completed
Job ZBAT-0012 Jobnumber 12 runnow completed
Job ZBAT-0013 Jobnumber 13 runnow completed
Job ZBAT-0014 Jobnumber 14 runnow completed
83} STATUS JOB *
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
11 ZBAT-0011 255,255 132 EXECUTING DEFAULT
12 ZBAT-0012 255,255 133 EXECUTING DEFAULT
13 ZBAT-0013 255,255 RUNNOW DEFAULT
14 ZBAT-0014 255,255 RUNNOW DEFAULT
84} STATUS EXECUTOR *
EXECUTOR CPU STATE JOB CLASS
----------------------- --- ------ ---- -------
__TEMP_EXEC_11 3 DELETE 11 DEFAULT
__TEMP_EXEC_12 2 DELETE 12 DEFAULT
```

MAX-PRI Scheduler Attribute

The MAX-PRI scheduler attribute specifies an upper execution-priority limit for executor-program processes and processes they create. The attribute overrides the PRI job attribute, thereby enabling you to prevent processes from executing at a priority higher than appropriate for your system.

```
M[AX]-[PRI] number
```

number

is a number in the range 1 through 199 (1 is lowest) specifying the execution-priority limit.

Considerations

- A scheduler adopts the attribute MAX-PRI 199 by default when cold started.
- The scheduler alters the priority of a process to that specified by MAX-PRI when the priority returned in system message -112 (job process creation) exceeds the MAX-PRI value. Regardless, MAX-PRI exercises no further control over the priority.

Example

This example shows the effect of the MAX-PRI attribute on a job. The scheduler alters the priority of the job's executor-program process and the program's child processes to the priority specified by MAX-PRI. The attribute has no effect when the priority of the executor-program process is later changed with the TACL ALTPRI command.

```
> FUP COPY INFILE
DELAY /NAME $D1, NOWAIT, PRI 160/ 3 MINS
DELAY /NAME $D2, NOWAIT, PRI 170/ 3 MINS
DELAY /NAME $D3, PRI 180/ 3 MINS
3 RECORDS TRANSFERRED
> BATCHCOM $ZBAT; INFO SCHEDULER, MAX-PRI, DEFAULT-PRI
SCHEDULER ATTRIBUTES
max-pri: 100
default-pri: 90
> BATCHCOM $ZBAT; SUBMIT JOB, EXECUTOR-PROGRAM TACL, IN
INFILE, NAME $TEST, PRI 199
Job ZBAT-0017 Jobnumber 17 submitted
> BATCHCOM $ZBAT; STATUS JOB 17, DETAIL .
.
Program … Pid … Pri …
TACL … $TEST … 100 …
DELAY … $D3 … 100 …
DELAY … $D2 … 100 …
DELAY … $D1 … 100 …
> ALTPRI $TEST, 199
> BATCHCOM $ZBAT; STATUS JOB 17, DETAIL .
```

```
.
Program … Pid … Pri …
TACL … $TEST … 199 …
DELAY … $D3 … 100 …
DELAY … $D2 … 100 …
DELAY … $D1 … 100 …
```

MAXPRINTLINES Job Attribute

The MAXPRINTLINES job attribute specifies the maximum number of print lines for a job output file. The attribute is, in effect, a spooler-job attribute, so it only applies if the output file is a spooler collector process.

```
MAXPRINTL[INES] { number | NON[E] }
```

number

is a number in the range 120 through 65534 specifying the maximum number of print lines.

NONE

specifies no maximum.

Considerations

- A job submitted without the MAXPRINTLINES attribute adopts the DEFAULT-MAXPRINTLINES scheduler attribute. To display the value of this attribute, use the INFO SCHEDULER command.
- The scheduler passes the MAXPRINTLINES attribute to a job's executor-program process as a spool DEFINE named =_ZBAT_OUT if the job's OUT attribute specifies a spooler collector process.
- The MAXPRINTLINES attribute applies to the sum of the lines of a job's log file and the output of each process started by the job.
- If the job log fills, the spooler logs error 45 (file is full) to the scheduler log but continues processing the job.
- If an NBEXEC or TACL process is the executor program and the process output file is full, the scheduler stops the job. The scheduler also stops all processes started by the job. Whether the job continues executing when the program is not an NBEXEC or TACL process depends on how the process handles spooler error 45.
- The MAXPRINTPAGES and MAXPRINTLINES attributes can conflict with each other if each has a value other than NONE. (MAXPRINTPAGES overrides MAXPRINTLINES if the number of pages specified by MAXPRINTPAGES converts to fewer lines. For example, if a page equals 60 lines, MAXPRINTPAGES 8 (480 lines) overrides MAXPRINTLINES 600.) To avoid conflicting attributes, specify a

value for one of MAXPRINTPAGES and MAXPRINTLINES, and NONE for the other.

- For information on spooler jobs, see the manual for the spooler product you are using.

Example

This example shows the submission of a job whose output file is a spooler location. The example shows the sources of the job's MAXPRINTLINES and MAXPRINTPAGES attributes and the effect the attributes have on the output file.

```
> BATCHCOM $ZBAT; INFO SCHEDULER, DEFAULT-MAXPRINTLINES,
DEFAULT-MAXPRINTPAGES
SCHEDULER ATTRIBUTES
default-maxprintlines: 1000
default-maxprintpages: None
> BATCHCOM $ZBAT; SUBMIT JOB LINES, OUT $S.#LINES,
MAXPRINTLINES 180
Job LINES job number 255 submitted
> BATCHCOM $ZBAT; INFO JOB LINES, MAXPRINTLINES,
MAXPRINTPAGES
JOB ATTRIBUTES for LINES
jobnumber: 255
maxprintlines: 180
maxprintpages: None
> BATCHCOM $ZBAT; STATUS JOB LINES
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
255 LINES 205,70 7 EXECUTING DEFAULT
> SPOOLCOM
)JOB 7, STATUS DETAIL
JOB: 7
STATE: READY
LOCATION: #LINES .
.
MAXIMUM LINES: 180
MAXIMUM PAGES:
```

MAXPRINTPAGES Job Attribute

The MAXPRINTPAGES job attribute specifies the maximum number of print pages for a job output file. The attribute is, in effect, a spooler-job attribute, so it applies only if the output file is a spooler collector process.

```
MAXPRINTP[AGES] { number | NON[E] }
```

number

is a number in the range 2 through 65534 specifying the maximum number of print pages.

NONE

specifies no maximum.

Considerations

- A job submitted without the MAXPRINTPAGES attribute adopts the DEFAULT-MAXPRINTPAGES scheduler attribute. To display the value of this attribute, use the INFO SCHEDULER command.
- The scheduler passes the MAXPRINTLINES attribute to a job's executor-program process as a spool DEFINE named =_ZBAT_OUT if the job's OUT attribute specifies a spooler collector process.
- The MAXPRINTPAGES attribute applies to the sum of the pages of a job's log file and the output of each process started by the job.
- If the job log fills, the spooler logs error 45 (file is full) to the scheduler log but continues processing the job.
- If an NBEXEC or TACL process is the executor program and the process output file is full, the scheduler stops the job. The scheduler also stops all processes started by the job. Whether the job continues executing when the program is not an NBEXEC or TACL process depends on how the process handles spooler error 45.
- The MAXPRINTPAGES and MAXPRINTLINES attributes can conflict with each other if each has a value other than NONE. (MAXPRINTPAGES overrides MAXPRINTLINES if the number of pages specified by MAXPRINTPAGES converts to fewer lines. For example, if a page equals 60 lines, MAXPRINTPAGES 25 (1500 lines) overrides MAXPRINTLINES 1200.) To avoid conflicting attributes, specify a value for one of MAXPRINTPAGES and MAXPRINTLINES, and NONE for the other.
- For information on spooler jobs, see the manual for the spooler product you are using.

Example

This example shows the submission of a job whose output file is a spooler location. The example shows the sources of the job's MAXPRINTLINES and MAXPRINTPAGES attributes and the effect the attributes have on the output file.

```
> BATCHCOM $ZBAT; INFO SCHEDULER, DEFAULT-MAXPRINTLINES,
DEFAULT-MAXPRINTPAGES
SCHEDULER ATTRIBUTES
default-maxprintlines: None
default-maxprintpages: 25
> BATCHCOM $ZBAT; SUBMIT JOB PAGES, OUT $S.#PAGES,
MAXPRINTPAGES 50
Job PAGES job number 257 submitted
> BATCHCOM $ZBAT; INFO JOB PAGES, MAXPRINTLINES,
```

```
MAXPRINTPAGES
JOB ATTRIBUTES for PAGES
jobnumber: 257
maxprintlines: None
maxprintpages: 50
> BATCHCOM $ZBAT; STATUS JOB PAGES
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
257 PAGES 205,70 11 EXECUTING DEFAULT
> SPOOLCOM
)JOB 11, STATUS DETAIL
JOB: 11
STATE: READY
LOCATION: #PAGES .
.
MAXIMUM LINES:
MAXIMUM PAGES: 50
```

MEM Job Attribute

The MEM job attribute specifies the minimum number of 2048-byte memory pages allotted to a job's executor-program process for user data.

```
ME[M] number
```

number

is a number in the range 0 through 64 specifying the minimum number of 2048-byte memory pages.

Considerations

- A job submitted without the MEM attribute has memory pages allotted according to the memory-pages value compiled or bound into the executor program. This default also applies when the MEM attribute specifies a number less than that value.
- The scheduler places the value of the MEM attribute in the `memory-pages` parameter of the PROCESS_CREATE_ procedure. For more information on the procedure, see the *Guardian Procedure Calls Reference Manual*.

Example

This example shows the execution of a job whose executor program returns the number of memory pages specified for it by the MEM attribute:

```
> BATCHCOM /NOWAIT/ $ZBAT; SUBMIT JOB X, OUT [#MYTERM],
EXECUTOR-PROGRAM MEMTST, MEM 45
> PAUSE
Job X Jobnumber 9 submitted
MEM=45
```

NAME Job Attribute

The NAME job attribute specifies a name for a job's executor-program process.

```
NA[ME] $process-name
```

process-name

specifies a unique name for the executor-program process. The name can contain from one to five alphanumeric characters. The first character must be alphabetic. For network access and for processes that use the C-series OPEN procedure instead of the D-series FILE_OPEN_ procedure, the name cannot contain more than four characters.

A job fails during startup (SPECIAL-3 state) if *process-name* specifies the name of an existing process.

Note. The operating system reserves for its own use process names of the form $Z *aaa*[*a*], $Y *ddd*[*d*], and $X *ddd*[*d*], where *a* is any alphanumeric character and *d* is any numeric character. Do not use names of these forms in *process-name*.

Considerations

- An executor-program process of a job submitted without the NAME attribute is assigned a system-generated name. A low PIN process is given a four-character name, and a high PIN process is given a five-character name.
- The scheduler places the value of the NAME attribute in the *name-option* parameter of the PROCESS_CREATE_ procedure. For more information about the procedure, see the *Guardian Procedure Calls Reference Manual*.

Example

This example shows the NAME attribute specifying the name of an executor-program process:

```
> BATCHCOM $ZBAT; SUBMIT JOB X, EXECUTOR-PROGRAM DELAY,
STARTUP
"1 MINS", NAME $ABCD
Job X Jobnumber 34 submitted
> STATUS *, GMOMJOBID $ZBAT.34, DETAIL .
.
Pid: 1,50 ($ABCD) Primary .
.
GMOMJOBID: $ZBAT.34
```

OUT Job Attribute

The OUT job attribute specifies the output file to which the scheduler writes data produced by an executing job. This data can include the job's log file, executor-program output, and output from processes started by the executor program.

```
OU[T] [ file-name ]
```

file-name

is the name of an output file. The file can be a device, a process, or a disk file. If you omit *file-name*, the scheduler passes spaces as the output file name. (The effect on the job of having spaces passed as the output file name depends on the job's executor program.) Specifying a nonexistent disk file for a job whose executor program is a TACL process makes the process create an EDIT file.

BATCHCOM expands a partially qualified file name by using the defaults specified in the last VOLUME command. If there was no such command, the defaults current at the start of the session apply.

Considerations

- A job submitted without the OUT attribute adopts the DEFAULT-OUT scheduler attribute. To display the value of this attribute, use the INFO SCHEDULER command.
- The scheduler does not create a log file for a job that does not have the JOB-LOG attribute and whose OUT attribute specifies a device, a nonspooler collector process, or a disk file. The scheduler creates the file, however, if the job has the JOB-LOG attribute directing log-file events to a specified file. For more information, see JOB-LOG Job Attribute on page 7-70.
- A process that sends output to a nonterminal device or disk file opened for exclusive access by another process abends if it cannot handle file-system error 12 (file in use). To prevent a job from failing because of this error, check that child processes of a job's executor program send their output to locations different from that specified by the OUT attribute.
- A job whose OUT attribute specifies a spooler collector process creates at least two spooler jobsæone for the log file and one for executor-program output. In addition, the job creates a separate spooler job for the output of each process created by the executor program.
- For more information on NetBatch jobs and the spooler, see the manual for the spooler product you are using.

Examples

- This example shows submission of a job whose output file is a spooler collector process. The example shows the spooler jobs created by the job's processes.

Spooler jobs 100 and 101 are the log file and executor-program output file respectively. Spooler jobs 102 and 103 are the output files of the job's two FUP processes.

```
> FUP COPY FUPS
FUP FILES $NB.*
FUP FILES $DATA6.*
> BATCHCOM $ZBAT; SUBMIT JOB A, IN FUPS, OUT $S.#FUPS
Job A job number 500 submitted
> PERUSE
JOB BATCH STATE PAGES COPIES PRI HOLD LOCATION REPORT
100 91 READY 22 1 5 #FUPS A 133 2
101 91 READY 1 1 5 #FUPS NB USER
102 91 READY 14 1 5 #FUPS NB USER
103 91 READY 6 1 5 #FUPS NB USER
_JOB 100; LIST ALL
BEGIN JOB (NB.USER)A E_FUP-JOBS L_100 J_500 $X260 U_133,2
UPDATE JOB A J_500
LIST JOB A EXECUTING J_500
START U_133,2 J_500 P_TACL \MELBDEV.$X260 0,55
START U_133,2 J_500 P_FUP \MELBDEV.<000FBBA7> 0,48
STOP CC_0 J_500 \MELBDEV.<000FBBA7> 0,48
START U_133,2 J_500 P_FUP \MELBDEV.<000FBBE7> 0,48
STOP CC_0 J_500 \MELBDEV.<000FBBE7> 0,48
STOP CC_0 EXECUTOR-PROGRAM J_500 $X260
FINISH JOB A J_500 P_TACL $X260 0,55
_JOB 101; LIST ALL
TACL (T9205Dxx - DDMMMYY), Operating System Dxx .
.
FUP FILES $NB.*
FUP FILES $DATA6.*
_JOB 102; LIST ALL
$NB.A012289
FILE1 FILE2 FILE3 … .
.
_JOB 103; LIST ALL
$DATA6.BATCHUTL
FILE1 FILE2 FILE3 … .
.
```

- This example shows the failure of a FUP process created by a job's TACL executor-program process. The FUP process fails because its RUN command specifies the same disk output file as that of the executor program.

```
4> FUP COPY FUPIN
FUP /OUT FUPOUT/ FILES *
> BATCHCOM $ZBAT; SUBMIT JOB X, IN FUPIN, OUT FUPOUT
Job X job number 341 submitted
> FUP COPY FUPOUT
TACL (T9205Dxx - DDMMMYY), Operating System Dxx .
.
FUP /OUT FUPOUT/ FILES *
ABENDED: 0,76
```

- The FUP process in this example fails because it tries to open the same disk output file as that of the executor program. This failure does not occur if output from the FUP and executor-program processes goes to the same spooler location.

```
> FUP COPY FUPIN
FUP FILES *
> BATCHCOM $ZBAT; SUBMIT JOB Y, IN FUPIN, OUT FUPOUT
Job Y job number 346 submitted
112> FUP COPY FUPOUT
TACL (T9205Dxx - DDMMMYY), Operating System Dxx .
.
FUP FILES *
ABENDED: 0,70
```

- The FUP process in this example runs successfully because its disk output file is different from that of the executor program:

```
> FUP COPY FUPIN
FUP /OUT FUPOUT/ FILES *
> BATCHCOM $ZBAT; SUBMIT JOB Z, IN FUPIN, OUT OUTFILE
Job Z job number 347 submitted
> FUP COPY OUTFILE
TACL (T9205Dxx - DDMMMYY), Operating System Dxx .
.
FUP /OUT FUPOUT/ FILES *
> FUP COPY FUPOUT
$DATA7.A65052L
FILE1 FILE2 FILE3 …
$DATA7.BANK
FILE1 FILE2 FILE3 … .
.
```

PARAM Attachment-Set Attribute

The PARAM attachment-set attribute specifies the name and value of a PARAM. (A PARAM is a parameter that supplies a user-defined value to a process requesting that value at creation time.) The scheduler passes the PARAM to the executor-program process of a job using the attachment set when the job starts.

```
( PA[RAM] PARAM-name PARAM-value )
```

PARAM-name

is the name of a PARAM. The name can contain from 1 through 31 letters, numbers, hyphens, and circumflexes. The name cannot contain spaces.

Note. To avoid confusion, do not use the attribute keywords ASSIGN, DEFINE, and PARAM as PARAM names.

PARAM-value

is the PARAM value, which can contain from 1 through 255 characters. The value cannot contain spaces.

Consideration

For more information on PARAMs, see the *TACL Reference Manual* and the *TACL Programming Guide*.

Examples

- This example shows the scheduler passing PARAMs from a job's attachment set to the job's TACL executor-program process:

```
> LOGON NB.USER, psswrd
> FUP COPY INFILE
PARAM
> BATCHCOM $ZBAT; ADD ATTACHMENT-SET A, (PARAM LOC 148),
(PARAM DEPT 7326), (PARAM EMPNUM 19197)
Attachment-set (NB.USER)A added
> BATCHCOM $ZBAT; SUBMIT JOB Z, EXECUTOR-PROGRAM TACL, IN
INFILE, OUT OUTFILE, ATTACHMENT-SET A
Job Z job number 269 submitted
> FUP COPY OUTFILE
TACL (T9205Dxx - DDMMMYY), Operating System Dxx .
.
PARAM
LOC .148.
DEPT .7326.
EMPNUM .19197.
```

- This example shows various attachment-set commands adding, altering, and displaying an attachment set's PARAM attributes. The TACL environment is the source of PARAMs PERIOD-BEGIN and PERIOD-END.

```
> LOGON NB.USER, psswrd
> PARAM PERIOD-BEGIN 01SEP93
> PARAM PERIOD-END 30SEP93
> PARAM
PERIOD-BEGIN .01SEP93.
PERIOD-END .30SEP93.
> BATCHCOM $ZBAT
1} SHOW ATTACHMENT-SET PARAM
ATTACHMENT-SET ATTRIBUTES
attachments: PARAM PERIOD-BEGIN 01SEP93
PARAM PERIOD-END 30SEP93
2} SET ATTACHMENT-SET (PARAM PERIOD-ID 9)
3} ADD ATTACHMENT-SET TODAY, (PARAM DAY WEDNESDAY), (PARAM
DATE 22), (PARAM MONTH SEPTEMBER), (PARAM YEAR 1993)
Attachment-set (NB.USER)TODAY added
4} INFO ATTACHMENT-SET TODAY, PARAM *
ATTACHMENT-SET ATTRIBUTES for (NB.USER)TODAY
attachments: PARAM PERIOD-BEGIN 01SEP93
PARAM PERIOD-END 30SEP93
PARAM PERIOD-ID 9
PARAM DAY WEDNESDAY
PARAM DATE 22
PARAM MONTH SEPTEMBER
```

```
PARAM YEAR 1993
5} ALTER ATTACHMENT-SET TODAY, (PARAM DAY 22)
Attachment-set (NB.USER)TODAY altered
6} DELETE ATTACHMENT-SET TODAY, PARAM DATE
Attachment-set (NB.USER)TODAY altered
7} INFO ATTACHMENT-SET TODAY, PARAM *
attachments: PARAM PERIOD-BEGIN 01SEP93
PARAM PERIOD-END 30SEP93
PARAM PERIOD-ID 9
PARAM DAY 22
PARAM MONTH SEPTEMBER
PARAM YEAR 1993
```

PFS Job Attribute

The PFS job attribute specifies the size in bytes of a job's executor-program process file segment (PFS).

```
PF[S] number
```

number

is a number in the range 131,072 through 1,048,576 specifying the PFS size in bytes, or is 0 for the number specified in the executor program. (131,072 bytes equals 128 kilobytes [KB], or one segment; 1,048,576 bytes equals 1024 KB, or eight segments).

Considerations

- The executor-program process of a job submitted without the PFS attribute has a PFS size as specified in the executor program.
- The scheduler places the value of the PFS attribute in the *pfs-size* parameter of the PROCESS_CREATE_ procedure. For more information on the procedure, see the *Guardian Procedure Calls Reference Manual*.

Example

This example shows the ALTER JOB command setting the PFS attribute of a job:

```
3} INFO JOB 12, PFS
JOB ATTRIBUTES for PFS-TEST
jobnumber: 12
4} ALTER JOB 12, PFS 131072
Job PFS-TEST Jobnumber 12 altered
5} INFO JOB 12, PFS
JOB ATTRIBUTES for PFS-TEST
jobnumber: 12
pfs: 131072
```

PRI Job Attribute

The PRI job attribute specifies the execution priority of a job's executor-program process.

```
PR[I] number
```

number

is a number in the range 1 through 199 (1 is lowest) specifying the execution priority of a job's executor-program process.

Considerations

- A job submitted without the PRI attribute adopts the DEFAULT-PRI scheduler attribute. To display the value of this attribute, use the INFO SCHEDULER command.
- The MAX-PRI scheduler attribute overrides the PRI job attribute. For details, see MAX-PRI Scheduler Attribute on page 7-81.
- To prevent a batch job from taking precedence over an online application, make the job's execution priority lower than the application's priority.

Examples

- This example shows the submission of two jobs. Job 11 has its PRI attribute specified by the SUBMIT JOB command. Job 12 adopts the DEFAULT-PRI scheduler attribute.

```
13} INFO SCHEDULER, DEFAULT-PRI
SCHEDULER ATTRIBUTES
default-pri: 120
14} SUBMIT JOB, PRI 99
Job ZBAT-0011 Jobnumber 11 submitted
15} INFO JOB 11, PRI
JOB ATTRIBUTES for ZBAT-0011
jobnumber: 11
pri: 99
16} SUBMIT JOB
Job ZBAT-0012 Jobnumber 12 submitted
17} INFO JOB 12, PRI
JOB ATTRIBUTES for ZBAT-0012
jobnumber: 12
pri: 120
```

- This example shows the execution priorities of a job's processes and the sources of those priorities. The job's TACL executor-program process $Z357 has an execution priority of 123 (specified by the PRI attribute). FUP processes $F1 and $F2 created by the TACL process run at the priority specified by their RUN commands. FUP process $F3 runs at a priority one less than that of the TACL process. (This setting is the TACL default for processes whose RUN commands do

not include the PRI run option. For more information, see the *TACL Reference Manual*.)

```
> STATUS *, TERM
Process Pri PFR %WT Userid Program file …
$C2 0,52 149 R 000 205,70 TACL …
$ZBAT 0,89 148 P 001 255,255 NETBATCH …
$C2 B 1,51 149 001 205,70 TACL …
$ZBAT B 3,27 148 P 001 255,255 NETBATCH …
> FUP COPY INPUT
FUP /OUT $NULL, NAME $F1, NOWAIT, PRI 89/ FILES *
FUP /OUT $NULL, NAME $F2, NOWAIT, PRI 175/ FILES *
FUP /OUT $NULL, NAME $F3, NOWAIT/ FILES *
> BATCHCOM $ZBAT; INFO SCHEDULER, DEFAULT-PRI
SCHEDULER ATTRIBUTES
default-pri: 120
> BATCHCOM $ZBAT; SUBMIT JOB FILES, EXECUTOR-PROGRAM TACL,
IN INPUT, PRI 123
Job FILES job number 447 submitted
> STATUS *, USER
Process Pri PFR %WT Userid Program file …
$C2 0,52 149 R 000 205,70 TACL …
$F1 0,90 89 P 004 205,70 FUP …
$Z357 0,91 123 R 000 205,70 TACL …
$F2 0,92 175 P 004 205,70 FUP …
$F3 0,93 122 P R 000 205,70 FUP …
$C2 B 1,51 149 001 205,70 TACL …
```

PURGE-IN-FILE Job Attribute

The PURGE-IN-FILE job attribute determines whether the scheduler is to purge a job's input file when it deletes the job.

△ **Caution.** The scheduler deletes the input file regardless of whether the job has run.

```
P[URGE]-[IN]-[FILE] { OF[F] | ON }
```

`OFF`

prevents the scheduler from purging the job's input file.

`ON`

causes the scheduler to purge the job's input file when it deletes the job. The scheduler does this only if the job owner has purge access to the file.

Considerations

- The scheduler treats a job submitted without the PURGE-IN-FILE attribute like a job with the attribute PURGE-IN-FILE OFF.

- BATCHCOM displays this message if you specify the PURGE-IN-FILE ON attribute for a job whose input file denies you purge access:

```
0530-W You have no PURGE access to IN file
```

Examples

- This example shows the effect of the PURGE-IN-FILE ON attribute:

```
> FILEINFO
$DATA7.TRASH
Code … Last Modification Owner RWEP …
INFILE 101 … 3-Oct-94 11:04:30 205,70 "AAAO" …
> LOGON 205,70, psswrd
> BATCHCOM $ZBAT; SUBMIT JOB, IN $DATA7.TRASH.INFILE, PURGE
IN-FILE
ON
Job ZBAT-0024 Jobnumber 24 stopped
0512-W Other users can WRITE or PURGE the IN file; resecure
 if
required
> BATCHCOM $ZBAT; STATUS JOB 24
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
24 ZBAT-0024 205,70 300 EXECUTING DEFAULT
> BATCHCOM $ZBAT; STATUS JOB 247
2099-E JOB does not exist
> FILEINFO
No files match \MELBDEV.$DATA7.TRASH.*
```

- These scheduler log file events record the stopping of the job from the previous example and the purging of its input file:

```
03OCT94 11:18:27 FINISH JOB ZBAT-0024 T_0:0:8:880 J_24 P_TACL
03OCT94 11:18:27 DELETE JOB ZBAT-0024 J_24
03OCT94 11:18:28 PURGE IN File: \A.$DATA7.TRASH.INFILE J_24
```

- This example shows the scheduler purging the input file of a job that has not run:

```
> FILEINFO
$NB.TEMP
Code EOF Last Modification Owner RWEP …
INFILE 101 2078 16-Sep-93 11:36:38 133,2 "NNNG" …
> LOGON 133,255, psswrd
> BATCHCOM $ZBAT; SUBMIT JOB A, IN INFILE, PURGE-IN-FILE
ON, HOLD ON
Job A job number 250 submitted
> BATCHCOM $ZBAT; STATUS JOB 250
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
250 A 133,255 1:Hold DEFAULT
> BATCHCOM $ZBAT; DELETE JOB 250
Job A job number 250 deleted
> BATCHCOM $ZBAT; STATUS JOB 250
2099-E JOB does not exist
```

```
> FILEINFO
No files match \MELBDEV.$NB.TEMP.*
```

- This example shows a job's input-file security preventing the scheduler from purging the file on deletion of the job:

```
> FILEINFO
$BIG1.MANAGER
Code EOF Last Modification Owner RWEP …
ONLYMGRS 101 2078 16-Sep-93 12:08:52 205,255 "NCNO" …
> LOGON 205,100, psswrd
> BATCHCOM $ZBAT; SUBMIT JOB XYZ, IN ONLYMGRS, PURGE-IN-
FILE ON
Job XYZ job number 251 submitted
0530-W You have no PURGE access to IN file
> BATCHCOM $ZBAT; STATUS JOB 251
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
251 XYZ 205,100 203 EXECUTING DEFAULT
> BATCHCOM $ZBAT; STATUS JOB 251
2099-E JOB does not exist
130> FILEINFO
$BIG1.MANAGER
Code EOF Last Modification Owner RWEP …
ONLYMGRS 101 2078 16-Sep-93 12:08:52 205,255 "NCNO" …
```

RESTART Job Attribute

The RESTART job attribute determines whether the scheduler restarts a job that stops with completion code 7 (restart request sent to the scheduler) or terminates because of CPU failure. The attribute works in combination with the job's IFFAILS, STALL, and STOP-ON-ABEND attributes.

```
REST[ART] { OF[F] | ON }
```

`OFF`

causes the scheduler to treat the job according to:

- The job's IFFAILS, STALL, and STOP-ON-ABEND attributes.
- Whether the job is recurrent or nonrecurrent (a recurrent job has the CALENDAR or EVERY attribute).
- The cause of the failure (for example, the job's executor-program process abends) For more information, see Table 4-5, Job States on Process Failure, on page 4-28, IFFAILS Job Attribute on page 7-64, STALL Job Attribute on page 7-102, and STOP-ON-ABEND Job Attribute on page 7-105.

ON

causes the scheduler to restart the job if it stops with completion code 7 (restart request sent to the scheduler) or terminates because of CPU failure. (This action depends on the values of the job's IFFAILS, STALL, and STOP-ON-ABEND attributes.) For more information, see Figure 4-9 on page 4-29.

Considerations

- The scheduler treats a job submitted without the RESTART attribute like a job with the attribute RESTART OFF.
- The scheduler reexecutes a restarted job from the beginning of the job, not from where the job stopped.
- There is no limit to the number of times the scheduler restarts a job while the failure condition continues.
- For help when setting the IFFAILS, RESTART, STALL, and STOP-ON-ABEND attributes, see the decision chart in Section 4, Job Planning, Submission, and Management.

Example

This example shows the effect of the RESTART attribute on two recurrent jobs whose CPU fails. The jobs have the same IFFAILS, STALL, and STOP-ON-ABEND attributes, but different RESTART attributes. Job X has the RESTART ON attribute, so the scheduler puts the job in the READY state after failure. Job Y has the RESTART OFF attribute, so the scheduler puts the job in the TIME state after failure.

```
> TIME
November 9, 1993 15:44:18
> FUP COPY DELAY05
DELAY 5 MINS
> BATCHCOM $ZBAT; SUBMIT JOB X, IN DELAY05, EVERY 1 DAYS,
RESTART ON, IFFAILS ON, STALL OFF, STOP-ON-ABEND ON
Job X job number 438 submitted
> BATCHCOM $ZBAT; SUBMIT JOB Y, LIKE X, RESTART OFF
Job Y job number 439 submitted
> BATCHCOM $ZBAT; STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
438 X 255,255 85 EXECUTING DEFAULT
439 Y 255,255 87 EXECUTING DEFAULT
> STATUS *, GMOMJOBID $ZBAT.438
Process … Program file Hometerm
$Z238 … $SYSTEM.SYS00.TACL $ZBAT
… $SYSTEM.SYS00.DELAY $ZBAT
> STATUS *, GMOMJOBID $ZBAT.439
Process … Program file Hometerm
$Z239 … $SYSTEM.SYS00.TACL $ZBAT
… $SYSTEM.SYS00.DELAY $ZBAT
```

```
> DIVER /CPU 3/
PROCESSOR FAILURE: 3
> BATCHCOM $ZBAT; STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
438 X 255,255 85 READY DEFAULT
439 Y 255,255 87 10NOV93 DEFAULT
```

RUND Job Attribute

The RUND job attribute specifies whether a job's executor program enters the Guardian debug facility Debug or the Inspect interactive symbolic debugger when the program runs.

```
RUND { OF[F] | ON }
```

OFF

specifies normal program execution.

ON

makes the executor program enter Debug or Inspect at the first executable instruction of the program's MAIN procedure. For information about Debug, see the *Debug Manual*. For information about Inspect, see the *Inspect Manual*. A job with the RUND ON attribute must also have the TERM attribute specifying a home terminal for Debug or Inspect output. Without the TERM attribute, the job fails during startup and goes into the SPECIAL-4 state.

Considerations

- The scheduler treats a job submitted without the RUND attribute like a job with the attribute RUND OFF.
- The scheduler places the value of the RUND attribute in the *debug-options* parameter of the PROCESS_CREATE_ procedure. For more information about the procedure, see the *Guardian Procedure Calls Reference Manual*.

Example

This example shows the effect of the RUND attribute:

```
> BATCHCOM $ZBAT; SUBMIT RUND-JOB, EXECUTOR-PROGRAM BATCHCOM,
TERM $ZTN0.#PTY0014, RUND ON
Job RUND-JOB Jobnumber 4 submitted
> BATCHCOM $ZBAT; STATUS JOB 4
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
```

```
4 RUND-JOB 205,70 EXECUTING DEFAULT
> PAUSE
INSPECT - Symbolic Debugger - T9673D30 - (31OCT94) System …
(C)1983 Tandem (C)2004 Hewlett Packard Development Company,
L.P.
INSPECT
247,01,00039 $Y225 #BATCHCOM.#11595(SBATCOM) + %1I
-$Y225-
```

SAVEABEND Job Attribute

The SAVEABEND job attribute specifies whether a job's executor-program process is to create a save file if the process traps or abends.

```
SA[VEABEND] { OF[F] | ON }
```

`OFF`

makes the executor-program process function according to the saveabend flags compiled or bound into the program.

`ON`

makes the executor-program process create a save file in the program-file subvolume if the process traps or abends.

Considerations

- The scheduler treats a job submitted without the SAVEABEND attribute like a job with the attribute SAVEABEND OFF.
- The scheduler places the value of the SAVEABEND attribute in the *`debug-options`* parameter of the PROCESS_CREATE_ procedure. For more information about the procedure, see the *Guardian Procedure Calls Reference Manual*.

Example

This example shows scheduler log-file events relating to a job that abends. The job has the attribute SAVEABEND ON, so the executor-program process creates a save file in the program's subvolume.

```
> RUN BATCHCOM $ZBAT; SUBMIT JOB ABEND-TEST, EXECUTOR-PROGRAM
$DATA7.OBJFILES.SAVEOBJ, SAVEABEND ON
Job ABEND-TEST Jobnumber 11 submitted
> FUP COPY $DATA7.ZBAT.LOGAAD,,SHARE,FOLD
NETBATCH SCHEDULER/MONITOR - T9190D30 - (31OCT94^01JUN94) .
.
15SEP94 14:45:37 BEGIN JOB (SUPER.SUPER)ABEND-TEST:1 E_E1
L_3530 J_11 P_SAVEOBJ
\MELBDEV.$Y290:16288557 U_255,255 .
.
15SEP94 14:45:39 START ERROR (alter PRI) 11 J_11 P_SAVEOBJ
```

```
\MELBDEV.$Y290:16288557
15SEP94 14:45:42 OPEN U_255,255 P_DMON
\MELBDEV.$DM01:9611309
15SEP94 14:45:42 MSG Save file
\MELBDEV.$DATA7.OBJFILES.ZZSA2285
created for process 1,51 ($Y290)
P_DMON \MELBDEV.$DM01:9611309
15SEP94 14:45:42 CLOSE \MELBDEV.$DM01:9611309 U_255,255
15SEP94 14:45:42 ABEND CC_5 EXECUTOR-PROGRAM J_11
\MELBDEV.$Y290:16288557 .
.
```

SECURITY Attachment-Set Attribute

The SECURITY attachment-set attribute controls user access to an attachment set.

```
SEC[URITY] " security "
```

security

is a four-character string specifying Guardian security for read, write, execute, and purge (RWEP) access. For each type of access, specify one of these codes to indicate the required security level:

Code	Description
A	Anyone—Lets any local user access the attachment set
N	Network—Lets any local or remote user access the attachment set
G	Group—Lets any local user in the owner's group access the attachment set
C	Community—Lets any local or remote user in the owner's group access the attachment set
O	Owner—Lets only the local owner access the attachment set
U	User—Lets the owner access the attachment set from a local or remote node
-	Hyphen—Lets only the local super ID (255,255) access the attachment set

Considerations

- An attachment set added without the SECURITY attribute adopts by default the attribute SECURITY "UUUU."
- You cannot use the Safeguard product to secure an attachment set.
- The operations you can perform on an attachment set depend on the command used and the set's SECURITY attribute. Table 7-3 on page 7-100 lists ATTACHMENT-SET commands (operations) and the RWEP access needed to perform those operations.

Table 7-3. Attachment-Set Access

Command	Required RWEP Access
ADD	NA
ALTER	Write access
ASSSUME	NA
DELETE	Purge access—to delete an entire attachment set Write access—to delete specified ASSIGNs, DEFINEs, and PARAMs from an attachment set
INFO	Read access—to display an entire attachment set Not applicable—to display *attachment-set-ID*, and SECURITY and TEMPORARY attributes only
RESET	NA
SET	NA
STATUS	NA

Examples

- The attachment set added in this example adopts the default SECURITY attribute:

```
3} CHANGEUSER NB.USER psswrd
3} ADD ATTACHMENT-SET MYSET, (DEFINE =_DEFAULTS, VOLUME
$A.NB), TEMPORARY ON
Attachment-set (NB.USER)MYSET added
4} INFO ATTACHMENT-SET MYSET, DETAIL
ATTACHMENT-SET ATTRIBUTES for (NB.USER)MYSET
security: "UUUU"
temporary: On
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$A.NB
```

- This example shows an attachment set's SECURITY attribute preventing a user from displaying the set's ASSIGN, DEFINE, and PARAM attributes:

```
10} CHANGEUSER SUPER.FPP psswrd
10} ADD ATTACHMENT-SET MNGRS, (DEFINE =_DEFAULTS, VOLUME
$ADMIN.STAFF), (ASSIGN SALARY-FILE, STAFF.SALARIES),
(PARAM SALARY 100000), SECURITY "GOGO"
Attachment-set (SUPER.FPP)MNGRS added
11} INFO ATTACHMENT-SET MNGRS, DETAIL
ATTACHMENT-SET ATTRIBUTES for (SUPER.FPP)MNGRS
security: "GOGO"
temporary: Off
attachments: DEFINE =_DEFAULTS, CLASS DEFAULTS, VOLUME
$ADMIN.STAFF
ASSIGN SALARY-FILE, STAFF.SALARIES
PARAM SALARY 100000
12} CHANGEUSER NB.USER psswrd
12} INFO ATTACHMENT-SET (SUPER.FPP)MNGRS, DETAIL
ATTACHMENT-SET ATTRIBUTES for (SUPER.FPP)MNGRS
```

```
security: "GOGO"
temporary: Off
0535-I Your user code does not give you access to
(SUPER.FPP)MNGRS data
```

SELPRI Job Attribute

The SELPRI job attribute specifies the selection priority of a job in its class.

```
SEL[PRI] number
```

number

is a number in the range 0 through 7 (0 is lowest) specifying the selection priority.

Considerations

- A job submitted without the SELPRI attribute adopts the DEFAULT-SELPRI scheduler attribute. To display the value of this attribute, use the INFO SCHEDULER command.
- Selection of jobs with the same CLASS attribute, SELPRI attribute, and run time is by submission time on a first-in, first-out basis. For more information about job selection, see Section 1, NetBatch Introduction.
- The RUNNEXT JOB and RUNNOW JOB commands override a job's SELPRI attribute. For more information, see RUNNEXT JOB Command on page 6-125 and RUNNOW JOB Command on page 6-128.

Example

This example shows the effect of the SELPRI attribute on job selection. Job 813, having the highest selection priority, runs first. Jobs 812 and 815 run second and third, respectively. (Both jobs have the same SELPRI attribute, but job 812 has an earlier submission time, so it runs before job 815.) Job 814 has the lowest selection priority (inherited from the scheduler), so it runs last.

```
4} INFO CLASS DEFAULT
CLASS ATTRIBUTE for DEFAULT
initiation: Off
5} INFO SCHEDULER, DEFAULT-CLASS, DEFAULT-SELPRI
SCHEDULER ATTRIBUTES
default-selpri: 3
default-class: DEFAULT
6} SET JOB EXECUTOR-PROGRAM DELAY, STARTUP "15 SECS"
7} SUBMIT JOB A-SELPRI-6, SELPRI 6; &
}} SUBMIT JOB B-SELPRI-7, SELPRI 7; &
}} SUBMIT JOB C-SELPRI-DEFAULT; &
}} SUBMIT JOB D-SELPRI-6, SELPRI 6
Job A-SELPRI-6 job number 812 submitted
Job B-SELPRI-7 job number 813 submitted
Job C-SELPRI-DEFAULT job number 814 submitted
```

```
Job D-SELPRI-6 job number 815 submitted
8} INFO JOB 814, SELPRI
JOB ATTRIBUTES for C-SELPRI-DEFAULT
jobnumber: 814
selpri: 3
9} ALTER CLASS DEFAULT, INITIATION ON
Class DEFAULT altered
10} STATUS JOB *, SELPRI
JOB JOBNAME USERID SEL STATE CLASSNAME
---- ------------------ ------- ---- ---------- ---------
812 A-SELPRI-6 255,205 6 READY DEFAULT
813 B-SELPRI-7 255,205 7 EXECUTING DEFAULT
814 C-SELPRI-DEFAULT 255,205 3 READY DEFAULT
815 D-SELPRI-6 255,205 6 READY DEFAULT
11} STATUS JOB *, SELPRI
JOB JOBNAME USERID SEL STATE CLASSNAME
---- ------------------ ------- ---- ---------- ---------
812 A-SELPRI-6 255,205 6 EXECUTING DEFAULT
814 C-SELPRI-DEFAULT 255,205 3 READY DEFAULT
815 D-SELPRI-6 255,205 6 READY DEFAULT
12} STATUS JOB *, SELPRI
JOB JOBNAME USERID SEL STATE CLASSNAME
---- ------------------ ------- ---- ---------- ---------
814 C-SELPRI-DEFAULT 255,205 3 READY DEFAULT
815 D-SELPRI-6 255,205 6 EXECUTING DEFAULT
13} STATUS JOB *, SELPRI
JOB JOBNAME USERID SEL STATE CLASSNAME
---- ------------------ ------- ---- ---------- ---------
814 C-SELPRI-DEFAULT 255,205 3 EXECUTING DEFAULT
```

STALL Job Attribute

The STALL job attribute determines whether the scheduler puts in the SPECIAL-9 state a failed job it would otherwise reschedule or delete. The attribute works in combination with the job's IFFAILS, RESTART, and STOP-ON-ABEND attributes.

```
STAL[L] { OF[F] | ON }
```

`OFF`

causes the scheduler to treat the job according to:

- The job's IFFAILS, RESTART, and STOP-ON-ABEND attributes
- Whether the job is recurrent or nonrecurrent (a recurrent job has the CALENDAR or EVERY attribute)
- The cause of the failure (for example, the job's executor-program process abends)

For more information, see IFFAILS Job Attribute on page 7-64, RESTART Job Attribute on page 7-95, and STOP-ON-ABEND Job Attribute on page 7-105.

ON

causes the scheduler to put the job in the SPECIAL-9 state if the job fails and either of these conditions exists. (This action depends on the values of the job's IFFAILS, RESTART, and STOP-ON-ABEND attributes.)

- The job is recurrent, and the scheduler would otherwise reschedule it.
- The job is nonrecurrent, and the scheduler would otherwise delete it.

Considerations

- A job submitted without the STALL attribute adopts the DEFAULT-STALL scheduler attribute. To display the value of this attribute, use the INFO SCHEDULER command.
- For help when setting the IFFAILS, RESTART, STALL, and STOP-ON-ABEND attributes, see the decision chart in Section 4, Job Planning, Submission, and Management.

Example

This example shows the effect of the STALL attribute on two recurrent jobs whose executor programs abend. The jobs have the same IFFAILS, RESTART, and STOP-ON-ABEND attributes, but different STALL attributes. Job X has the STALL OFF attribute, so the scheduler puts the job in the TIME state after failure. Job Y has the STALL ON attribute, which causes the scheduler to put the job in the SPECIAL-9 state after failure.

```
> TIME
October 13, 1993 12:37:55
> FUP COPY DELAY05
DELAY 5 MINS
> BATCHCOM $ZBAT; SUBMIT JOB X, IN DELAY05, EVERY 1 DAYS,
IFFAILS ON, RESTART OFF, STOP-ON-ABEND ON, STALL OFF
Job X job number 380 submitted
> BATCHCOM $ZBAT; SUBMIT JOB Y, LIKE X, STALL ON
Job Y job number 381 submitted
> BATCHCOM $ZBAT; STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
380 X 205,70 30 EXECUTING DEFAULT
381 Y 205,70 32 EXECUTING DEFAULT
> STATUS *, GMOMJOBID $ZBAT.380
Process … Program file Hometerm
$Z075 … $SYSTEM.SYS00.TACL $ZBAT
… $SYSTEM.SYS00.DELAY $ZBAT
> STATUS *, GMOMJOBID $ZBAT.381
Process … Program file Hometerm
… $SYSTEM.SYS00.DELAY $ZBAT
$Z076 … $SYSTEM.SYS00.TACL $ZBAT
> STOP $Z075
```

```
> STOP $Z076
> BATCHCOM $ZBAT; STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
380 X 205,70 30 14OCT93 DEFAULT
381 Y 205,70 32 9:STALL DEFAULT
```

STARTUP Job Attribute

The STARTUP job attribute specifies one or more program parameters the scheduler sends a job's executor-program process in the startup message.

```
STARTU[P] " param-set "
```

param-set

is a string of 1 through 961 characters.

Consideration

S-M (an abbreviation of STARTUP-MESSAGE) is an alias of the attribute keyword STARTUP.

Examples

- This example shows the submission, execution, and result of a job with the STARTUP attribute. The attribute specifies parameters that cause the job's BATCHCOM executor-program process to open scheduler $SCHD and execute a STATUS EXECUTOR command.

```
> BATCHCOM; SUBMIT JOB S, OUT OUTFILE, EXECUTOR-PROGRAM
BATCHCOM, STARTUP "$SCHD; STATUS EXECUTOR *"
Job S job number 354 submitted
> FUP COPY OUTFILE
EXECUTOR STATUS
EXECUTOR CPU STATE JOB CLASS
----------------------- --- ------ ---- -------
E1 0 ACTIVE 354 DEFAULT
E2 1 ON
E3 2 ON
E4 3 ON
```

- This example shows the STARTUP attribute specifying parameters for a job's NBEXEC executor program:

```
1} SUBMIT JOB A, EXECUTOR-PROGRAM NBEXEC, OUT
$DATA7.NB.PURGEME, STARTUP "LABEL XYZ, PURGE, LIMIT 3:00"
Job A job number 501 submitted
2} INFO JOB A, STARTUP
JOB ATTRIBUTES for A
```

```
jobnumber: 501
startup: LABEL XYZ, PURGE, LIMIT 3:00
```

STOP-ON-ABEND Job Attribute

The STOP-ON-ABEND job attribute determines whether the scheduler stops a job and all its processes if any process of the job terminates because of CPU failure; abends with any completion code; or stops with completion code -3, -2, -1, 2, 3, 4, 5, or 6. The attribute works in combination with the job's IFFAILS, RESTART, and STALL attributes.

```
S[TOP]-[ON]-[ABEND] { OF[F] | ON }
```

`OFF`

causes the scheduler to treat the job according to:

- The job's IFFAILS, RESTART, and STALL attributes
- Whether the job is recurrent or nonrecurrent (a recurrent job has the CALENDAR or EVERY attribute)
- The cause of the failure (for example, the executor-program process abends) For more information, see Table 4-5 on page 4-28, IFFAILS Job Attribute on page 7-64, RESTART Job Attribute on page 7-95, and STALL Job Attribute on page 7-102.

`ON`

causes the scheduler to stop the job when any process of the job terminates, abends, or stops. The stopped job's state depends on the values of the job's IFFAILS, RESTART, and STALL attributes. For more information, see Figure 4-9 on page 4-29.

Note. An executor-program process that is not a TACL process might start new processes before the scheduler receives the terminate, abend, or stop message. To prevent these new processes from starting, make the executor program test the completion code of each process before starting the next process.

Considerations

- A job submitted without the STOP-ON-ABEND attribute adopts the DEFAULT-STOP-ON-ABEND scheduler attribute. To display the value of this attribute, use the INFO SCHEDULER command.
- A failed process of a job with the STOP-ON-ABEND ON attribute does not stop the job if dissociated from it by the TACL RUN command option JOBID or by the JOBID-ZERO job attribute.
- Both processes of a process pair must cease to exist before the STOP-ON-ABEND ON attribute takes effect.

- Do not use STOP-ON-ABEND for NBEXEC jobs. NBEXEC always terminates on an untrapped error condition.
- For help when setting the IFFAILS, RESTART, STALL, and STOP-ON-ABEND attributes, see Figure 4-9 on page 4-29.

Example

This example shows the effect of the STOP-ON-ABEND attribute on two recurrent jobs whose child processes stop because of CPU failure. The jobs have the same IFFAILS, RESTART, and STALL attributes, but different STOP-ON-ABEND attributes. Job X has the STOP-ON-ABEND ON attribute, so the scheduler puts the job in the SPECIAL-6 state after failure. Job Y has the STOP-ON-ABEND OFF attribute, so the scheduler puts the job in the TIME state after failure.

```
> TIME
November 10, 1993 14:43:15
> FUP COPY DELAY05
DELAY /CPU 3/ 5 MINS
> BATCHCOM $ZBAT; SUBMIT JOB X, IN DELAY05, EVERY 1 DAYS,
IFFAILS OFF, RESTART OFF, STALL OFF, STOP-ON-ABEND ON
Job X job number 445 submitted
> BATCHCOM $ZBAT; SUBMIT JOB Y, LIKE X, STOP-ON-ABEND OFF
Job Y job number 446 submitted
> BATCHCOM $ZBAT; STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
445 X 255,255 1 EXECUTING DEFAULT
446 Y 255,255 3 EXECUTING DEFAULT
> STATUS *, GMOMJOBID $ZBAT.445
Process … Program file Hometerm
$Z333 … $SYSTEM.SYS00.TACL $ZBAT
… $SYSTEM.SYS00.DELAY $ZBAT
> STATUS *, GMOMJOBID $ZBAT.446
Process … Program file Hometerm
$Z334 … $SYSTEM.SYS00.TACL $ZBAT
… $SYSTEM.SYS00.DELAY $ZBAT
> DIVER /CPU 3/
PROCESSOR FAILURE: 3
> BATCHCOM $ZBAT; STATUS JOB *
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
445 X 255,255 1 6:Fail RsOffDEFAULT
446 Y 255,255 3 11NOV93 DEFAULT
```

SUBMIT-ALLOWED Scheduler Attribute

The SUBMIT-ALLOWED scheduler attribute allows or disallows job submission by permitting or preventing use of the SUBMIT JOB command.

```
S[UBMIT]-A[LLOWED] { OF[F] | ON }
```

`OFF`

disallows job submission by preventing use of the SUBMIT JOB command. BATCHCOM displays this message when a scheduler with the SUBMIT-ALLOWED OFF attribute rejects a submitted job:

```
2128-E SUBMIT-ALLOWED is OFF; the scheduler is not
accepting job submissions
```

`ON`

allows job submission.

Considerations

- A scheduler adopts the attribute SUBMIT-ALLOWED ON by default when cold started.
- The SUBMIT-ALLOWED OFF attribute affects the SUBMIT JOB command only. It does not affect other scheduler operations.
- ADD is an alias of SUBMIT. Therefore ADD-ALLOWED is an acceptable variant of SUBMIT-ALLOWED.

Examples

- This example shows the effect of the SUBMIT-ALLOWED attribute:

```
14} INFO SCHEDULER, SUBMIT-ALLOWED
SCHEDULER ATTRIBUTES
submit-allowed: Off
15} SUBMIT JOB X
2128-E SUBMIT-ALLOWED is OFF; the scheduler is not accepting
job submissions
16} ALTER SCHEDULER, SUBMIT-ALLOWED ON
Scheduler altered
17} INFO SCHEDULER, SUBMIT-ALLOWED
SCHEDULER ATTRIBUTES
submit-allowed: On
18} SUBMIT JOB X
Job X job number 213 submitted
```

- This example shows a scheduler running a job, despite having the SUBMIT-ALLOWED OFF attribute. The example illustrates the consideration that states the

attribute affects only the SUBMIT JOB command and not other scheduler operations.

```
8} SUBMIT JOB Y, HOLD ON
Job Y job number 253 submitted
9} ALTER SCHEDULER, SUBMIT-ALLOWED OFF
Scheduler altered
10} STATUS JOB Y
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
253 Y 255,255 1:Hold DEFAULT
11} ALTER JOB Y, HOLD OFF
Job Y job number 253 altered
12} STATUS JOB Y
JOB STATUS
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
253 Y 255,255 30 EXECUTING DEFAULT
13} SUBMIT JOB Z
2128-E SUBMIT-ALLOWED is OFF; the scheduler is not accepting
job submissions
```

SWAP Job Attribute

The SWAP job attribute specifies the name of the swap file for the user data stack segment of a job's executor-program process. The file provides space for memory swaps of the user data stack during process execution.

```
SWA[P] { $volume-name | file-name }
```

volume-name

is the name of the volume where the executor-program process is to create a temporary swap file. When the process terminates, the operating system automatically purges the file.

file-name

is the name of a swap file. The file must be on the same node as the executor-program process and can be used by only one process at a time. The file is not purged when the executor-program process terminates.

BATCHCOM expands a partially qualified file name by using the defaults specified in the last VOLUME command. If there was no such command, the defaults current at the start of the session apply.

Considerations

- A job submitted without the SWAP attribute creates the swap file on the volume specified by the SWAP attribute of DEFINE =_DEFAULTS. If the DEFINE does not have the SWAP attribute, the operating system selects a volume for the file.

- The scheduler places the value of the SWAP attribute in the *swap-file* parameter of the PROCESS_CREATE_ procedure. For more information on the procedure, see the *Guardian Procedure Calls Reference Manual*.

Example

This example shows the SWAP attribute specifying the swap file for a job's TACL executor-program process:

```
> BATCHCOM $ZBAT; SUBMIT JOB SWAP-JOB, EXECUTOR-PROGRAM TACL,
IN \MELRISK.$QA.FILES.INFILE, SWAP $TRASH.ZBAT.SWAP
Job SWAP-JOB Jobnumber 6 submitted
> STATUS *, GMOMJOBID $ZBAT.6, DETAIL .
.
Swap File Name: $TRASH.ZBAT.SWAP .
.
GMOMJOBID: $ZBAT.6
```

TAPEDRIVES Job Attribute

The TAPEDRIVES job attribute specifies the number of tape drives required by a job.

```
TA[PEDRIVES] number
```

number

is a number in the range 0 through 99 specifying the number of tape drives required.

Considerations

- The scheduler treats a job submitted without the TAPEDRIVES attribute like a job with the attribute TAPEDRIVES 0.
- When you submit a job with the TAPEDRIVES attribute, the scheduler checks the number of drives required against an internal counter. The scheduler's treatment of the job (assuming no other factors are delaying job execution) then depends on whether the number is:
  - Less than or equal to the counter. In this case, the scheduler runs the job immediately.
  - Greater than the counter, but no greater than the TAPEDRIVES scheduler attribute. In this case, the scheduler runs the job when the required drives are available. The state of jobs waiting for tape drives is TAPE.
  - Greater than the TAPEDRIVES scheduler attribute. In this case, the scheduler never runs the job. To avoid this situation, do not specify more drives for a job than are available to the scheduler. If the situation occurs, to resolve it do one of:

- Increase the number of drives specified by the TAPEDRIVES scheduler attribute
- Reduce the number of drives required by the job

- For more information on the scheduler's internal tape drives counter, see TAPEDRIVES Scheduler Attribute on page 7-110.

Example

This example shows the submission of two jobs, each of which requires tape drives. The scheduler puts job 396 into the TAPE state because insufficient drives are available. The job does not run until job 395 finishes.

```
40} INFO SCHEDULER, TAPEDRIVES
SCHEDULER ATTRIBUTES
tape-drives: 2
41} SUBMIT JOB BACKUP-TO-TAPE, IN TAPEBKUP, TAPEDRIVES 1
Job BACKUP-TO-TAPE job number 395 submitted
42} SUBMIT JOB STATEMENTS, IN STATMNTS, TAPEDRIVES 2
Job STATEMENTS job number 396 submitted
43} STATUS JOB *
JOB JOBNAME USERID LOG STATE
CLASSNAME
---- ---------------------- ------- ---- ----------- -----
395 BACKUP-TO-TAPE 255,205 159 EXECUTING DEFAULT
396 STATEMENTS 255,205 TAPE DEFAULT
```

TAPEDRIVES Scheduler Attribute

The TAPEDRIVES scheduler attribute specifies the number of tape drives available for use by jobs.

Note. The scheduler cannot determine the actual number of tape drives configured for its node. Consequently, you must check tape drive availability before specifying the TAPEDRIVES scheduler attribute. (Use the TAPECOM STATUS command or the PUP LISTDEV TYPE 4 command, or their equivalent MEDIACOM and SCF commands, respectively.)

```
TA[PEDRIVES] number
```

number

is a number in the range 0 through 99 specifying the number of tape drives available.

Considerations

- A scheduler adopts the attribute TAPEDRIVES 2 by default when cold started.

- The scheduler calculates the number of tape drives available for use by using its TAPEDRIVES attribute to initialize an internal counter. When you submit a job, the scheduler compares the number of drives required by the job with the counter. The scheduler's treatment of the job (assuming no other factors are delaying job execution) depends on whether the number is:
  - Less than or equal to the counter. In this case, the scheduler runs the job immediately and subtracts the number from the counter. When the job finishes, the scheduler adds the number to the counter.
  - Greater than the counter, but no greater than the TAPEDRIVES scheduler attribute. In this case, the scheduler runs the job when the required drives are available. The state of jobs waiting for tape drives is TAPE.
  - Greater than the TAPEDRIVES scheduler attribute. In this case, the scheduler never runs the job. To avoid this situation, do not specify more drives for a job than are available to the scheduler. If the situation occurs, to resolve it do one of:
    - Increase the number of drives specified by the TAPEDRIVES scheduler attribute
    - Reduce the number of drives required by the job
- To display the number of tape drives configured for a scheduler and the number in use, use the STATUS SCHEDULER command.

Example

This example shows the configuration of a scheduler's TAPEDRIVES attribute. The example also shows the submission of a job whose tape drives requirement reduces the available drives from three to one.

```
> BATCHCOM $ZBAT; INFO SCHEDULER, TAPEDRIVES
SCHEDULER ATTRIBUTES
tape-drives: 2
> TAPECOM STATUS
DEVICE TYPE VID STATE OPEN I/O EXC PROCESS-ID …
$TAPE1 NONE FREE
$TAPE2 NONE FREE
$TAPE3 NONE FREE
24> BATCHCOM $ZBAT; ALTER SCHEDULER, TAPEDRIVES 3
Scheduler altered
25> BATCHCOM $ZBAT; STATUS SCHEDULER
SCHEDULER STATUS
Process : \MELBDEV.$ZBAT Primary : 1,37 Backup : 3,27
Database: $DATA7.ZBAT
Logfile : $DATA7.ZBAT.LOGAAW
Time : 02OCT93 12:43:56
EXECUTORS JOBS CLASSES …
---------- -------------- -------- …
OFF 0 READY 0 OFF 0 …
ON 1 EXECUTING 0 ON 1 …
```

```
ACTIVE 0 SPECIAL 0
STOP 0 TIME 0
DOWN 0 EVENT 0 TAPE DRIVES …
DELETE 0 SUSPENDED 0 ------------ …
RUN NEXT 0 CONFIGURED 3 …
RUN NOW 0 IN USE 0
TAPE 0
26> BATCHCOM $ZBAT; SUBMIT JOB BACKUP, TAPEDRIVES 2
Job BACKUP job number 1 submitted
27> BATCHCOM $ZBAT; STATUS SCHEDULER
SCHEDULER STATUS
Process : \MELBDEV.$ZBAT Primary : 1,37 Backup : 3,27
Database: $DATA7.ZBAT
Logfile : $DATA7.ZBAT.LOGAAW
Time : 02OCT93 12:45:34
EXECUTORS JOBS CLASSES …
---------- -------------- -------- …
OFF 0 READY 0 OFF 0 …
ON 0 EXECUTING 1 ON 1 …
ACTIVE 1 SPECIAL 0
STOP 0 TIME 0
DOWN 0 EVENT 0 TAPE DRIVES …
DELETE 0 SUSPENDED 0 ------------ …
RUN NEXT 0 CONFIGURED 3 …
RUN NOW 0 IN USE 2
TAPE 0
```

TEMPORARY Attachment-Set Attribute

The TEMPORARY attachment-set attribute determines whether the scheduler automatically deletes an attachment set not assigned to a job.

```
TEM[PORARY] { OF[F] | ON }
```

OFF

prevents the scheduler from automatically deleting the attachment set.

ON

causes the scheduler to delete automatically the attachment set in any of these circumstances. Deletion occurs only when the set is not in use by a job.

- The set's TEMPORARY attribute changes from OFF to ON during a BATCHCOM session.
- BATCHCOM closes the set's scheduler during a session (for example, in response to an OPEN command).
- The set's scheduler stops (the scheduler deletes the set when restarted).
- The BATCHCOM session during which the set was created ends.

Consideration

The scheduler assigns a default TEMPORARY attribute to an attachment set added by a command that does not specify that attribute. The value of the default attribute depends on whether the set has a user-specified or scheduler-generated (through #CURRENT) identifier. Possible values are:

- TEMPORARY OFF—for sets added with user-specified identifiers (that is, named attachment sets). For example:

```
17} ADD ATTACHMENT-SET (FPP.QA)ADP
Attachment-set (FPP.QA)ADP added
18} INFO ATTACHMENT-SET (FPP.QA)ADP, TEMPORARY
ATTACHMENT-SET ATTRIBUTES for (FPP.QA)ADP
temporary: Off
```

- TEMPORARY ON—for sets added with scheduler-generated identifiers (that is, numbered attachment sets). For example:

```
22} CHANGEUSER NB.USER psswrd
22} SUBMIT JOB X, ATTACHMENT-SET #CURRENT
Attachment-set (NB.USER)12 added
Job X job number 16 submitted
23} INFO ATTACHMENT-SET 12, TEMPORARY
ATTACHMENT-SET ATTRIBUTES for (NB.USER)12
temporary: On
```

Examples

- This example shows the automatic deletion of an attachment set at the end of a BATCHCOM session:

```
9} ADD ATTACHMENT-SET #CURRENT
Attachment-set (NB.USER)34 added
10} INFO ATTACHMENT-SET 34, TEMPORARY
ATTACHMENT-SET ATTRIBUTES for (NB.USER)34
temporary: On
11} SUBMIT JOB A, IN INFILE, HOLDAFTER ON, ATTACHMENT-SET 34
Job A job number 387 submitted
12} STATUS JOB 387
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
387 A 133,2 132 EXECUTING DEFAULT
13} INFO JOB 387, ATTACHMENT-SET
JOB ATTRIBUTES for A
jobnumber: 387
attachment-set: (NB.USER)34
14} STATUS ATTACHMENT-SET 34
ATTACHMENT SET IN USE BY JOBS
-------------------------------- --------------
(NB.USER)34 (NB.USER)A
15} STATUS JOB 387
JOB JOBNAME USERID LOG STATE CLASSNAME
---- --------------- ------- ---- ----------- ---------
```

```
387 A 133,2 132 1:Hold DEFAULT
16} STATUS ATTACHMENT-SET 34
ATTACHMENT SET IN USE BY JOBS
-------------------------------- -------------
(NB.USER)34 (NB.USER)A
17} DELETE JOB 387
Job A job number 387 deleted
18} STATUS ATTACHMENT-SET 34
ATTACHMENT SET IN USE BY JOBS
-------------------------------- -------------
(NB.USER)34 None
19} EXIT
> BATCHCOM $ZBAT; STATUS ATTACHMENT-SET 34
2172-E (NB.USER)34 does not exist
```

- This example shows the automatic deletion of an attachment set whose TEMPORARY attribute changes from OFF to ON during a session:

```
27} INFO ATTACHMENT-SET (FPP.QA)DATE, TEMPORARY
ATTACHMENT-SET ATTRIBUTES for (FPP.QA)DATE
temporary: Off
28} STATUS ATTACHMENT-SET (FPP.QA)DATE
ATTACHMENT SET IN USE BY JOBS
-------------------------------- -------------
(FPP.QA)DATE None
29} ALTER ATTACHMENT-SET (FPP.QA)DATE, TEMPORARY ON
Attachment-set (FPP.QA)DATE altered
0534-W Deleted temporary attachment set (FPP.QA)DATE
No jobs were referencing this attachment set.
```

TERM Job Attribute

The TERM job attribute specifies the home terminal of a job's executor-program process.

```
TERM $process-name
```

process-name

specifies the name of the home-terminal process.

BATCHCOM expands a partially qualified process name by using the defaults specified in the last VOLUME command. If there was no such command, the defaults current at the start of the session apply.

If you specify TERM without *process-name* in an ALTER JOB command, the scheduler removes the TERM attribute from the job. Specifying TERM without *process-name* in a SET JOB command makes BATCHCOM remove the TERM attribute from the working-attributes set.

Considerations

- A job submitted without the TERM attribute uses the scheduler as its home terminal.
- A job with the RUND ON attribute must also have the TERM attribute specifying a home terminal for Debug or Inspect output. Without the TERM attribute, the job fails during startup and goes into the SPECIAL-4 state.
- Processes of a job with the JOBID-ZERO ON attribute cannot access the scheduler as a home terminal. For this reason, use the TERM attribute to specify a home terminal for the job.
- The scheduler places the value of the TERM attribute in the *hometerm* parameter of the PROCESS_CREATE_ procedure. For more information about the procedure, see the *Guardian Procedure Calls Reference Manual*.

Example

This example shows the submission of two jobs—one with the TERM attribute, the other without. The job with the attribute sends home-terminal output to the specified terminal. The job without the attribute sends home-terminal output to the scheduler.

```
> BATCHCOM $ZBAT; SUBMIT JOB TERM, EXECUTOR-PROGRAM DELAY,
STARTUP "1 MINS", TERM $ZTN0.#PTY6
Job TERM Jobnumber 5 submitted
> STATUS *, GMOMJOBID $ZBAT.5, DETAIL .
.
Myterm: $ZTN0.#PTY6 .
.
GMOMJOBID: $ZBAT.5
> BATCHCOM $ZBAT; SUBMIT JOB NO-TERM, EXECUTOR-PROGRAM DELAY,
STARTUP "1 MINS"
Job NO-TERM Jobnumber 6 submitted
> STATUS *, GMOMJOBID $ZBAT.6, DETAIL .
.
Myterm: $ZBAT .
.
GMOMJOBID: $ZBAT.6
```

VOLUME Job Attribute

The VOLUME job attribute specifies the default node, volume, and subvolume for unqualified file references in a job's input file. You also can use the attribute to specify the default security for disk files created by the job.

```
V[OLUME] { \node. [ volume ] [ , " security " ]
[ \node. ] volume [ , " security " ]
[ \node. ] [ volume ] , " security ") }
```

node

is a node name.

volume

is one of:

$*volume-name.subvolume-name*

are volume and subvolume names.

$*volume-name*

is a volume name.

subvolume-name

is a subvolume name.

security

is a four-character string specifying Guardian security for read, write, execute, and purge (RWEP) file access. For each type of access, specify one of these codes to indicate the required security level:

Code	Description
A	Anyone—lets any local user access the file.
N	Network—lets any local or remote user access the file.
G	Group—lets any local user in the owner's group access the file.
C	Community—lets any local or remote user in the owner's group access the file.
O	Owner—lets only the local owner access the file.
U	User—lets the owner access the file from a local or remote node.
-	Hyphen—lets only the local super ID (255,255) access the file. Available as an option for write, execute, and purge access only.

Considerations

- The default VOLUME attribute values are the node, volume, subvolume, and security defaults set by the last SET JOB VOLUME command. If there was no such command, the default values are the defaults current when the BATCHCOM session began.
- The VOLUME job attribute always overrides the VOLUME attribute of defaults DEFINE =_DEFAULTS in a job's attachment set.

Example

This example shows the VOLUME attribute supplying the default volume and subvolume for partial file names in the file BKUP. The example also shows the attribute assigning file security to the file NEWFILE created by job I.

```
$DATA7.TEST 54> FILEINFO $TRASH.NBFILES.*
$TRASH.NBFILES
Code EOF Last Modification Owner RWEP …
OLDFILE 101 622 23-Oct-93 10:02:06 205,70 "N-N-" …
$DATA7.TEST 55> FUP COPY BKUP
DUP OLDFILE, NEWFILE
$DATA7.TEST 56> BATCHCOM $ZBAT
1} SHOW JOB
JOB ATTRIBUTES
volume: $DATA7.TEST, "NCNC"
user: 205,70
2} SET JOB VOLUME $TRASH.NBFILES, "OOOO"
3} SHOW JOB
JOB ATTRIBUTES
volume: $TRASH.NBFILES, "OOOO"
user: 205,70
4} SUBMIT JOB I, IN BKUP, EXECUTOR-PROGRAM FUP
Job I job number 460 submitted
5} STATUS JOB I
JOB JOBNAME USERID LOG STATE CLASSNAME
---- ---------------- ------- ---- ----------- ---------
460 I 205,70 58 EXECUTING DEFAULT
6} INFO JOB I, VOLUME, IN
JOB ATTRIBUTES for I
jobnumber: 460
volume: $TRASH.NBFILES, "OOOO"
in: $DATA7.TEST.BKUP
7} EXIT
$DATA7.TEST 57> FILEINFO $TRASH.NBFILES.*
$TRASH.NBFILES
Code EOF Last Modification Owner RWEP …
NEWFILE 101 622 11-Nov-93 10:15:34 205,70 "OOOO" …
OLDFILE 101 622 23-Oct-93 10:02:06 205,70 "N-N-" …
```

WAIT Job Attribute

The WAIT job attribute delays execution of a job for a specified period from the current time. (The current time is the system date and time on the node where the job's scheduler is running.)

```
WAIT hours [ : mins ]
```

hours

is a number in the range 0 through 168 specifying the number of hours delay (168 hours equals 7 days).

mins

is a number in the range 0 through 59 specifying the number of minutes delay.

Considerations

- The scheduler treats a job submitted without the AFTER, AT, or WAIT attributes like a job with the attribute WAIT 0:0. Such a job is eligible to run immediately on submission unless delayed by another attribute such as HOLD or TAPEDRIVES.
- The scheduler determines the next run time of a job with the WAIT attribute by adding the delay period to the submission time. For example, the run time of a job submitted at 1300 with a delay of three hours is 1600.

  To display the run time of a job, specify AFTER or AT in an INFO JOB command. For example:

  ```
  8} INFO JOB X, AFTER
  JOB ATTRIBUTES for X
  jobnumber: 26
  after: 14DEC94 23:00:00
  next-runtime: 14DEC94 23:00:00
  ```

  If the job's state is SPECIAL- *n*, BATCHOM does not display the run time.
- The AFTER, AT, and WAIT job attributes are mutually exclusive. This restriction means a job can have any one of the attributes, but not two or all three. By way of illustration, consider job X, whose attributes include AFTER 17:00. The job runs at 1615 if altered at 1610 by the command ALTER JOB X, WAIT 00:05.

  (If you mistakenly specify more than one of AFTER, AT, and WAIT in a command, the last attribute specified takes precedence. For example, job Y submitted at 0900 by the command SUBMIT JOB Y, AFTER 09:45, WAIT 00:15 runs at 0915.)

Examples

- This example shows the WAIT attribute delaying execution of a job by 10 minutes:

```
> TIME
September 10, 1993 10:30:40
> BATCHCOM $ZBAT; SUBMIT JOB A, WAIT 0:10
Job A job number 226 submitted
> BATCHCOM $ZBAT; INFO JOB A, AFTER
JOB ATTRIBUTES for A
jobnumber: 226
after: 10SEP93 10:40:48
next-runtime: 10SEP93 10:40:48
```

- This example shows the WAIT attribute overriding a job's AFTER attribute:

```
> TIME
September 10, 1993 10:35:10
> BATCHCOM $ZBAT; SUBMIT JOB B, AFTER 01JAN94
Job B job number 227 submitted
> BATCHCOM $ZBAT; INFO JOB B, AFTER
JOB ATTRIBUTES for B
jobnumber: 227
after: 01JAN94
next-runtime: 01JAN94 00:00:00
> TIME
September 10, 1993 10:35:53
> BATCHCOM $ZBAT; ALTER JOB B, WAIT 1
Job B job number 227 altered
> BATCHCOM $ZBAT; INFO JOB B, AFTER
JOB ATTRIBUTES for B
jobnumber: 227
after: 10SEP93 11:36:20
next-runtime: 10SEP93 11:36:20
```

WAITON Job Attribute

The WAITON job attribute specifies the names of up to 50 jobs on which execution of another job depends. A job with the WAITON attribute is a dependent job. The jobs on which it depends are master jobs. For more information on job dependencies, see Section 4, Job Planning, Submission, and Management.

```
WAITO[N] [ job-name [ case ]
( job-name [ case ] [ , job-name [ case ] ]… ) ]
```

job-name

is the name of a master job. Job *job-name* must be in the same scheduler as the dependent job when *case* specifies STOP or STOP-ABEND.

If you specify WAITON without *job-name* in an ALTER JOB command, the scheduler removes the WAITON attribute from the dependent job. Specifying

WAITON without *job-name* in a SET JOB command makes BATCHCOM remove the WAITON attribute from the working-attributes set.

case

is one of:

REL[EASE]

makes the dependent job wait for its master to release it with the ZBAT:RELEASE macro, the NBEXEC command $RELEASE, or the NetBatch programmatic command RELEASE JOB. (For information about the RELEASE JOB command, see the *NetBatch Management Programming Manual*.)

STO[P]

makes the dependent job wait for its master to release it on normal termination of the master. The release occurs automatically and does not require use of the ZBAT:RELEASE macro, the NBEXEC command $RELEASE, or the NetBatch programmatic command RELEASE JOB.

STO[P]-[ABEND]

makes the dependent job wait for its master to release it on normal or abnormal termination of the master. The release occurs automatically and does not require use of the ZBAT:RELEASE macro, the NBEXEC command $RELEASE, or the NetBatch programmatic command RELEASE JOB.

A STOP-ABEND release takes place only if the master job starts successfully.

Job *job-name* must be in the same scheduler as the dependent job when *case* specifies STOP or STOP-ABEND.

Omitting *case* makes the job wait for its master to release it with the ZBAT:RELEASE macro, the NBEXEC command $RELEASE, or the NetBatch programmatic command RELEASE JOB.

Considerations

- A dependent job cannot depend on itself (that is, you cannot specify a dependent job as one of its own masters). A dependent job can be a master of another dependent job.

  To prevent the scheduler from checking that dependent jobs do not wait on themselves, configure the NBFLAGS procedure to disable WAITON validation. For details, see on page 2-3.

- A dependent job does not run until released by all its masters. For the master jobs to release their dependents, each master must do one of:

  - Invoke the TACL macro ZBAT:RELEASE—if the master job's executor program is a TACL process.

- Execute an explicit or implicit $RELEASE command—if the master job's executor program is an NBEXEC process. NBEXEC automatically executes an implicit $RELEASE * command if the job runs without errors.
- Execute the NetBatch programmatic command RELEASE JOB.
- Terminate normally (if the dependent has the attribute `WAITON` *`master-job`* `STOP` or `WAITON` *`master-job`* `STOP-ABEND`) or abnormally (`WAITON` *`master-job`* `STOP-ABEND`).

For information about the ZBAT:RELEASE macro and $RELEASE command, see Section 4, Job Planning, Submission, and Management. For information about the RELEASE JOB command, see the *NetBatch Management Programming Manual.*

- Use the RELEASE-WAITON command to release dependent jobs from their masters without running the masters. For more information, see RELEASE-WAITON Command on page 6-115.
- A dependent job can depend on masters from its own scheduler and from other local and remote schedulers. (However, the dependent job must be in the same scheduler as the master job when `case` specifies STOP or STOP-ABEND.) To set up a dependency between a dependent job on one scheduler and a master job on another:
  1. Assign the WAITON attribute to the dependent job, specifying for *`job-name`* the name of the master job. (*`job-name`* cannot specify the master job's scheduler. Use the ZBAT:RELEASE macro or $RELEASE command for this.)
  2. Add the ZBAT:RELEASE or $RELEASE job-release command to the master job's input file, specifying the dependent job's scheduler and name. When the master job runs, it releases the specified dependent job.
- A dependent job cannot determine the schedulers of its masters because its WAITON attribute specifies master job names only. As a result, the dependent job can receive its release from any master running in any scheduler. For this reason, use unique names for all master and dependent jobs, regardless of the nodes on which they run.
- For an effective dependency to exist between two recurrent jobs, the dependent job should have the same execution interval as its master. (A recurrent job has the CALENDAR or EVERY attribute.) For example, a dependent job whose master has the attribute EVERY 21 DAYS should also have the attribute EVERY 21 DAYS.

  What happens when the execution interval differs depends on whether the master runs more or less frequently than the dependent:

  - If the master runs more frequently than the dependent, it releases the dependent each time it runs. As a result, the dependent runs only at its specified execution interval despite having received perhaps many releases from the master.

- If the master runs less frequently than the dependent, the dependent runs only when released by the master. As a result, the dependent runs at the same frequency as its master even though its execution interval specifies otherwise.

- Use of RUNNOW command on a dependent job, in order to run the dependent job without running the master job, is not recommended.

  If both the master job and the dependent job are recurring jobs and if the master runs less frequenly than the dependent, the dependent runs only when released by the master. As a result, the dependent runs at the same frequency as its master. But a dependent job in this case does not run on its release from the master job if it was already run using RUNNOW while it was in TIME or EVENT state.

  Use the RELEASE-WAITON command to release dependent jobs from their masters without running the masters.

Examples

- This example shows the submission of a dependent job with one master:

```
38} SUBMIT JOB A, WAITON B
Job A Jobnumber 11 submitted
39} INFO JOB A, WAITON
JOB ATTRIBUTES for A
jobnumber: 11
waiton: B, Not Released
```

- This example shows the submission of a dependent job with many masters and the release of that job by the RELEASE-WAITON command:

```
64} SET JOB WAITON (Y1 RELEASE, Y2 STOP, Y3 STOP-ABEND, Y4)
65} SUBMIT JOB Y0
Job Y0 Jobnumber 18 submitted
66} INFO JOB Y0, WAITON
JOB ATTRIBUTES for Y0
jobnumber: 18
waiton: Y1, Not Released
Y2 Stop, Not Released
Y3 Stop Abend, Not Released
Y4, Not Released
67} RELEASE-WAITON Y0 FROM *
Job Y0 Jobnumber 18 altered
68} INFO JOB Y0, WAITON
JOB ATTRIBUTES for Y0
jobnumber: 18
waiton: Y1, Released
Y2 Stop, Released
Y3 Stop Abend, Released
Y4, Released
```

- This example shows an ALTER JOB command changing the master jobs specified by a dependent job's WAITON attribute:

```
70} INFO JOB DEPENDENT-JOB, WAITON
JOB ATTRIBUTES for DEPENDENT-JOB
jobnumber: 19
waiton: MASTER-JOB-1, Not Released
MASTER-JOB-2, Not Released
71} ALTER JOB DEPENDENT-JOB, WAITON &
}} (MASTER-JOB-3, MASTER-JOB-4, MASTER-JOB-5)
Job DEPENDENT-JOB Jobnumber 19 altered
72} INFO JOB DEPENDENT-JOB, WAITON
JOB ATTRIBUTES for DEPENDENT-JOB
jobnumber: 19
waiton: MASTER-JOB-3, Not Released
MASTER-JOB-4, Not Released
MASTER-JOB-5, Not Released
```

- This example shows an ALTER JOB command removing a job's WAITON attribute:

```
74} INFO JOB DJ, WAITON
JOB ATTRIBUTES for DJ
jobnumber: 20
waiton: MJ1, Not Released
MJ2, Not Released
MJ3, Not Released
MJ4, Not Released
MJ5, Not Released
75} ALTER JOB DJ, WAITON
Job DJ Jobnumber 20 altered
76} INFO JOB DJ, WAITON
JOB ATTRIBUTES for DJ
jobnumber: 20
```

A Messages

This appendix describes the three types of NetBatch messages and provides detailed information on NetBatch messages in the range 512 through 2237:

Topic	Page
Message Types	A-2
Message Descriptions	A-3

The appendix does not contain information about these messages:

Message Numbers	Message type	For cause, effect, and recovery information, see:
-14— -1	Subsystem Programmatic Interface (SPI) errors	Descriptions of SPI errors in the *Guardian Procedure Errors and Messages Manual*
1—2555	File-system warnings and errors	Descriptions of file-system errors in the *Guardian Procedure Errors and Messages Manual*
260—380	NBEXEC errors	TFORM disk file NBEXDOC in the NetBatch installation subvolume
720—845	NetBatch-Plus errors	*NetBatch-Plus Reference Manual*
4512—4634	Sequential I/O errors	Descriptions of sequential I/O errors in the *Guardian Procedure Errors and Messages Manual* (To match a Guardian sequential I/O error with a NetBatch sequential I/O error, subtract 4000 from the NetBatch error number.)
6049—6081	DEFINE errors	Descriptions of DEFINE errors in the *Guardian Procedure Errors and Messages Manual* (To match a Guardian DEFINE error with a NetBatch DEFINE error, subtract 4000 from the NetBatch error number.)

Message Types

There are three types of NetBatch messages: error, informational, and warning.

Error Messages

Error messages (prefix *nnnn*-E) indicate NetBatch software did not perform the requested function and give a reason. For example, this error message appears if you try to delete an executing, over-limit, or suspended job with the DELETE JOB command:

```
2077-E Job is executing or suspended; DELETE command ignored
```

Informational Messages

Informational messages (prefix *nnnn*-I) confirm NetBatch software performed the requested function, but the function had no effect. For example, this informational message appears if you enter an FC, !, or ? command that specifies a nonexistent line:

```
0541-I No such line
```

Warning Messages

Warning messages (prefix *nnnn*-W) indicate NetBatch software performed the requested function, but a condition exists that might affect the function's successful outcome. For example, this warning message appears if you submit a job to a scheduler without specifying the job's input file:

```
0513-W IN file does not exist; create and secure as required
```

Message Descriptions

This subsection contains cause, effect, and recovery information for NetBatch messages with numbers in the range 512 through 2237.

0512-W

```
Other users can WRITE or PURGE the IN file; resecure if
required
```

Cause. The IN job attribute specified an input file to which other users have write orpurge access.

Effect. The command executed successfully. However, users with write access to the input file can alter any attribute of or delete the job using the file. With the file as a medium, these users also can assume your level of security. As a result, they could modify the input file to purge your files, change your password, and so on.

Recovery. Secure the file against write and purge access by using the Safeguard distributed security management facility or the Guardian standard security system. For information about Safeguard security, see the *Safeguard Reference Manual*. For information about Guardian security, see the *Guardian User's Guide*.

0513-W

```
IN file does not exist; create and secure as required
```

Cause. The IN job attribute specified a nonexistent input file.

Effect. The command executed successfully, but the job abends when it runs if its executor program requires an input file.

Recovery. Not applicable unless the executor program requires an input file. In that case, alter the IN attribute to specify an existing input file. Alternatively, create the specified file before the job runs.

0514-W

```
The EXECUTOR-PROGRAM file-name does not exist
```

Cause. The EXECUTOR-PROGRAM job attribute specified a nonexistent program file.

Effect. The command executed successfully, but the job fails when it runs if the specified file does not exist at that time. The scheduler puts the job in the SPECIAL-3 state when the job fails.

Recovery. Alter the EXECUTOR-PROGRAM attribute to specify an existing program file or create the specified file before the job runs.

0515-W

```
ALTER CLASS expects INITIATION ON or OFF
```

Cause. The ALTER CLASS command omitted the INITIATION attribute completely or specified INITIATION without a value. The command must specify the INITIATION attribute in full.

Effect. The command failed.

Recovery. Retry the command, specifying the attribute INITIATION OFF or INITIATION ON.

0516-W

```
Executor executor-name has already started
```

Cause. The START EXECUTOR command specified a started executor (state ON) instead of a stopped executor (state OFF).

Effect. None.

Recovery. Not applicable.

0517-W

```
Job job-name is currently executing or suspended; altered
attribute(s) might not affect current execution
```

Cause. The ALTER JOB command operated on an executing, over-limit, or suspended job (state EXECUTING, OVER LIMIT, or SUSPENDED).

Effect. The command executed successfully, but only these attributes affect the job: HOLDAFTER, IFFAILS, PURGE-IN-FILE, RESTART, STALL, and STOP-ON-ABEND. If the job is recurrent, all altered attributes apply the next time the job runs. (A recurrent job has the CALENDAR or EVERY attribute.)

Recovery. Not applicable.

0518-W

```
Job has already been released
```

Cause. The dependent job received a second or subsequent release from its master or a job with the same name as its master.

Effect. None.

Recovery. Not applicable unless the names of the jobs receiving the releases conflict. In that case, make sure the jobs in the master-dependent relationship have names that no other jobs use at any time.

0522-W

```
File file-name cannot be referenced over network
```

Cause. The remote volume specified by the VOLUME job attribute contains eight or more characters (including the dollar ($) sign). Remote volume names cannot be more than seven characters long (including $).

Effect. The command failed.

Recovery. Retry the command, specifying a seven-character remote volume name in the VOLUME attribute. Alternatively, include the TACL VOLUME command in the job's input file if the job's executor program is the TACL program.

0524-W

```
Requested tape drives are unavailable. Assigning remaining
drives to your executing or suspended job.
```

Cause. The ALTER JOB, TAPEDRIVES command specified, for an executing, over-limit, or suspended job, more tape drives than were available.

Effect. The command executed successfully, but the scheduler assigned only spare tape drives to the job, not the required number. The scheduler reserves the spare drives for the job (thus preventing other jobs from using them), but the drive shortfall remains.

Recovery. Use the SUSPEND JOB command to suspend the job if it is executing or over limit. Next, use the STATUS SCHEDULER command to monitor tape drives availability. When the number of drives available equals the drive shortfall, retry the ALTER JOB command, specifying the full drive requirement in the TAPEDRIVES attribute.

0525-W

```
Executor executor-name not started; CPU cpu-number is not
available
```

Cause. The START EXECUTOR command specified an executor whose CPU is unavailable.

Effect. None.

Recovery. Retry the command when the CPU is available or after altering the executor's CPU attribute to specify an available CPU.

0526-W

```
Executor executor-name has stopped, or will stop or be deleted
when current job finishes
```

Cause. The STOP EXECUTOR command specified a stopped executor (state OFF) or an executor in the STOP or DELETE state.

Effect. None.

Recovery. Not applicable.

0527-W

```
You have no READ access to IN file
```

Cause. The IN job attribute specified an input file to which you have no read access.

Effect. The command executed successfully, but the job abends when it runs if its executor program requires an input file.

Recovery. Not applicable unless the executor program requires an input file. In that case, ask the owner of the file to resecure it for read access before the job runs. Alternatively, alter the IN attribute to specify an input file to which you have read access.

0528-W

```
You have no WRITE access to OUT file or OUT file does not
exist
```

Cause. The OUT job attribute specified a nonexistent output file or an output file to which you have no write access.

Effect. The command executed successfully, but the job abends when it runs if its executor program is incapable of creating or writing to the output file.

Recovery. Create the output file or ask the file's owner to resecure it for write access before the job runs.

0529-W

```
You have no EXECUTE access to EXECUTOR-PROGRAM file-name
```

Cause. The EXECUTOR-PROGRAM job attribute specified a program file to which you have no execute access. You need execute access to the file for the job to start.

Effect. The command executed successfully, but the job fails when the scheduler tries to start it. The scheduler puts the job in the SPECIAL-3 state on failure.

Recovery. Alter the EXECUTOR-PROGRAM attribute before the job starts to specify a program file to which you have execute access. Alternatively, ask your system administrator to give you execute access to the required program.

0530-W

```
You have no PURGE access to IN file
```

Cause. The value of the PURGE-IN-FILE job attribute is ON, but you do not have purge access to the job's input file.

Effect. The command executed successfully. However, the scheduler does not purge the job's input file when it deletes the job.

Recovery. Not applicable unless you want the scheduler to purge the input file. In that case, ask the owner of the file to resecure it for purge access before the scheduler deletes the job.

0531-W

```
Error file-system-error-number opening CALENDAR file.
```

Cause. The CALENDAR job attribute specified a calendar file the scheduler could not open because of a file-system error.

Effect. The command executed successfully. The scheduler put the job in the SPECIAL-7 state, however, thus preventing the job from running.

Recovery. To resolve the SPECIAL-7 state and make the job ready to run:

1. Correct the condition indicated by error *file-system-error-number*. For information on the cause of the error, see the descriptions of file-system errors in the *Guardian Procedure Errors and Messages Manual*.
2. Make sure the job's CALENDAR attribute specifies the correct calendar file.
3. Alter the job's HOLD attribute to HOLD OFF by using the ALTER JOB command.

0532-W

```
CALENDAR file has expired.
```

Cause. The CALENDAR job attribute specified a calendar file that does not contain any future times.

Effect. The command executed successfully. The scheduler put the job in the SPECIAL-8 state, however, thus preventing the job from running.

Recovery. To resolve the SPECIAL-8 state and make the job ready to run:

1. Regenerate the specified calendar file with future times or alter the CALENDAR attribute to specify another file containing future times.
2. Alter the job's HOLD attribute to HOLD OFF by using the ALTER JOB command.

0534-W

```
Deleted temporary attachment set ( user-ID) attachment-set-ID.
No jobs were referencing this attachment set.
```

Cause. The scheduler automatically deleted the attachment set because the set had the TEMPORARY ON attribute and was not in use by jobs.

Effect. The scheduler deleted the attachment set.

Recovery. Not applicable.

0535-I

```
Your user code does not give you access to
( user-ID) attachment-set-ID data
```

Cause. The command specified an attachment set to which you have no access.

Effect. None.

Recovery. Not applicable.

0536-I

```
=_DEFAULTS cannot be deleted
```

Cause. The DELETE ATTACHMENT-SET command specified the defaults DEFINE=_DEFAULTS. You cannot delete this DEFINE.

Effect. None.

Recovery. Not applicable.

0537-W

```
( user-ID) attachment-set-ID does not exist
```

Cause. The command specified a nonexistent attachment set.

Effect. The command failed.

Recovery. Retry the command, specifying an existing attachment set.

0540-W

```
Job job-name job number job-number will run when tapes are
available
```

Cause. The RUNNOW JOB command operated on a job whose TAPEDRIVES attribute specified more drives than are available.

Effect. The command executed successfully, but the job does not run until the required drives become available.

Recovery. Not applicable.

0541-I

```
No such line
```

Cause. The FC, !, or ? command specified a nonexistent line in BATCHCOM's history buffer.

Effect. None.

Recovery. Retry the command after checking for the correct line in the history buffer by using the HISTORY command.

0542-W

```
DEFINE names beginning with =_ZBAT reserved for Tandem use
```

Cause. An attachment-set command specified a DEFINE whose name began with =_ZBAT. HP reserves DEFINE names beginning with =_ZBAT for its own use.

Effect. The command executed successfully, but the scheduler rejected the DEFINE.

Recovery. Retry the command, specifying a valid DEFINE name (first character an equals sign (=) and second character a letter).

0543-W

```
The specified job is not waiting on master(s)
```

Cause. The job is not dependent on the specified masters.

Effect. None.

Recovery. Recovery depends on the requirements of the job owners but might include:

- Adding the WAITON attribute to the nondependent job to make it dependent on one or more of the masters (if a dependency was meant to exist between the jobs)
- Retrying the command with the intended dependent job specified

0544-W

```
SWAP, EXTSWAP, LIB, EXECUTOR-PROGRAM must be on same system
```

Cause. The job's EXTSWAP, LIB, or SWAP attribute specified a file on a different node from that of the job's executor program. The attributes must specify files on the same node as the executor program.

Effect. The command failed.

Recovery. Retry the command, specifying extended swap, swap, and library files on the same node as the executor program.

0545-W

```
The specified job was waiting on another job(s)
```

Cause. The master job sent a release to a dependent job whose WAITON attribute did not specify that master.

Effect. None.

Recovery. Recovery depends on the requirements of the master job's owner but might involve one of:

- Altering the dependent job's WAITON attribute to make the job dependent on the master (if a dependency was meant to exist between the jobs)
- Changing the dependent job specified by the master job to the intended job

0547-W

```
SWITCHCPU deferred until no jobs reading startup messages
```

Cause. The SWITCHCPU SCHEDULER command was issued while jobs were reading their startup messages.

Effect. The command executed successfully, but the actual CPU change does not occur until no more jobs are reading startup messages.

Recovery. Not applicable.

0548-W

```
AT/AFTER specifies a time in the past.
```

Cause. A job was submitted or altered to run at or after a time in the past.

Effect. The job runs.

Recovery. Not applicable.

0549-I

```
process-handles-count Process handles omitted
```

Cause. In response to a STATUS command, the *process-handle-count* process handles started by a NetBatch job could not be accommodated in the Reply buffer due to resource limitations.

Effect. None. The command completes successfully but displays this informational message.

Recovery. To find out the job whose process handles have been omitted:

1. Set the Batchcom command Display-Spi to ON.
2. Issue the STATUS JOB command. From the SPI information, detemine the job that returns the informational message.

   This is the job for which process handles have been omitted.
3. Set the Batchcom command Display-Spi to OFF.
4. To see the maximum possible process handles for the job, use STATUS JOB *job-number/job-name*, DETAIL.

2048-E

```
Job is not suspended; ACTIVATE command ignored.
```

Cause. The ACTIVATE JOB command specified a nonsuspended job. The command only operates on suspended jobs (state SUSPENDED).

Effect. The command failed.

Recovery. Not applicable.

2050-E

```
In AFTER date: YEAR must be in the range 1993 to 2525
```

Cause. The AFTER job attribute's date option specified a year value outside the allowable range 1993 through 2525.

Effect. The command failed.

Recovery. Retry the command, specifying a year value in the range 1993 through 2525.

2051-E

```
In AFTER date: MONTH must be in the range 1 to 12
```

Cause. The AFTER job attribute's date option specified a month value outside the allowable range 1 through 12.

Effect. The command failed.

Recovery. Retry the command, specifying a month value in the range 1 through 12.

2052-E

```
In AFTER date: DAY must be in the range 1 to 31
```

Cause. The AFTER job attribute's date option specified a day value outside the allowable range 1 through 31.

Effect. The command failed.

Recovery. Retry the command, specifying a day value in the range 1 through 31.

2053-E

```
In AFTER time: HOUR must be in the range 0 to 23
```

Cause. The AFTER job attribute's time option specified an hour value outside the allowable range 0 through 23.

Effect. The command failed.

Recovery. Retry the command, specifying an hour value in the range 0 through 23.

2054-E

```
In AFTER time: MINUTE must be in the range 0 to 59
```

Cause. The AFTER job attribute's time option specified a minute value outside the allowable range 0 through 59.

Effect. The command failed.

Recovery. Retry the command, specifying a minute value in the range 0 through 59.

2055-E

```
Scheduler is already started; command ignored.
```

Cause. The START SCHEDULER command operated on a started scheduler.

Effect. None.

Recovery. Not applicable.

2056-E

```
AT-ALLOWED is currently OFF; submit AFTER time
```

Cause. The scheduler has the attribute AT-ALLOWED OFF. This attribute prevents use of the RUNNOW JOB command and submission of jobs with the AT attribute.

Effect. The command failed.

Recovery. Retry the command after your system administrator alters the AT-ALLOWED scheduler attribute to AT-ALLOWED ON. Alternatively, use the RUNNEXT JOB command or submit the job with the AFTER attribute.

2057-E

```
AT-ALLOWED invalid value
```

Cause. A requester specified an invalid value for the scheduler's AT-ALLOWED attribute.

Effect. The command failed.

Recovery. Modify the requester to specify an AT-ALLOWED value of -1 (true) or 0 (false).

2058-E

```
In AT date: YEAR must be in the range 1993 to 2525
```

Cause. The AT job attribute's date option specified a year value outside the allowable range 1993-2525.

Effect. The command failed.

Recovery. Retry the command, specifying a year value in the range 1993 through 2525.

2059-E

```
In AT date: MONTH must be in the range 1 to 12
```

Cause. The AT job attribute's date option specified a month value outside the allowable range 1 through 12.

Effect. The command failed.

Recovery. Retry the command, specifying a month value in the range 1 through 12.

2060-E

```
In AT date: DAY must be in the range 1 to 31
```

Cause. The AT job attribute's date option specified a day value outside the allowable range 1 through 31.

Effect. The command failed.

Recovery. Retry the command, specifying a day value in the range 1 through 31.

2061-E

```
In AT time: HOUR must be in the range 0 to 23
```

Cause. The AT job attribute's time option specified an hour value outside the allowable range 0 through 23.

Effect. The command failed.

Recovery. Retry the command, specifying an hour value in the range 0 through 23.

2062-E

```
In AT time: MINUTE must be in the range 0 to 59
```

Cause. The AT job attribute's time option specified a minute value outside the allowable range 0 through 59.

Effect. The command failed.

Recovery. Retry the command, specifying a minute value in the range 0 through 59.

2063-E

```
BACKUPCPU out of range 0 to 15, or * for any CPU
```

Cause. The BACKUPCPU scheduler attribute specified a nonexistent CPU on the scheduler's node.

Effect. The command failed.

Cause. Retry the command, specifying a CPU configured for the scheduler's node.

2064-E

```
Primary CPU must not equal Backup CPU
```

Cause. The CPUs specified by the BACKUPCPU scheduler attribute are the same.

Effect. The command failed.

Recovery. Retry the command, specifying different CPUs.

2066-E

CALENDAR file *file-name* is not a valid file name

Cause. A requester specified an invalid file name for the CALENDAR job attribute.

Effect. The command failed.

Recovery. Modify the requester to specify a valid file name.

2068-E

```
CALENDAR file is not a valid calendar file; it must be created
by BATCHCAL
```

Cause. The CALENDAR job attribute specified a file that was not generated by BATCHCAL.

Effect. The command failed.

Recovery. Retry the command, specifying a file generated by BATCHCAL.

2069-E

```
Use ADD SCHEDULER to create and initialize the database
```

Cause. The file JOB was missing from the scheduler's database during a warm start of the scheduler.

Effect. The warm start failed.

Recovery. Cold start the scheduler.

2071-E

```
CLASS-COUNT must be in the range 1 to 8 inclusive
```

Cause. The CLASS executor attribute specified more than eight classes for the executor. The maximum number of classes allowed for an executor is eight.

Effect. The command failed.

Recovery. Retry the command, specifying no more than eight classes in the CLASS attribute.

2072-E

```
An executor must have at least one CLASS assigned to it
```

Cause. A requester specified the executor attribute CLASS without any accompanying class names.

Effect. The command failed.

Recovery. Modify the requester to specify class names for the CLASS attribute.

2073-E

```
Invalid SPI CONTEXT token, command terminated
```

Cause. A requester sent an invalid context token.

Effect. The command failed.

Recovery. Modify the requester to send a valid context token.

2074-E

```
CPU must be cpu-number to cpu-number
```

Cause. The CPU executor attribute specified a nonexistent CPU on the scheduler's node.

Effect. The command failed.

Recovery. Retry the command, specifying a CPU configured for the scheduler's node (that is, a CPU in the range *cpu-number* to *cpu-number*).

2075-E

```
HOLDAFTER invalid value
```

Cause. The ZHOLD-AFTER field of ZBAT-MAP-DEF-JOB specified an invalid value for the job's HOLDAFTER attribute.

Effect. The command failed.

Recovery. Retry the command, setting the ZHOLD-AFTER field to a valid Boolean value (for example, ZSPI-VAL-TRUE or ZSPI-VAL-FALSE).

2076-E

```
CPU must be specified
```

Cause. The ADD EXECUTOR command omitted the executor's CPU attribute.

Effect. The command failed.

Recovery. Retry the command, specifying the CPU attribute.

2077-E

```
Job is executing or suspended; DELETE command ignored.
```

Cause. The DELETE JOB command specified an executing, over-limit, or suspended job (state EXECUTING, OVER LIMIT, or SUSPENDED). The command only operates on jobs whose states are EVENT, READY, RUNNEXT, RUNNOW, SPECIAL-*n*, TAPE, or TIME.

Effect. The command failed.

Recovery. Use the STOP JOB command to stop the job. Alternatively, retry the DELETE JOB command, specifying a job whose state is EVENT, READY, RUNNEXT, RUNNOW, SPECIAL- n, TAPE, or TIME.

2078-E

```
You must specify a frequency greater than zero for EVERY
```

Cause. The ZEVERY-DAYS field of ZBAT-MAP-DEF-JOB had a null value, and the ZEVERY-HOURS and ZEVERY-MINUTES fields specified zero hours and zero minutes.

Effect. The command failed.

Recovery. Retry the command, setting ZEVERY-DAYS to a number in the range 1 through 365. Alternatively, set ZEVERY-HOURS and ZEVERY-MINUTES to specify a value greater than zero hours and zero minutes and then retry the command. (Valid values for ZEVERY-HOURS are numbers in the range 0 through 168. Valid values for ZEVERY-MINUTES are numbers in the range 0 through 59.).

2079-E

```
EVERY must be in the range { 1 to 365 | 0 to 168 | 0 TO 59 }
inclusive
```

Cause. The EVERY job attribute's DAYS or HOURS option specified an execution interval outside the allowable range. This range for DAYS is 1 through 365. For HOURS, 0 through 168 (hours) and 0 through 59 (minutes).

Effect. The command failed.

Recovery. Retry the command, specifying a DAYS or HOURS value in the allowable range.

2080-E

```
You cannot specify both CALENDAR and EVERY attributes for a
job
```

Cause. The command specified two or more of the CALENDAR attribute, the EVERY attribute, and ZBAT-MAP-DEF-CRONTAB. CALENDAR, EVERY, and ZBAT-MAP-DEF-CRONTAB are mutually exclusive.

Effect. The command failed.

Recovery. Retry the command, specifying only one of CALENDAR, EVERY, and ZBAT-MAP-DEF-CRONTAB.

2082-E

```
EXECUTOR-PROGRAM file-name is not a valid file name
```

Cause. A requester specified an invalid file name for the EXECUTOR-PROGRAM job attribute.

Effect. The command failed.

Recovery. Modify the requester to specify a valid file name.

2085-E

```
A job cannot be directly or indirectly dependent on itself
```

Cause. The WAITON job attribute specified the dependent job as one of that job's own masters. A dependent job cannot depend on itself.

Effect. The command failed.

Recovery. Retry the command, omitting the dependent job from the master jobs specified by the WAITON attribute.

2086-E

```
EXECUTOR executor-name already exists
```

Cause. The ADD EXECUTOR command specified the name of an existing executor.

Effect. The command failed.

Recovery. Retry the command, specifying a unique executor name.

2087-E

```
EXECUTOR executor-name does not exist
```

Cause. The executor command specified a nonexistent executor.

Effect. The command failed.

Recovery. Retry the command, specifying an existing executor.

2088-E

```
The EXECUTOR file is full
```

Cause. The scheduler attempted to add an executor to its database in response to an ADD EXECUTOR command. The attempt failed because the EXECUTOR file that records executor details was full.

Effect. The command failed.

Recovery. Retry the command after your system administrator increases the maximum extents of the EXECUTOR file by using the FUP ALTER command. (The default maximum extents for the file is 100.)

2090-E

```
Received multiple tokens when only one is expected
```

Cause. A requester sent more tokens in a command than the scheduler required.

Effect. The command failed.

Recovery. Modify the requester to send only allowed tokens.

2091-E

```
HOLD invalid value
```

Cause. A requester specified an invalid value for a job's HOLD attribute.

Effect. The command failed.

Recovery. Modify the requester to specify a HOLD value of -1 (true) or 0 (false).

2092-E

```
IFFAILS invalid value
```

Cause. A requester specified an invalid value for a job's IFFAILS attribute.

Effect. The command failed.

Recovery. Modify the requester to specify an IFFAILS value of -1 (true) or 0 (false).

2093-E

```
IN file file-name is an invalid file name
```

Cause. A requester specified an invalid file name for the IN job attribute.

Effect. The command failed.

Recovery. Modify the requester to specify a valid file name.

2095-E

```
INITIATION invalid value
```

Cause. A requester specified an invalid value for a class's INITIATION attribute.

Effect. The command failed.

Recovery. Modify the requester to specify an INITIATION value of -1 (true) or 0 (false).

2096-E

```
Guardian user is undefined
```

Cause. Someone deleted the logged-on user after the last CHANGEUSER command or, if no such command was issued, after the current BATCHCOM session began.

Effect. The command failed.

Recovery. Retry the command after using the CHANGEUSER command to log on as a different user.

2098-E

```
Either the JOB or CHKQUE file is full
```

Cause. The scheduler attempted to add a job to its database in response to a SUBMIT JOB command. The attempt failed because either the JOB or CHKQUE file that records job details was full. This error can occur:

- When one of the files contains the maximum number of job records (32767 for systems running G-series RVUs, and 9999 for systems running H-series RVUs)
- When someone reduces the maximum extents specified for one of the files after scheduler database creation

Effect. The command failed.

Recovery. Retry the command after your system administrator does either or both of:

- Deletes unwanted jobs from the database by using the DELETE JOB and STOP JOB commands.
- Increases the maximum extents of the JOB and CHKQUE files by using the FUP ALTER command. (For systems running G-series RVUs, the default maximum extent is 400 for JOB and 64 for CHKQUE. For systems running H-series RVUs, the default maximum extent is 100 for JOB and 16 CHKQUE.)

2099-E

```
JOB does not exist
```

Cause. The job command specified a nonexistent job.

Effect. The command failed.

Recovery. Retry the command, specifying an existing job.

2102-E

```
CLASS class-name already exists
```

Cause. The ADD CLASS command specified the name of an existing class.

Effect. The command failed.

Recovery. Retry the command, specifying a unique class name.

2103-E

```
The CLASS file is full
```

Cause. The scheduler attempted to add a class to its database in response to an ADD CLASS command. The attempt failed because the JOBCLASS file that records class details was full.

Effect. The command failed.

Recovery. Retry the command after your system administrator increases the maximum extents of the JOBCLASS file by using the FUP ALTER command. (The default maximum extents for the file is 100.)

2104-E

```
CLASS class-name is in use by one or more executors. To delete
the CLASS, you must remove it from the executors first.
```

Cause. The DELETE CLASS command specified a class assigned to one or more executors. The command deletes only a class dissociated from its executors.

Effect. The command failed.

Recovery. Retry the command after dissociating the class from its executors by using the ALTER EXECUTOR command.

2105-E

```
CLASS class-name does not exist
```

Cause. The class command specified a nonexistent class.

Effect. The command failed.

Recovery. Retry the command, specifying an existing class.

2106-E

```
Job has invalid name job-name
```

Cause. The command specified an invalid job name.

Effect. The command failed.

Recovery. Retry the command, specifying a valid job name. For information on the form of a job name, see SUBMIT JOB Command on page 6-183.

2107-E

```
Job job-name already exists
```

Cause. The SUBMIT JOB command specified the name of an existing job.

Effect. The command failed.

Recovery. Retry the command, specifying a unique job name.

2108-E

```
You must specify a job name
```

Cause. The command did not specify the name of a job (a required syntax item).

Effect. The command failed.

Recovery. Retry the command, specifying a job name.

2110-E

```
Job number must be in the range 1 to 32767 inclusive
```

Cause. The command specified a job number outside the allowable range 1 through 32767.

Effect. The command failed.

Recovery. Retry the command, specifying a job number in the range 1 through 32767.

Note. This error message is applicable only for systems running G-series RVUs. For systems running H-series RVUs, the job number range is 1 to 9999. The error message and the recovery method change accordingly.

2117-I

```
No { ATTACHMENT SET | CLASS | EXECUTOR | JOB } selected
```

Cause. The command used wild-card characters to specify a range of attachment-set, class, executor, or job names. No names matched the wild-card specification or names that did match were of records to which you have no access.

Effect. None.

Recovery. Not applicable.

2118-E

```
MAXPRINTLINES is out of range 120 to 65534
```

Cause. The MAXPRINTLINES job attribute specified a maximum number of print lines outside the allowable range 120 through 65534.

Effect. The command failed.

Recovery. Retry the command, specifying a MAXPRINTLINES value in the range 120 through 65534.

2119-E

```
MAXPRINTPAGES is out of range 2 to 65534
```

Cause. The MAXPRINTPAGES job attribute specified a maximum number of print pages outside the allowable range 2 through 65534.

Effect. The command failed.

Recovery. Retry the command, specifying a MAXPRINTPAGES value in the range 2 through 65534.

2120-E

```
MAXRESP must be in the range MAXRESP-range inclusive
```

Cause. A requester sent a maximum-response token with an invalid value.

Effect. The command failed.

Recovery. Modify the requester to send the token with a value in the range *MAXRESP-range*.

2121-E

```
You must specify attributes with this command
```

Cause. The command did not specify any attributes (required syntax items).

Effect. The command failed.

Recovery. Retry the command, specifying one or more attributes.

2122-E

```
You must specify a CLASS
```

Cause. The command omitted the CLASS attribute.

Effect. The command failed.

Recovery. Retry the command, specifying the CLASS attribute.

2123-E

```
Only one CONTEXT token is allowed per command
```

Cause. A requester sent two or more context tokens in a command where the scheduler expected only one.

Effect. The command failed.

Recovery. Modify the requester to send one context token.

2124-E

```
Only one MAP token is allowed per command
```

Cause. A requester sent two or more map tokens in a command where the scheduler expected only one.

Effect. The command failed.

Recovery. Modify the requester to send one map token.

2125-E

```
Only one MAXRESP token is allowed per command
```

Cause. A requester sent two or more maximum-response tokens in a command where there should be only one such token.

Effect. The command failed.

Recovery. Modify the requester to send one maximum-response token.

2126-E

```
The job number and job name specified are not the same job.
Job job-number is job-name.
```

Cause. An internal error prevented scheduler validation of the job name or number.

Effect. The command failed.

Recovery. Report the error to your HP representative.

2127-E

```
You must specify either the job name or the job number
```

Cause. The job command did not specify the job on which it was to operate.

Effect. The command failed.

Recovery. Retry the command, specifying a job name or number.

2128-E

```
SUBMIT-ALLOWED is OFF; the scheduler is not accepting job
submissions
```

Cause. The scheduler has the attribute SUBMIT-ALLOWED OFF.

Effect. The command failed.

Recovery. Retry job submission after your system administrator alters the SUBMIT-ALLOWED scheduler attribute to SUBMIT-ALLOWED ON.

2129-E

```
The command command-keyword is invalid with the object object-
keyword
```

Cause. *command-keyword* is not an object-keyword command.

Effect. The command failed.

Recovery. Retry the command, specifying the correct object keyword for the command or the correct command for the object keyword.

2131-E

```
Use START SCHEDULER to enable the scheduler for processing
```

Cause. The command operated on a scheduler that has not started.

Effect. The command failed.

Recovery. Retry the command after your system administrator makes the scheduler available for use by means of the START SCHEDULER command. Alternatively, retry the command after opening a started scheduler with the OPEN command.

2132-E

```
Your user code does not give you access to
{ ( user-ID) attachment-set-ID | job-name }
```

Cause. The command specified an attachment set or job to which you have no access.

Effect. The command failed.

Recovery. Not applicable.

2133-E

```
The scheduler is being SHUTDOWN; command ignored.
```

Cause. The command operated on a scheduler that was shutting down.

Effect. The command failed.

Recovery. Not applicable.

2136-E

```
OUT file file-name is not a valid file name
```

Cause. A requester specified an invalid file name for the OUT job attribute.

Effect. The command failed.

Recovery. Modify the requester to specify a valid file name.

2137-E

```
PRI must be in the range 1 to 199 inclusive
```

Cause. The PRI job attribute specified an execution priority outside the range 1 through 199.

Effect. The command failed.

Recovery. Retry the command, specifying an execution priority in the range 1 through 199.

2139-E

```
RESTART invalid value
```

Cause. A requester specified an invalid value for a job's RESTART attribute.

Effect. The command failed.

Recovery. Modify the requester to specify a RESTART value of -1 (true) or 0 (false).

2140-E

```
STOP-ON-ABEND invalid value
```

Cause. A requester specified an invalid value for a job's STOP-ON-ABEND attribute.

Effect. The command failed.

Recovery. Modify the requester to specify a STOP-ON-ABEND value of -1 (true) or 0 (false).

2141-E

```
Job job-name job number job-number is not in EVENT, READY,
TAPE, TIME, or RUNNOW state. Command ignored.
```

Cause. The RUNNEXT JOB command specified a job whose state was EXECUTING, OVER LIMIT, RUNNEXT, SPECIAL- *n*, or SUSPENDED. The command works only for a job whose state is EVENT, READY, RUNNOW, TAPE, or TIME.

Effect. The command failed.

Recovery. Not applicable.

2142-E

```
Job job-name job number job-number is not in EVENT, READY,
TAPE, TIME, or RUNNEXT state. Command ignored.
```

Cause. The RUNNOW JOB command specified a job whose state was EXECUTING, OVER LIMIT, RUNNOW, SPECIAL- *n*, or SUSPENDED. The command works only for a job whose state is EVENT, READY, RUNNEXT, TAPE, or TIME.

Effect. The command failed.

Recovery. Not applicable.

2143-E

```
Scheduler cannot write to Edit-format log file, file-name
```

Cause. The SWITCHLOG SCHEDULER command specified an EDIT file as the scheduler's log file. EDIT files cannot be log files.

Effect. The command failed.

Recovery. Retry the command, specifying one of these as the log file: a device; a process; or any type of unstructured disk file except an EDIT file; or a nonexistent disk file.

2144-E

```
SELPRI must be in range 0 to 7
```

Cause. The SELPRI job attribute specified an invalid selection priority.

Effect. The command failed.

Recovery. Retry the command, specifying a selection priority in the range 0 through 7.

2145-E

```
STARTUP-MESSAGE is invalid
```

Cause. ZBAT-TKN-STARTUP-MESSAGE specified a startup message containing more than 961 characters or omitted the message altogether.

Effect. The command failed.

Recovery. Retry the command, changing ZBAT-TKN-STARTUP-MESSAGE to specify a startup message containing no more than 961 characters.

2146-E

```
Job is not executing or suspended; STOP command ignored.
```

Cause. The STOP JOB command specified a job whose state was EVENT, READY, RUNNEXT, RUNNOW, SPECIAL- *n*, TAPE, or TIME. The command operates only on executing, over-limit, and suspended jobs (states EXECUTING, OVER LIMIT, and SUSPENDED).

Effect. The command failed.

Recovery. Use the DELETE JOB command to delete the job. Alternatively, retry the STOP JOB command, specifying a job whose state is EXECUTING, OVER LIMIT, or SUSPENDED.

2147-E

```
SUBMIT-ALLOWED invalid value
```

Cause. A requester specified an invalid value for the SUBMIT-ALLOWED attribute.

Effect. The command failed.

Recovery. Modify the requester to specify a SUBMIT-ALLOWED value of -1 (true) or 0 (false).

2148-E

```
Job is not executing; SUSPEND command ignored.
```

Cause. The SUSPEND JOB command specified a job that was not executing or over limit. The command only operates on executing and over-limit jobs (states EXECUTING and OVER LIMIT).

Effect. The command failed.

Recovery. Retry the command, specifying an executing or over-limit job.

2149-E

```
TAPEDRIVES must be in range 0 to 99
```

Cause. The TAPEDRIVES attribute specified a number outside the range 0 through 99.

Effect. The command failed.

Recovery. Retry the command, specifying a TAPEDRIVES number in the range 0 through 99.

2150-E

```
Command command-keyword is invalid
```

Cause. *command-keyword* is not available in your version of the NetBatch product.

Effect. None.

Recovery. Not applicable.

2151-E

```
Object object-keyword is invalid
```

Cause. The *object-keyword* object is not a NETBATCH object. NETBATCH objects are ATTACHMENT-SET, CLASS, EXECUTOR, JOB, and SCHEDULER.

Effect. The command failed.

Recovery. Retry the command, specifying an allowable NETBATCH object.

2153-E

```
Invalid token token-code in SPI buffer
```

Cause. A requester sent a token not recognized by the scheduler.

Effect. The command failed.

Recovery. Modify the requester to send the correct token.

2154-E

```
You must specify a VOLUME
```

Cause. A requester did not specify the VOLUME job attribute.

Effect. The command failed.

Recovery. Modify the requester to specify the VOLUME attribute.

2155-E

```
The volume \node.vol.subvol is not a valid volume or
subvolume
```

Cause. The VOLUME job attribute specified a nonexistent volume or subvolume on node \ *node*.

Effect. The command failed.

Recovery. Retry the command, specifying for the VOLUME attribute the names of a volume and a subvolume available on \ *node*.

2157-E

```
RELEASE failed on one or more dependent jobs matching the
specification
```

Cause. The master job did not release all its dependent jobs because the job range specified by the ZBAT:RELEASE macro in the master's input file excluded some of those jobs.

Effect. The master released only some of its dependents.

Recovery. Ask the master job's owner to check and, where appropriate, to change the job range specified by the ZBAT:RELEASE macro. Then perform whatever actions the owner requests on the unreleased dependent jobs.

2158-E

```
Count of WAITON jobs must be in range 0 to 50
```

Cause. The WAITON job attribute specified more than 50 master jobs. A dependent job can have a maximum of 50 masters.

Effect. The command failed.

Recovery. Retry the command, specifying no more than 50 jobs in the WAITON attribute.

2160-E

```
WAITON job job-name is a duplicate name
```

Cause. The WAITON job attribute specified two or more job names that are the same.

Effect. The command failed.

Recovery. Retry the command, specifying unique job names for the WAITON attribute.

2161-E

```
The specified job is not waiting on any jobs
```

Cause. The ZBAT:RELEASE macro invoked by the master job sent a release to a nondependent job (a job without the WAITON attribute).

Effect. None.

Recovery. Recovery depends on the requirements of the master job's owner but might include:

- Adding the WAITON attribute to the nondependent job to make it dependent on the master (if a dependency was meant to exist between the jobs)
- Changing the job specified by the ZBAT:RELEASE macro to the intended job

2162-E

```
The specified job was waiting on another job(s)
```

Cause. The ZBAT:RELEASE macro invoked by the master job sent a release to a dependent job whose WAITON attribute did not specify that master.

Effect. None.

Recovery. Recovery depends on the requirements of the master job's owner but might involve one of:

- Altering the dependent job's WAITON attribute to make the job dependent on the master (if a dependency was meant to exist between the jobs)
- Changing the job specified by the ZBAT:RELEASE macro to the intended job

2166-E

```
The SSID subsystem-ID sent to NetBatch contains the wrong
subsystem ID. NetBatch uses NetBatch-subsystem-ID.
```

Cause. A requester specified an invalid value for the NetBatch subsystem ID.

Effect. The command failed.

Recovery. Modify the requester to specify a subsystem ID of *NetBatch-subsystem-ID*.

2167-E

```
Cannot switch CPU without a backup process
```

Cause. The scheduler attempted to switch CPUs in response to a SWITCHCPU SCHEDULER command. The attempt failed because the scheduler was running without a backup in the only available CPU on its node.

Effect. The command failed.

Recovery. The scheduler automatically creates its backup when another CPU becomes available.

2168-E

```
Error file-system-error-number trying to switch to log file
file-name
```

Cause. The scheduler attempted to switch log files in response to a SWITCHLOG SCHEDULER command. The attempt was unsuccessful because of a file-system error.

Effect. The command failed.

Recovery. Retry the command after correcting the condition indicated by error *file-system-error-number*. For information on the cause of the error, see the descriptions of file-system errors in the *Guardian Procedure Errors and Messages Manual*.

2169-E

```
File file-name was created by a pre C20 scheduler
```

Cause. The C20 version or later version of the scheduler that you tried to warm start had a file in its database created with a version of the scheduler earlier than the C20 version.

Effect. The warm start failed.

Recovery. Retry the warm start after running the UPDATENB file conversion program supplied with the C20 version of the NetBatch product. For information about running the program, see the software release document (softdoc) for NetBatch product version C20.

2170-E

```
CONVERTTIMESTAMP Error CONVERTTIMESTAMP-procedure-error-num-
ber;
check daylight savings time (DST) table
```

Cause. The AFTER or AT job attribute specified a run time in a daylight-saving time (DST) transition period, resulting in a CONVERTTIMESTAMP procedure error.

Effect. The command failed.

Recovery. Retry the command, specifying a run time outside the DST transition period. (Your system administrator will tell you when the period occurs.) Alternatively, retry the command after your system administrator corrects the CONVERTTIMESTAMP error condition. For information about the cause of the error, see the description of the CONVERTTIMESTAMP procedure in the *Guardian Procedure Errors and Messages Manual*.

2171-E

```
( user-ID) attachment-set-name already exists
```

Cause. The ADD ATTACHMENT-SET command specified the name of an existing attachment set.

Effect. The command failed.

Recovery. Retry the command, specifying a unique attachment-set name.

2172-E

```
( user-ID) attachment-set-ID does not exist
```

Cause. The attachment-set command specified a nonexistent attachment set.

Effect. The command failed.

Recovery. Retry the command, specifying an existing attachment set.

2173-E

```
( user-ID) attachment-set-ID is referenced by one or more
jobs
```

Cause. The DELETE ATTACHMENT-SET command specified an attachment set in use by one or more jobs.

Effect. The command failed.

Recovery. Retry the command after using the ALTER JOB, ATTACHMENT-SET command to dissociate the attachment set from the jobs using it. You can list the jobs using the set with the STATUS ATTACHMENT-SET command.

2174-E

```
( user-ID) attachment-set-ID is referenced by requester BATCH-
COM-process-ID
```

Cause. The DELETE ATTACHMENT-SET command specified an attachment set created by a BATCHCOM process (*BATCHCOM-process-ID*) that is still running.

Effect. The command failed.

Recovery. Retry the command after stopping the set's BATCHCOM creator.

2175-E

```
invalid-attachment-set-ID Attachment ID is invalid
```

Cause. A requester specified an invalid attachment-set ID.

Effect. The command failed.

Recovery. Modify the requester to specify a valid attachment-set ID.

2176-E

```
( user-ID) attachment-set-ID error file-system-error-number
```

Cause. The scheduler could not load DEFINEs from attachment set (*user-ID*)*attachment-set-ID* into the program file space of the job's executor-program process because of file-system error *file-system-error-number*.

Effect. The command failed.

Recovery. Retry the command after correcting the file-system error condition.

2177-E

```
( user-ID) attachment-set-ID storage overflow
```

Cause. An internal storage overflow momentarily prevented the scheduler from updating the attachment-set record.

Effect. The command failed.

Recovery. Retry the command.

2178-E

```
Attachment set was updated during multiple replies to this
command. Retry the command
```

Cause. The attachment-set command referred to a set the scheduler was updating in response to another command.

Effect. The command failed.

Recovery. Retry the command.

2179-E

```
This scheduler version supports only one attachment set per
job
```

Cause. The ATTACHMENT-SET job attribute specified more than one attachment set for the job. Your NetBatch scheduler allows only one attachment set per job.

Effect. The command failed.

Recovery. Retry the command, specifying only one attachment set.

2188-E

```
NETBATCH Internal error: P = program-counter-register, E =
environment-register. Contact your Tandem representative
```

Cause. The scheduler abended while processing your request.

Effect. The scheduler abended.

Recovery. Warm start the scheduler, then report the error to your HP representative.

2189-E

```
Error file-system-error-number, File: file-name
```

Cause. A file-system error on file *file-name* prevented command execution.

Effect. The command failed.

Recovery. Retry the command after correcting the condition indicated for file file-name by error *file-system-error-number*. For information on the cause of the error, see the descriptions of file-system errors in the *Guardian Procedure Errors and Messages Manual*.

2190-E

```
Invalid NetBatch ID attachment-set-name
```

Cause. The command specified an attachment-set name in the wrong form.

Effect. The command failed.

Recovery. Retry the command, specifying the attachment-set name in the correct form. For information on attachment-set names, see ADD ATTACHMENT-SET Command on page 6-34 and ALTER ATTACHMENT-SET Command on page 6-52.

2191-E

```
SCHEDULER or requester version does not support all
attributes--
Use the latest version of BATCHCOM to display or specify
attributes.
```

Cause. The command specified one or more attributes that the scheduler or requester does not support.

Effect. The command failed.

Recovery. Retry the command, specifying only those attributes supported by the scheduler and requester.

2192-E

```
SPI Error SPI-error-number, invalid SPI request
```

Cause. A requester sent an invalid SPI request to the scheduler.

Effect. The command failed.

Recovery. Determine the cause of the error by checking the documentation for SPI procedures SSGET and SSGETTKN in the *SPI Programming Manual*. Modify the requester if the error comes from the requester. Report the error to your HP representative if it comes from the scheduler.

2193-E

```
Invalid NetBatch name { class-name | executor-name }
```

Cause. The command specified a class or executor name in the wrong form.

Effect. The command failed.

Recovery. Retry the command, specifying the class or executor name in the correct form. For information about class and executor names, see ADD CLASS Command on page 6-40 and ADD EXECUTOR Command on page 6-42.

2194-E

```
Cannot SUSPEND job-ID
```

Cause. One of these file-system errors occurred when the scheduler tried to suspend job *job-ID* in response to a SUSPEND JOB command:

File-system error number	Description
11	Process does not exist
48	Security violation
201	Unable to communicate with the process's CPU

Effect. The command failed.

Recovery. Retry the command after determining from the job's log file which file-system error occurred and, where necessary, correcting the error condition. For information about the cause of the error, see the descriptions of file-system errors in the *Guardian Procedure Errors and Messages Manual*.

2195-E

```
Cannot ACTIVATE job-ID
```

Cause. One of these file-system errors occurred when the scheduler tried to reactivate job *job-ID* in response to an ACTIVATE JOB command:

File-system error number	Description
11	Process does not exist
48	Security violation
201	Unable to communicate with the process's CPU

Effect. The command failed.

Recovery. Retry the command after determining from the job's log file which file-system error occurred and, where necessary, correcting the error condition. For information about the cause of the error, see the descriptions of file-system errors in the *Guardian Procedure Errors and Messages Manual*.

2196-E

```
Cannot STOP job-ID
```

Cause. A file-system error occurred when the scheduler tried to stop job *job-ID* in response to a STOP JOB command.

Effect. The command failed.

Recovery. Retry the command after determining from the job's log file which file-system error occurred and, where necessary, correcting the error condition. For information on the cause of the error, see the descriptions of file-system errors in the *Guardian Procedure Errors and Messages Manual*.

2197-E

```
STALL inv val
```

Cause. The ZSTALL field of ZBAT-MAP-DEF-JOB specified an invalid value for the job's STALL attribute.

Effect. The command failed.

Recovery. Retry the command, setting the ZSTALL field to a valid Boolean value (for example, ZSPI-VAL-TRUE or ZSPI-VAL-FALSE).

2198-E

```
Wild-card names are not supported for this command
```

Cause. The RUNNEXT JOB or RUNNOW JOB command used wild-card characters to specify a range of job names. These commands do not support wild-card character searching in your version of the NetBatch product.

Effect. The command failed.

Recovery. Retry the command, specifying the full name or number of a single job.

2199-E

```
Greater than 3 attachment sets specified for job
```

Cause. The ATTACHMENT-SET job attribute specified more than three attachment sets for the job. Your NetBatch scheduler allows only three attachment sets per job.

Effect. The command failed.

Recovery. Retry the command, specifying no more than three attachment sets.

2200-E

```
Invalid Date specification
```

Cause. The ZDATE field of ZBAT-MAP-DEF-JOB specified values for some but not all of ZYEAR, ZMONTH, and ZDAY. The field must specify values for each of ZYEAR, ZMONTH, and ZDAY.

Effect. The command failed.

Recovery. Retry the command, changing ZDATE to specify values for each of ZYEAR, ZMONTH, and ZDAY.

2201-E

```
Invalid Time specification
```

Cause. The ZTIME field of ZBAT-MAP-DEF-JOB specified values for some but not all of ZHOUR, ZMINUTE, ZSECOND, ZMILLISECOND, and ZMICROSECOND. The field must specify values for each of ZHOUR, ZMINUTE, ZSECOND, ZMILLISECOND, and ZMICROSECOND.

Effect. The command failed.

Recovery. Retry the command, changing ZTIME to specify values for each of ZHOUR, ZMINUTE, ZSECOND, ZMILLISECOND, and ZMICROSECOND.

2202-E

```
AT inv val
```

Cause. The ZAT-FLAG field of ZBAT-MAP-DEF-JOB specified an invalid value for the job's AT flag.

Effect. The command failed.

Recovery. Retry the command, setting the ZAT-FLAG field to a valid Boolean value (for example, ZSPI-VAL-TRUE or ZSPI-VAL-FALSE).

2203-E

```
You must specify an EXECUTOR
```

Cause. The executor command omitted ZBAT-TKN-SEL-EXECUTORNAME (a required token) or specified ZBAT-TKN-SEL-EXECUTORNAME without a valid value.

Effect. The command failed.

Recovery. Retry the command, setting ZBAT-TKN-SEL-EXECUTORNAME to a valid value.

2204-E

```
You must specify an attachment set ID
```

Cause. The attachment-set command omitted ZBAT-TKN-ATT-SET-ID (a required token) or specified ZBAT-TKN-ATT-SET-ID without a valid value.

Effect. The command failed.

Recovery. Retry the command, setting ZBAT-TKN-ATT-SET-ID to a valid value.

2205-E

```
You must specify the RELEASE MAP
```

Cause. The RELEASE JOB command omitted ZBAT-MAP-PAR-RELEASE-JOB (a required token) or specified ZBAT-MAP-PAR-RELEASE-JOB without valid values.

Effect. The command failed.

Recovery. Retry the command, specifying ZBAT-MAP-PAR-RELEASE-JOB with valid values.

2206-E

```
( user-ID) attachment-set-ID ASSIGN-name error
```

Cause. The attachment-set command specified an invalid ASSIGN name.

Effect. The command failed.

Recovery. Retry the command, specifying a valid ASSIGN name.

2207-E

```
( user-ID) attachment-set-ID DEFINE-details error DEFINE-
error-number
```

Cause. The attachment-set command generated a DEFINE error.

Effect. The command failed.

Recovery. Retry the command after correcting the condition indicated by *DEFINE-error-number*. For information on the cause of the error, see the descriptions of DEFINE errors in the *Guardian Procedure Errors and Messages Manual*.

2208-E

```
( user-ID) attachment-set-ID PARAM-name error
```

Cause. The attachment-set command specified an invalid PARAM name.

Effect. The command failed.

Recovery. Retry the command, specifying a valid PARAM name.

2209-E

```
Duplicate Attachment Sets
```

Cause. The command specified a duplicate attachment-set name.

Effect. The command failed.

Recovery. Retry the command without the duplicate attachment-set name.

2210-E

```
In AFTER time: SECOND must be in the range 0 to 59
```

Cause. The AFTER job attribute's time option specified a seconds value outside the allowable range 0 through 59.

Effect. The command failed.

Recovery. Retry the command, specifying a seconds value in the range 0 through 59.

2211-E

```
In AFTER time: MILLISECS must be in the range 0 to 999
```

Cause. The AFTER job attribute's time option specified a milliseconds value outside the allowable range 0 through 999.

Effect. The command failed.

Recovery. Retry the command, specifying a milliseconds value in the range 0 through 999.

2212-E

```
In AFTER time: MICROSECS must be in the range 0 to 999
```

Cause. The AFTER job attribute's time option specified a microseconds value outside the allowable range 0 through 999.

Effect. The command failed.

Recovery. Retry the command, specifying a microseconds value in the range 0 through 999.

2213-E

```
Invalid class name class-name
```

Cause. The command specified an invalid class name.

Effect. The command failed.

Recovery. Retry the command, specifying a valid class name. For information on the form of a class name, see ADD CLASS Command on page 6-40.

2214-E

```
Invalid WAITON name job-name
```

Cause. The command specified an invalid master-job name.

Effect. The command failed.

Recovery. Retry the command, specifying a valid name of a master job.

2215-E

```
Invalid executor name executor-name
```

Cause. The command specified an invalid executor name.

Effect. The command failed.

Recovery. Retry the command, specifying a valid executor name. For information on the form of an executor name, see ADD EXECUTOR Command on page 6-42.

2216-E

```
ALTER CLASS expects INITIATION ON or OFF
```

Cause. The ALTER CLASS command did not specify the INITIATION attribute.

Effect. The command failed.

Recovery. Retry the command, specifying the INITIATION attribute.

2217-E

```
Total JOB attributes exceeded storage storage-capacity
```

Cause. The accumulated size of these job attributes exceeded the scheduler's internal-storage capacity:

DESCRIPTION	IN	NAME	SWAP
EXECUTOR-PROGRAM	LIB	OUT	TERM
EXTSWAP	LOG	STARTUP	VOLUME

Effect. The command failed.

Recovery. Reduce the size of or delete the DESCRIPTION attribute and retry the command. If the error recurs, reduce the size of or delete one by one and in order of increasing importance other items listed in the cause of this message (starting with the STARTUP attribute) and retry the command.

2218-E

```
Crontab inv val
```

Cause. One or more of the EVERY attribute's crontab-entry fields specified an invalid value.

Effect. The command failed.

Recovery. Retry the command, specifying valid values for all crontab-entry fields. For more information, see EVERY Job Attribute on page 7-55.

2219-E

```
PURGE-IN-FILE inv val
```

Cause. The ZPURGE-IN-FILE field of ZBAT-MAP-DEF-JOB specified an invalid value for the job's PURGE-IN-FILE attribute.

Effect. The command failed.

Recovery. Retry the command, setting the ZPURGE-IN-FILE field to a valid Boolean value (for example, ZSPI-VAL-TRUE or ZSPI-VAL-FALSE). Total job attributes exceeded storage capacity

2220-E

```
HIGHPIN inv val
```

Cause. The ZHIGHPIN field of ZBAT-MAP-DEF-JOB or ZBAT-MAP-DEF-SCHEDULER specified an invalid value.

Effect. The command failed.

Recovery. Retry the command, setting the ZHIGHPIN field to a valid Boolean value (for example, ZSPI-VAL-TRUE or ZSPI-VAL-FALSE).

2221-E

```
POSIX inv val
```

Cause. The ZPOSIX field of ZBAT-MAP-DEF-JOB specified an invalid value.

Effect. The command failed.

Recovery. Retry the command, using procedure SSNULL to initialize to null values the fields of ZBAT-MAP-DEF-JOB.

2222-E

```
SAVEABEND inv val
```

Cause. The ZSAVEABEND field of ZBAT-MAP-DEF-JOB specified an invalid value for the job's SAVEABEND attribute.

Effect. The command failed.

Recovery. Retry the command, setting the ZSAVEABEND field to a valid Boolean value (for example, ZSPI-VAL-TRUE or ZSPI-VAL-FALSE).

2223-E

```
RUND inv val
```

Cause. The ZRUND field of ZBAT-MAP-DEF-JOB specified an invalid value for the job's RUND attribute.

Effect. The command failed.

Recovery. Retry the command, setting the ZRUND field to a valid Boolean value (for example, ZSPI-VAL-TRUE or ZSPI-VAL-FALSE).

2224-E

```
JOBID inv val
```

Cause. The ZJOBID-ZERO field of ZBAT-MAP-DEF-JOB specified an invalid value for the job's JOBID-ZERO attribute.

Effect. The command failed.

Recovery. Retry the command, setting the ZJOBID-ZERO field to a valid Boolean value (for example, ZSPI-VAL-TRUE or ZSPI-VAL-FALSE).

2225-E

```
MEM must be in range 0 to 64
```

Cause. The MEM job attribute specified a number of memory pages outside the allowable range 0 through 64.

Effect. The command failed.

Recovery. Retry the command, specifying a MEM value in the range 0 through 64.

2226-E

```
LIMIT must be in range hours-range to minutes-range
```

Cause. The ZTIME-LIMIT field of ZBAT-MAP-DEF-JOB specified a time limit outside the allowable range *hours-range* to *minutes-range*.

Effect. The command failed.

Recovery. Retry the command, specifying a time limit in the range hours-range to minutes-range.

2227-E

```
DESCRIPTION too long
```

Cause. The size of ZBAT-TKN-DESCRIPTION exceeded 1000 bytes.

Effect. The command failed.

Recovery. Retry the command, reducing the size of ZBAT-TKN-DESCRIPTION to 1000 bytes or less.

2228-E

```
Too many selectors
```

Cause. The accumulated size of the specified job-selection criteria exceeded 600 bytes.

Effect. The command failed.

Recovery. Retry the command, reducing the size of or deleting job-selection criteria.

2229-E

```
Bad Nodename node
```

Cause. The ZLOCALNAMES field of ZBAT-MAP-DEF-SCHEDULER specified an invalid remote node.

Effect. The command failed.

Recovery. Recovery. Retry the command, specifying a valid remote node.

2230-E

```
MAXPRI must be from 1 to 199
```

Cause. The MAX-PRI scheduler attribute specified a maximum priority outside the allowable range 1 through 199.

Effect. The command failed.

Recovery. Retry the command, specifying a MAX-PRI value in the range 1 through 199.

2231-E

```
MAXCONCURRENT must be from 0 to 500
```

Cause. The MAX-CONCURRENT-JOBS scheduler attribute specified a concurrent-jobs limit outside the allowable range 0 through 500.

Effect. The command failed.

Recovery. Retry the command, specifying a concurrent-jobs limit in the range 0 through 500.

2232-E

```
MAXTEMPEXECUTORS must be from 0 to 500
```

Cause. The MAX-CONCURRENT-JOBS scheduler attribute specified a temporary-executors limit outside the allowable range 0 through 500.

Effect. The command failed.

Recovery. Retry the command, specifying a temporary-executors limit in the range 0 through 500.

2233-E

```
EVERY-CATCHUP inv val
```

Cause. The ZEVERY-CATCHUP field of ZBAT-MAP-DEF-SCHEDULER specified an invalid value for the scheduler's CATCHUP attribute.

Effect. The command failed.

Recovery. Retry the command, setting the ZEVERY-CATCHUP field to a valid Boolean value (for example, ZSPI-VAL-TRUE or ZSPI-VAL-FALSE).

2234-E

```
EMS inv val
```

Cause. The ZEMS field of ZBAT-MAP-DEF-SCHEDULER specified an invalid value for the scheduler's EMS attribute.

Effect. The command failed.

Recovery. Retry the command, setting the ZEMS field to a valid ZBAT-DDL-EMS value.

2235-E

```
PFS range 131072-1048576
```

Cause. The PFS job attribute specified a nonzero process-file-segment size outside the allowable range 131,072 through 1,048,576 bytes.

Effect. The command failed.

Recovery. Retry the command, specifying a process-file-segment size of zero bytes or 131,072 through 1,048,576 bytes.

2236-E

```
No help available for topic
```

Cause. The HELP command specified a topic for which no help is available.

Effect. The command failed.

Recovery. Not applicable.

2237-E

```
Command not configured
```

Cause. The NBFLAGS procedure bound into the BATCHLIB library file disabled the command. For information on the procedure, see on page 2-3.

Effect. None.

Recovery. Not applicable.

B NBEXEC

This appendix describes NBEXEC, the NetBatch executor program:

Topic	Page
Introducing NBEXEC	B-1
Command and Variable Reference Summary	B-3
NBEXEC Syntax Summary	B-5

Note. For detailed information on the NBEXEC program, print the TFORM disk file NBEXDOC in the NetBatch installation subvolume. NBEXDOC describes NBEXEC's features and functions, and describes and gives examples of the syntax, operation, and results of NBEXEC commands and variables.

Introducing NBEXEC

NBEXEC is the program-file ID of the NetBatch executor program. Compatible with BPROC (the batch execution process of obsolete product MIS Batch), NBEXEC executes control-file commands, supplies data to started processes, and logs process output. NBEXEC can run as a NonStop process pair and offers a simple-but-powerful job control language that includes error testing and job-recovery facilities.

TACL Alternative

NBEXEC is an easy-to-learn alternative to the TACL program for job programming. It also simplifies job migration from MIS Batch to NetBatch by letting you move MIS Batch jobs to a NetBatch environment without rewriting them.

Application Design Tool

NBEXEC lets programmers define user variables and perform dynamic string substitution while a job is executing. Input to user programs can be part of the job.

Improved Resource Usage

NBEXEC can detect errors during execution. This feature enables programmers to embed error testing routines and conditional branching instructions in the job definition. If an error occurs, these instructions can prevent further, unnecessary processing and so improve system productivity.

Automatic Restart

NBEXEC can automatically restart processing of a control file from a user-defined checkpoint if a CPU failure stops a currently executing waited process. Where a control file contains multiple checkpoints, NBEXEC restarts processing from the last checkpoint reached before CPU failure.

Log Files

Each NBEXEC process logs to its OUT file details of the interaction between the process, the control file, and controlled processes. Each OUT file has three parts:

- A timestamped header that lists the job's defaults and execution environment
- A timestamped list of the commands executed and their responses
- Termination lines that record the completion status of the job

For a sample NBEXEC log file, see the example in High PIN Capabilities.

NBEXEC Processing

NBEXEC performs these functions when it runs as the executor program of a job:

- Reads and validates the job's control file. If the control file is invalid, NBEXEC logs errors and terminates the job.
- Starts and runs the job according to user-specified options.
- Executes the instructions contained in the control file and performs any string substitutions in control-file commands.
- Recognizes abnormal termination of application processes and starts any user-defined error recovery.
- Gathers program output in the OUT file (unless directed elsewhere).

High PIN Capabilities

NBEXEC processes can have high PIN creators, and the :PPD command can detect named high PIN processes in NBEXEC versions D20 and later. NBEXEC versions earlier than D20 do not have high PIN capabilities.

Example

This example shows submission to a scheduler of an NBEXEC job that starts another scheduler. The example lists the job's control file, shows the BATCHCOM command that submits the job, and displays the resulting NBEXEC log file.

```
> FUP COPY ZBATWARM
$KILL OFF
$ALLOW NOERRORS
:PPD $ZBAT
$IF NOT #ERROR $GOTO END
:NETBATCH /NAME $ZBAT, NOWAIT/ $DATA7.ZBAT
$IF #ERROR $GOTO END
:BATCHCOM $ZBAT; START SCHEDULER
.END
8 RECORDS TRANSFERRED
> BATCHCOM $SCHD; SUBMIT JOB ZBAT-WARM-START, EXECUTOR
```

```
PROGRAM
NBEXEC, IN $NB.INFILES.ZBATWARM, OUT [#MYTERM], STARTUP "ID
D940920, LIMIT 00:10"
Job ZBAT-WARM-START Jobnumber 1 submitted
> PAUSE
NBEXEC T9190D30 - (31OCT94^01JUN94) - 20 Sep 1994 09:06:12
(C)1986 Tandem (C)2004 Hewlett Packard Development Company,
L.P.
JOB ID: D940920
SYSTEM SUPERVISOR: \MELBDEV.$SCHD
JOB NUMBER: 1
NBEXEC PROCESS NAME: \MELBDEV.$X630
HOME TERMINAL: Self
CURRENT VOLUME: \MELBDEV.$NB.INFILES
USERID: 255,255
USERNAME: SUPER.SUPER
SECURITY: NCNU
XPRI: 147
CONTROL FILE NAME: \MELBDEV.$NB.INFILES.ZBATWARM
LIST FILE NAME: \MELBDEV.$ZTN0.#PTY4
TIME LIMIT: 00:10
09:06:14 $KILL OFF
09:06:14 $ALLOW NOERRORS
09:06:14 :PPD $ZBAT
0274-E Process does not exist, \MELBDEV.$ZBAT
09:06:14 FAIL $IF NOT #ERROR $GOTO END
09:06:14 :NETBATCH /NAME $ZBAT, NOWAIT/ $DATA7.ZBAT
09:06:50 $IF #ERROR $GOTO END
09:06:50 :BATCHCOM $ZBAT; START SCHEDULER
Scheduler started
09:07:02 .END
09:07:03 BATCH ending
```

Command and Variable Reference Summary

Command Identifiers

NBEXEC command identifiers are:

- $ (dollar sign) indicates a control-file command. These commands alter the control-file processing sequence, determine control-file output, send messages to other processes and devices, and set parameters. Control-file commands are:

$ALLOW	$GOTO	$LIST	$RELEASE	$SETS
$CHAIN	$IF	$MESSAGE	$RETURN	
$CHKPNT	$KILL	$PAGE	$SETL	

- : (colon) indicates a command-interpreter command. These commands get information about and control a job's execution environment, and start and control processes. They also can create disk files, execute TACL commands in a disk file, and add comments to a control file. Command-interpreter commands are:

:ACTIVATE	:DELETE DEFINE	:PURGE	:SUSPEND
:ADD DEFINE	:FILES	:RENAME	:SYSTEM
:ASSIGN	:LOGON	:RUN	:TIME
:COMMENT	:OBEY	:SET DEFINE	:VOLUME
:CLEAR	:PARAM	:STATUS	:WHO
:CREATE	:PPD	:STOP	

- ! (exclamation point) indicates a nonexecutable comment.
- * (asterisk) indicates data that NBEXEC passes to a process it starts.
- . (period) indicates a label.

NBEXEC-Defined Variables

NBEXEC defines several logical and string variables:

- NBEXEC logical variables return values that are either true or false:

  #ERROR #NEXTFILENAME
  #NEXTOPEN #NEXTROLL-FORWARD
  #NEXTAUDIT #NEXTLICENSE
  #NEXTPROGID

- NBEXEC string variables return information such as the time or date. They are:

  #BP #NEXTBLOCK #NEXTRECLEN #NEXTUSER

  #BPID #NEXTCODE #NEXTSECURE #PID

  #CC #NEXTDATE #NEXTSIZE #TI

  #HOME #NEXTEXTSIZE #NEXTTIME #TIME

  #JOBID #NEXTNAME #NEXTTYPE #TODAY

- When using a string variable to return a value in a control file, enclose the variable in single quotes (' '); for example, '#TODAY'. NBEXEC replaces the quotes and variable name with the value of the variable. Quotes are not required when testing the value of the variable, as in a $IF command (for example, $IF #JOBID …, not $IF '#JOBID' …).

NBEXEC Syntax Summary

This subsection provides a quick reference to NBEXEC command and variable syntax. The table includes the syntax of the TACL RUN command that runs the NBEXEC program.

Command to Run NBEXEC

```
[ RUN ] [ \node. ] [ volume. ] NBEXEC / NAME [ $process-name ] ,
IN file-name ,
OUT [ list-file ] [ , run-option ]… / [ , program-parameter ]…
```

program-parameter is one of these run parameters:

A[PPEND]

specifies that NBEXEC is to append output to *list-file* if the file exists. Omitting APPEND causes NBEXEC to overwrite *list-file*.

B[ACKUPCPU]{ *cpu-number* -1 }

specifies the backup CPU for an NBEXEC process that is to run as a NonStop process.

cpu-number specifies any available CPU but must not specify the same CPU as that of the executor under which the NBEXEC process runs.

-1 specifies the highest-numbered available CPU.

I[D] { *job-ID* T[ERM] }

ID *job-ID* specifies a job identifier that appears in *list-file*. *job-ID* can contain up to eight alphanumeric characters, but the first character must be a letter.

ID TERM causes NBEXEC to emulate a physical terminal rather than a process.

L[ABEL] *label*

causes NBEXEC to begin processing of *file-name* at the specified label in the file.

L[IMIT] [*h*] *h*:[*m*] *m*

specifies a time limit (range 00:00 through 99:59) for the NBEXEC job.

P[URGE]

causes NBEXEC to purge *list-file* if the job terminates successfully.

S[YNTAX]

causes NBEXEC to validate the syntax of, but not execute, statements in *file-name*.

Control-file ($) Commands

- $ALLOW determines whether NBEXEC allows or disallows errors when executing control-file commands:

  ```
  $ALLOW { ERRORS NOERRORS }
  ```

- $CHAIN causes NBEXEC to execute the next command from a specified file or label:

  ```
  $CHAIN { file-name , LABEL label file-name , LABEL label }
  ```

- $CHKPNT defines a restart point in the NBEXEC control file or declares there is no current restart point:

  ```
  $CHKPNT { OFF ON }
  ```

- $GOTO causes NBEXEC to continue processing from a specified label:

  ```
  $GOTO label
  ```

- $IF causes NBEXEC to test a condition and transfer command processing if the condition is true:

  ```
  $IF [ NOT ] { logical-variable $GOTO label #NEXTFILENAME
  $GOTO label
  string-identifier { <
  <=
  =
  ==
  >
  >= } { " literal " string-identifier}
  $GOTO label }
  ```

  string-identifier specifies a string identifier in the form:

  ```
  variable-name [ @ edit-start ] [ : edit-length ]
  ```

- $KILL determines whether NBEXEC terminates processes running with the NOWAIT option when the NBEXEC process stops:

  ```
  $KILL { OFF ON }
  ```

- $LIST determines what NBEXEC logs to its output file:

  ```
  $LIST { BATCH  OFF ON }
  ```

- $MESSAGE causes NBEXEC to send a message to $0 or to a specified device or process:

  ```
  $MESSAGE [ OPR $device-name $process-name ]
  [ , NOHEAD ] [ , WAIT centiseconds ] / [ text ]
  ```

- $PAGE determines whether NBEXEC issues a new page on receiving an OPEN or CLOSE system message from a process it started:

  ```
  $PAGE { OFF ON }
  ```

- $RELEASE causes NBEXEC to release one or more dependent NetBatch jobs:

  ```
  $RELEASE [ [ \node. ] $schd , ] job-name-range
  ```

- $RETURN causes NBEXEC to resume command execution from the line following the most recent $CHAIN command:

  ```
  $RETURN
  ```

- $SETL sets a specified logical variable:

  ```
  $SETL logical-variable { FALSE TRUE }
  ```

- $SETS sets a specified string variable:

  ```
  $SETS string-variable string-variable-value
  ```

Command-Interpreter (:) Commands

- :ACTIVATE reactivates a suspended process or process pair:

  ```
  :ACTIVATE [ [ \node. ] {$process-name cpu, pin, } ]
  ```

- :ADD DEFINE creates a DEFINE in the process file segment of the current NBEXEC process. NBEXEC passes the DEFINE to processes it starts.

  ```
  :ADD DEFINE DEFINE-name-1 [ [ , LIKE DEFINE-name-2 ] [ ,
  attribute-spec ]… ]
  ```

  For a description of *attribute-spec*, see the description of the ADD DEFINE command in the *TACL Reference Manual.*

- :ASSIGN assigns names of actual files to logical file names used in programs and optionally specifies the characteristics of such files. When you omit the command's parameters, the command displays the assigned values for all current assignments. NBEXEC passes ASSIGNs specified in a job's control file to processes the job starts.

  ```
  :ASSIGN [ logical-unit , actual-file-name [ , create-open-
  spec ]… ]
  ```

  For a description of *logical-unit*, *actual-file-name*, and *create-open-spec*, see the description of the ASSIGN command in the *TACL Reference Manual.*

- :CLEAR deletes ASSIGNs set with the :ASSIGN command and PARAMs set with the :PARAM command:

  ```
  :CLEAR { ALL ALL ASSIGN ALL PARAM ASSIGN logical-unit PARAM
  PARAM-name }
  ```

For a description of *logical-unit*, see the description of the CLEAR command in the *TACL Reference Manual*.

- :COMMENT causes NBEXEC to ignore the rest of the command line. (As an alternative to :COMMENT, use an exclamation point.)

  ```
  :COMMENT [ comment-text ]
  ```

- :CREATE creates an unstructured disk fileL

  ```
  :CREATE file-name [ , extent-size [ , sec-extent-size ] ]
  ```

- :DELETE DEFINE deletes one or more DEFINEs from the process file segment of the current NBEXEC process:

  ```
  :DELETE DEFINE { DEFINE-name ** }
  ```

- :FILES displays the names of files that match a specified file-name template:

  ```
  :FILES [ file-name-template ] [ INFO,DEFER ]
  [ [NOT] filter [ , [NOT] filter ]… ]
  ```

  For a description of *file-name-template*, see the description of the FILENAMES command in the *TACL Reference Manual*.

- :LOGON changes the user during an NBEXEC session:

  ```
  :LOGON { group-name.user-name
  group-ID, user-ID } [ , password ]
  ```

- :OBEY causes NBEXEC to start a TACL process that executes the commands in a specified input file:

  ```
  :O[BEY] file-name
  ```

- :PARAM creates a parameter and gives it a value, or displays all current parameters and their values. NBEXEC passes PARAMs specified in a job's control file to processes the job starts.

  ```
  :PARAM [ PARAM-name PARAM-value [,PARAM-name PARAM-value]… ]
  ```

- :PPD (process-pair directory) displays the names, CPU and PIN designations, and ancestors of one or more processes in the destination-control table:

  ```
  :PPD [ [ \node. ] [$process-name cpu, pin ] ]
  ```

  process-name can specify a high PIN process or a low PIN process. *cpu*,*pin* can only specify a low PIN process.

- :PURGE deletes one or more disk files:

  ```
  :PURGE file-name [ [,] file-name ]…
  ```

- :RENAME changes the name of an existing disk file:

  ```
  :RENAME old-file-name [,] new-file-name
  ```

- :RUN runs a program:

  ```
  :[ RUN ] program-file-name [ /run-option [ ,run-option ]… / ]
  [ param-set ]
  ```

  run-option is a TACL RUN command option. For a list and descriptions of the options, see the RUN[D] command description in the *TACL Reference Manual*.

  param-set specifies program parameters for program-file-name.

- :SET DEFINE sets values for DEFINE attributes in the working-attributes set:

  ```
  :SET DEFINE { attribute-spec
  LIKE DEFINE-name } [ , attribute-spec ]…
  ```

  For a description of *attribute-spec*, see the description of the SET DEFINE command in the *TACL Reference Manual.*

- :STATUS displays information about one or more running processes:

  ```
  :STATUS [ range ] [ , condition ]… [ , DETAIL ]
  ```

  range is one of:

  ```
  [ \node. ] cpu,pin
  [ \node. ] cpu-number
  [ \node. ] $process-name
  [ \node. ] *
  ```

  condition is one of:

  ```
  GMOMJOBID $process-name.num
  PRI [ priority ]
  PROG [ program-file-name ]
  TERM [ $terminal-name ]
  USER [ group-name.user-name
  group-ID, user-ID ]
  ```

- :STOP terminates a process or process pair:

  ```
  :STOP [ [ \node. ] { $process-name cpu, pin,} ]
  ```

- :SUSPEND suspends a process or process pair:

  ```
  :SUSPEND [ [ \node. ] { $process-name cpu, pin,} ]
  ```

- :SYSTEM sets the current default node:

  ```
  :SYSTEM [ \node ]
  ```

- :TIME displays the current setting of the system date and time-of-day clock:

  ```
  :TIME
  ```

- :VOLUME sets the current default node, volume, and subvolume:

  ```
  :VOLUME [ [ \node. ] volume ] [ , " security " ]
  ```

- :WHO displays information about the current NBEXEC process:

Other Commands

- ! (exclamation point) causes NBEXEC to ignore the rest of the command line. (As an alternative to !, use :COMMENT.)

  ```
  ! [ comment-text ]
  ```

- * (asterisk) is the identifier of data that NBEXEC passes to a process it starts:

  ```
  * data
  ```

- . (period) is the identifier of a label marking the start of a section in a control file:

  ```
  . label
  ```

NBEXEC-Defined Logical Variables

- #ERROR indicates (TRUE or FALSE) whether the NBEXEC error state is set:

  ```
  #ERROR
  ```

- #NEXTAUDIT indicates (TRUE or FALSE) whether the file is audited by TMF:

  ```
  #NEXTAUDIT
  ```

- #NEXTFILENAME indicates (TRUE or FALSE) whether a file matches the search criteria:

  ```
  #NEXTFILENAME
  ```

- #NEXTLICENSE indicates (TRUE or FALSE) whether the file is licensed to run in privileged mode:

  ```
  #NEXTLICENSE
  ```

- #NEXTOPEN indicates (TRUE or FALSE) whether the file is open or has an incomplete TMF transaction against it:

  ```
  #NEXTOPEN
  ```

- #NEXTPROGID indicates (TRUE or FALSE) whether the file is secured by the PROGID option of the FUP SECURE command:

  ```
  #NEXTPROGID
  ```

- #NEXTROLLFORWARD indicates (TRUE or FALSE) whether the file has the TMF rollforward-needed flag set:

  ```
  #NEXTROLLFORWARD
  ```

NBEXEC-Defined String Variables

- #BP contains the text supplied in the last stop message received by NBEXEC from a waited process:

  ```
  #BP
  ```

- #BPID contains the NBEXEC process name:

  ```
  #BPID
  ```

- #CC contains the text representation of the completion code supplied in the last stop message received by NBEXEC from a waited process:

  ```
  #CC
  ```

- #HOME contains the name of the home terminal of the NBEXEC process:

  ```
  #HOME
  ```

- #JOBID contains the number of the current NBEXEC job:

  ```
  #JOBID
  ```

- #NEXTBLOCK contains the number of bytes in each block of records in the file:

  ```
  #NEXTBLOCK
  ```

- #NEXTCODE contains the file code:

  ```
  #NEXTCODE
  ```

- #NEXTDATE contains the date when the file was last modified:

  ```
  #NEXTDATE
  ```

- #NEXTEXTSIZE contains the primary and secondary extent sizes of the file:

  ```
  #NEXTEXTSIZE
  ```

- #NEXTNAME contains the fully qualified file name:

  ```
  #NEXTNAME
  ```

- #NEXTRECLEN contains the maximum number of bytes in a logical record:

  ```
  #NEXTRECLEN
  ```

- #NEXTSECURE contains the file security string:

  ```
  #NEXTSECURE
  ```

- #NEXTSIZE contains the current allocated file size in bytes:

  ```
  #NEXTSIZE
  ```

- #NEXTTIME contains the time when the file was last modified:

  ```
  #NEXTTIME
  ```

- #NEXTTYPE contains the file type U (unstructured), R (relative), E (entry-sequenced), or K (key-sequenced):

  ```
  #NEXTTYPE
  ```

- #NEXTUSER contains the user ID of the file owner:

  ```
  #NEXTUSER
  ```

- #PID contains the ID of the process started by the last :RUN command:

  ```
  #PID
  ```

- #TI contains the text representation of the termination information supplied in the last stop message received by NBEXEC from a waited process:

  ```
  #TI
  ```

- #TIME contains the current time plus or minus a specified number of minutes:

  ```
  #TIME [ { + - } minutes ]
  ```

- #TODAY contains the current date plus or minus a specified number of days:

  ```
  #TODAY [ { + - } days ]
  ```

C National Language Support

This appendix explains how to change BATCHCOM keywords and messages to suit your operational environment. To change the keywords and messages, use BATCHUTL, the NetBatch national-language-support program.

Changing BATCHCOM Keywords and Messages

Step 1: Log On as the Super ID

Log on as the super ID (255,255):

```
> LOGON SUPER.SUPER, password
```

Consideration

Logging on as the super ID lets you avoid security violations when completing the procedure.

Step 2: Make a Backup Copy of BATCHLIB

Make a backup copy of BATCHLIB, the NetBatch library file:

```
> FUP DUP $DATA7.TEST.BATCHLIB, $NB.BACKUPS.BATCHLIB
FILES DUPLICATED: 1
```

Consideration

Backing up BATCHLIB ensures you have a fallback copy of the file if needed.

Step 3: Extract Keywords and Messages From BATCHLIB

Run BATCHUTL with the BUILD-SYMBOLS parameter to extract keywords and messages from BATCHLIB and place them in an EDIT file:

```
> BATCHUTL /IN $DATA7.TEST.BATCHLIB, OUT EDITSRC/ BUILD-
SYMBOLS
1
```

Considerations

Run the BATCHUTL program to extract keywords and messages from BATCHLIB and to place them in an EDIT file:

```
[ RUN ] BATCHUTL / IN BATCHLIB-file , OUT EDIT-file /
B[UILD-SYMBOLS] [ n ]
```

BATCHLIB-file

specifies the BATCHLIB file from which you want to extract keywords and messages.

EDIT-file

specifies the name of the EDIT file BATCHUTL creates for the keywords and messages it extracts from *BATCHLIB-file*.

n

is a number in the range 0 through 6 identifying the keyword and message source. You use this number in the RUN BATCHUTL command at Step 5 when converting *EDIT-file* to a TAL source file. You also use the number in DEFINE =_ZBAT_NLS at Step 9 to specify the keywords and messages for future BATCHCOM sessions. The default is 0.

Step 4: Change Keywords and Messages in EDIT Source File

Make the required keyword and message changes in the EDIT file:

```
> EDIT EDITSRC
CURRENT FILE IS $DATA7.TRASH.EDITSRC
*FIX 654
654 2237 2Command not configured
.......... rdisabled by NBFLAGS procedure
654 2237 2Command disabled by NBFLAGS procedure
..........
*EXIT
```

Considerations

The BATCHUTL-created EDIT file from Step 3 contains six sections. The first section contains messages. The remaining sections contain keywords, tokens for days, months, and times, and a NetBatch vocabulary. You can change the items in these sections to suit your operational environment.

Supported Characters

BATCHCOM supports US ASCII characters in the ranges 64 through 93 and 96 through 125 (that is, characters from @ to] inclusive and from ` to } inclusive). Support for this character set enables you to specify keywords and messages in any language that uses characters from the set.

Message Format

The first *EDIT-file* section contains messages in the form:

```
[ message^literal = ] [ - ] number [ ± number ] ,
-- severity-indicator message-text [ & -- message-text ]…
```

`message^literal`

is a valid TAL literal. `message^literal` is for HP internal use only.

`number`

is a message number. *`number`* is for HP internal use only.

`severity-indicator`

is a message-severity indicator with a value of 0 (informational message), 1 (warning message), or 2 (error message). The default is 2. *`severity-indicator`* is for HP internal use only.

`message-text`

is user-modifiable text. *`message-text`* can contain substitutions and line-break characters. For details, see the description of message 0.

`&`

is a line-break character.

Message 0

The first message in the EDIT file is message 0:

```
0 .IWE~&.English Unexpected error
```

Message 0 is user modifiable and has special significance. The characters IWE~& specify message-severity (IWE), substitution (~), and line-break (&) characters used by all other messages. The first word (English) identifies the language used. The remaining words (Unexpected error) specify text for messages returned without text.

- BATCHCOM displays message-severity characters with message numbers and text. For example:

  ```
  0541-I No such line
  0515-W ALTER CLASS expects INITIATION ON or OFF
  2076-E CPU must be specified
  ```

- A substitution character followed by a number in message-text (for example, ~1, ~2, …) indicates a scheduler-supplied message-token value. (For a list of NetBatch message tokens, see the *NetBatch Management Programming Manual*.) BATCHCOM replaces ~1 with the first value received, ~2 with the second value received, and so on. For example, in message 2189 (2189 2Error~1, File:~2), the scheduler returns two tokens. The first contains a file-system error number, and the second contains a file name. The resulting BATCHCOM message appears as:

  ```
  2189-E Error file-system-error-number, File: file-name
  ```

When specifying a substitution in message-text, do not leave spaces between the substitution character, the substitution number, and the surrounding words.

- A line-break character in message-text makes BATCHCOM wrap text to the next line after the character.

Message Displays

BATCHCOM displays messages in the form:

```
message-number - severity-character message-text
```

message-number

is the message number.

severity-character

is the message-severity character as specified by message 0.

message-text

is the message text.

Keywords

The EDIT-file section beginning with =BATCHCOM contains BATCHCOM keywords; for example:

```
=BATCHCOM
<AT-AFTER-TIME> T^008 .
.
```

Days

The EDIT-file section beginning with =DAYS contains day tokens. For example:

```
=DAYS
FRIDAY T^005 .
.
```

Months

The EDIT-file section beginning with =MONTHS contains month tokens. For example:

```
=MONTHS
APRIL T^004 .
.
```

Times

The EDIT-file section beginning with =TIMES contains time tokens. For example:

```
=TIMES
AM T^014 .
.
```

Vocabulary

The EDIT-file section beginning with =VOCAB contains a NetBatch vocabulary. For example:

```
=VOCAB
ABORTED T^001 .
.
```

Editing Keywords, Days, Months, Times, and the Vocabulary

You can edit the names associated with various tokens in the EDIT file and include aliases for keywords. These considerations apply:

- A name must have a unique token number T^ *number*; for example, T^010.
- An alias can reference an existing token by appending A^128 to the alias; for example, ADD T^22 SUBMIT T^22 A^128 == Alias for ADD.
- In a section, a name must be unique and consist of US ASCII characters in the ranges 64 through 93 and 96 through 125 (that is, from @ to] inclusive and from ` to } inclusive).
- A name is terminated by any character outside the ranges specified in the preceding paragraph.
- Preferably, choose names that require the minimum number of characters to establish uniqueness. The name-recognition procedure requires at least the first character of a name and every hyphen character contained in the name. All other characters can be omitted when the name entered matches only one name in the section. Regardless of ambiguity, if an exact match is found, the name is valid. For example, O T^45 OBEY T^45 A^128 ON T^100 OFF T^101. (O is an inadequate abbreviation of OBEY, ON, and OFF, but because it is itself included as a keyword, it is accepted).
- Names must begin in the first column of the EDIT file.
- You can enter names in any order in a section. BATCHUTL sorts them alphabetically at Step 5.
- At least one space must separate the token number from the name and the alias flag from the token number.
- You can specify a single special character instead of a name. A special character is any printable character that is not a character in the range A through Z or 0 through 9.
- Names surrounded by angle brackets (for example, <AT-AFTER-TIME>) represent nonkeyword terminal symbols defined in a particular grammar. The names are for HP internal use only. You can leave them in the file or delete them.

Step 5: Convert EDIT Source File to TAL Source File

Run BATCHUTL with the COMPILE-SYMBOLS parameter to convert the EDIT file to a TAL source file:

```
> BATCHUTL /IN EDITSRC, OUT TALSRC/ COMPILE-SYMBOLS 1
```

Considerations

Run the BATCHUTL program to convert the modified EDIT file to a TAL source file:

```
[ RUN ] BATCHUTL / IN EDIT-file , OUT TAL-source-file /
C[OMPILE-SYMBOLS] [ n ]
```

EDIT-file

specifies the name of the EDIT file created at Step 3 and modified at Step 4.

TAL-source-file

specifies the name of the TAL source file. You compile this file into a TAL object at Step 6.

n

is the number specified for *n* at Step 3. The default is 0.

Step 6: Compile TAL Source File

Use the TAL compiler to compile the TAL source file created by BATCHUTL at Step 5 into a TAL object. You bind this object into BATCHLIB at Step 7.

```
> TAL /IN TALSRC/ TALOBJ; SUPPRESS
BINDER - OBJECT FILE BINDER …
Object file \MELBDEV.$DATA7.TRASH.TALOBJ .
.
```

Step 7: Bind Compiled Object Into BATCHLIB

Use the Binder program to bind the TAL object created at Step 6 into BATCHLIB:

```
> BIND
BINDER - OBJECT FILE BINDER …
@ADD * FROM $DATA7.TEST.BATCHLIB
@REPLACE * FROM $DATA7.TRASH.TALOBJ
@BUILD $DATA7.TEST.BATCHLIB !
.
.
@EXIT
```

Considerations

You can bind up to seven such objects into BATCHLIB (each object specifying a set of keywords and messages identified by the BATCHUTL run parameter *n*). This example shows three objects being bound into BATCHLIB:

```
> BATCHUTL /IN BATCHLIB, OUT EDITSRC1/ BUILD-SYMBOLS 1
> BATCHUTL /IN BATCHLIB, OUT EDITSRC2/ BUILD-SYMBOLS 2
> BATCHUTL /IN BATCHLIB, OUT EDITSRC3/ BUILD-SYMBOLS 3
> EDIT EDITSRC1 …
> EDIT EDITSRC2 …
> EDIT EDITSRC3 …
> BATCHUTL /IN EDITSRC1, OUT TALSRC1/ COMPILE-SYMBOLS 1
> BATCHUTL /IN EDITSRC2, OUT TALSRC2/ COMPILE-SYMBOLS 2
> BATCHUTL /IN EDITSRC3, OUT TALSRC3/ COMPILE-SYMBOLS 3
> TAL /IN TALSRC1/ TALOBJ1; SUPPRESS
> TAL /IN TALSRC2/ TALOBJ2; SUPPRESS
> TAL /IN TALSRC3/ TALOBJ3; SUPPRESS
> BIND
BINDER - OBJECT FILE BINDER …
@ADD * FROM BATCHLIB
@REPLACE * FROM TALOBJ1
@REPLACE * FROM TALOBJ2
@REPLACE * FROM TALOBJ3
@BUILD BATCHLIB !
@EXIT .
.
```

Step 8: Assign Updated BATCHLIB File to BATCHCOM

Assign the updated BATCHLIB to BATCHCOM by using the LIB option of the TACL RUN command:

```
> BATCHCOM /LIB $DATA7.TEST.BATCHLIB/; EXIT
```

Consideration

Use the LIB option of the TACL RUN command to assign the BATCHLIB file modified at Step 7 to BATCHCOM.

Step 9: Add DEFINE =_ZBAT_NLS to the TACL Environment

Add DEFINE =_ZBAT_NLS to the TACL environment to specify the modified keywords and messages for future BATCHCOM sessions:

```
> ADD DEFINE =_ZBAT_NLS, CLASS MAP, FILE L1.MDY
```

Considerations

BATCHCOM can use only one set of keywords and messages at a time. To select the keyword and message set you want to use for future BATCHCOM sessions, add

DEFINE =_ZBAT_NLS to the TACL environment by using this ADD DEFINE command:

```
ADD DEFINE =_ZBAT_NLS, CLASS MAP, FILE L n.[ date-format ]
```

n

is the number identifying the keyword and message set you want to use. You specified this number for the set at Steps 3 and 5.

date-format

specifies your preferred date format for command input and output. The options are DMY (for day followed by month and year), MDY (for month followed by day and year), and YMD (for year followed by month and day). The default is DMY.

Glossary

access privileges. Screen and function usage rights assigned by the system administrator to NetBatch-Plus users on a series of three linked screens: Security Supervise, Screen Security, and Utility Security.

ad hoc job. A job selected from a list of job names displayed on the Ad Hoc Job Selection screen. You can submit the selected job for execution from either the Ad Hoc Job Selection screen or Job Definition screen. You can also create a temporary copy of the selected job and then submit that copy as a one-off job. See also one-off job.

application. A set of programs designed to perform a specific task, usually involving specified operations on a database. For example, the NetBatch-Plus application consists of programs allowing you to set up and control NetBatch schedulers and their objects. You can also use the NetBatch-Plus programs to define and store job descriptions in the NetBatch-Plus database and submit, in a single operation, many related jobs for execution by NetBatch schedulers.

ASSIGN. A parameter that assigns the name of an actual file to a logical file name in a program and optionally specifies the creation attributes of the file.

attachments. A collective name for catalog and job ASSIGNs, PARAMs, and DEFINEs.

BATCHCOM. The program name of the NetBatch command interpreter.

BPROC. The former name of NBEXEC, the NetBatch executor program. See NBEXEC.

bulk job selection criteria. The criteria by which the bulk submit program selects a job for inclusion in a bulk submit run.

bulk submit. A NetBatch-Plus job selection and submission method enabling you to select and submit up to 2500 jobs at a time to NetBatch schedulers.

bulk submit control job. A job that submits the jobs in a bulk submit run to their respective schedulers. You specify control job parameters on the Bulk Submit Environment screen.

bulk submit environment. An environment that specifies the NetBatch scheduler and the class, owner, and default OUT attribute for the bulk submit control job. Also specifies the time interval that determines the retention period for temporary output files created during bulk submit runs. This interval also determines the number of days before execution that you can schedule those runs.

calendar category. A set of bulk submit selection dates identified by a unique name.

catalog attachment. An ASSIGN, PARAM, or DEFINE that several jobs in the NetBatch-Plus database can share. You define catalog attachments on the Catalog Attachments screens.

catalog DEFINEs. See DEFINE.

category. See calendar category and selection category.

class. A logical entity in a NetBatch scheduler. Classes control the flow of jobs to executors and therefore to those executors' CPUs. You can use classes to group jobs according to their demand for system resources. For example, you could group CPU-bound jobs such as program compilations into one class, I/O-bound jobs such as reports into another class, and so on. You can assign classes to multiple executors, the order in which you do so determines the order in which the executors process them.

completion code. A status code returned by a process to its creator. The code indicates whether the process terminated successfully or otherwise.

concurrent job. A job executing at the same time as another job. A NetBatch scheduler can execute up to 64 concurrent jobs.

control job. See bulk submit control job.

CPU. Central processing unit.

daily production bulk submit run. A production bulk submit run in which the bulk submit program runs automatically at the same time each day. See also production bulk submit run.

defaults DEFINEs. See DEFINE.

defaults set. A set of job attributes owned by a NonStop system user and recorded by NetBatch-Plus in its database. Jobs in the database can share a common defaults set and adopt their attributes from that set unless otherwise specified.

DEFINE. A named set of attributes and associated values. In a DEFINE (as with an ASSIGN), you can specify information that jobs communicate to processes they start. The NetBatch-Plus application supports five types of DEFINEs:

- Defaults DEFINEs hold the standard default values of a process such as the default volume.
- Map DEFINEs redirect or substitute files. You can enter the logical name of a map DEFINE in place of a physical file name in a command or procedure call.
- Spool DEFINEs pass information to the spooler collector process. The attributes of a spool DEFINE specify parameters such as the spooler location and batch name.
- SQL catalog DEFINEs specify the locations of NonStop SQL/MP catalogs. You can enter the logical name of the catalog DEFINE in place of a catalog name in CATALOG clauses in NonStop SQL/MP data manipulation language (DML) statements.

- Tape DEFINEs pass information to the tape process during labeled-tape operations. Tape DEFINE attributes specify parameters such as the tape device name and the record format.

dependency. A relationship between two jobs that prevents one of the jobs (the dependent job) from executing before the other job (the master job) releases it.

dependent job. A job with the WAITON attribute. Dependent jobs do not execute until released by all the jobs specified by the attribute.

EBCDIC. Extended Binary Coded Decimal Interchange Code. One of the codes used to represent characters in computers.

executor. A logical entity in a NetBatch scheduler. Executors link jobs, via their classes, to CPUs. This link enables the scheduler to execute, in the specified CPU, the initial process (the executor program) of each job. The number of active executors in a scheduler determines the number of jobs that can run concurrently. For example, a scheduler with 10 active executors can run up to 10 jobs at the same time. Similarly, for a scheduler to run the maximum possible number of concurrent jobs (64), it would need 64 active executors.

executor program. An object file started as a process by a NetBatch scheduler. The object file can be a command interpreter (for example, TACL) or a compiler (for example, COBOL). It can also be a query language and report formatter (for example, ENFORM), a utility (for example, FUP), or a user program.

IMMU. Informational Message Management Utilities. A product used for handling text such as message text, help text, and keyword substitution. It consists of a utility that loads and unloads key-sequenced message tables and the procedures needed to access those tables. NetBatch-Plus uses the IMMU product for handling help text.

input file. A file containing information an executor program needs to execute a job. For example, if a TACL process is the executor program, the file contains TACL commands. If the COBOL compiler is the executor program, the file contains the program source. The syntax of the information in the file must comply with the syntax rules of the executor program. In a traditional batch processing environment, the job input file is known as a job control file.

JCF. Job control file. See input file.

job. A logical entity in a NetBatch scheduler. A job's attributes specify information the scheduler uses to start an executor program as a NonStop system process. This information includes:

- The name of the executor program, its execution priority, input and output files, and startup parameters
- Details of the job's class and selection priority within that class
- Details of the job's dependencies

- Resource requirements such as the number of tape drives the job needs
- Run information such as timing details, hold flags, and the action to be taken in the event of job failure

You can record job descriptions in the NetBatch-Plus database. You can then select jobs from this database at any time and submit them for execution by a NetBatch scheduler.

job attachment. An ASSIGN, PARAM, or DEFINE used by a job. You define job attachments on the Job Attachments screens.

job control file. Job control file. See input file.

logon defaults. The system, volume, subvolume, and file security values in effect when you logged on to the system. For more information about logon defaults, current defaults, file-name expansion, and qualifying file names, see the *Guardian User's Guide*.

map DEFINEs. See DEFINE.

master job. A job on which another job (the dependent job) depends. Master jobs are the jobs specified by the WAITON attributes of their dependent jobs.

NBEXEC. The program name of the NetBatch executor program. NBEXEC superseded BPROC as the name of this program.

NetBatch. A job management system used to submit, schedule, execute, and control batch jobs.

NetBatch-Plus. A Pathway application that provides a screen-driven interface to the NetBatch job management system. You can use the application, which has its own database, to control NetBatch systems running on different nodes.

NetBatch-Plus application. See application.

NetBatch-Plus database. A database containing descriptions of jobs that NetBatch-Plus users can select and submit for execution by a NetBatch scheduler. Job descriptions can include information about dependencies, attachments, and bulk job selection criteria. In addition to job descriptions, the database records details of NetBatch-Plus users and configuration information for the bulk submit program.

Next Page key. The key you press on your keyboard to display the next 24 lines of help text. The actual name of the key depends on the keyboard you are using. For example, the Next Page key on some keyboards is Next. On others it is Page Down or PgDn.

nonrecurrent job. A job that does not have the CALENDAR or EVERY attribute. The scheduler deletes nonrecurrent jobs from its database after they finish executing.

one-off job. A job submitted from within the NetBatch-Plus application to a NetBatch scheduler and whose details are not recorded in the NetBatch-Plus database. You can create and submit one-off jobs by using the Job Info screen or Ad Hoc Job Selection screen.

PARAM. A parameter supplying a user-defined value to a process requesting that value at creation time.

Prev Page key. The key you press on your keyboard to display the previous 24 lines of help text. The actual name of the key depends on the keyboard you are using. For example, the Previous Page key on some keyboards is Prev. On others it is Page Up or PgUp.

production bulk submit run. A bulk submit run in which the bulk submit program selects jobs and submits them to their NetBatch schedulers. The Bulk Submit Submissions report, which the program produces automatically for the run, lists the jobs submitted in the run.

recurrent job. A job that has the CALENDAR or EVERY attribute. The scheduler automatically executes recurrent jobs on the dates or at the interval specified by the attribute.

scheduler. A fault-tolerant, process-pair server controlling the execution of batch jobs. Scheduler functions include:

- Scheduling jobs for execution
- Distributing the processing load across different CPUs
- Monitoring the processing states of jobs
- Restarting failed jobs (if specified)
- Managing job dependencies
- Maintaining job and configuration information

A scheduler can execute up to 64 jobs at the same time. Jobs executing at the same time are called concurrent jobs.

scheduler database. A database containing scheduler configuration information that includes details of the scheduler's executors and classes. Each scheduler has its own database. The database also contains jobs submitted to the scheduler for execution. For systems running G-series RVUs, the scheduler can store up to 32767 jobs, whereas for systems running H-series RVUs, it can store up to 9999 jobs. The scheduler deletes jobs from its database after they finish executing unless the jobs are recurrent (that is, they have an attribute specifying automatic rescheduling).

selection category. A means of selecting jobs from the NetBatch-Plus database for inclusion in a bulk submit run. A selection category can be a calendar category or simply a name used as a means of grouping jobs for selection purposes.

selection date. A date used by the bulk submit program to select a job from the NetBatch-Plus database.

session. The period of time from when you start and sign on to the NetBatch-Plus application to when you sign off.

spool DEFINEs. See DEFINE.

Software Product Revision The generic term for a distributed software product object. An SPR is distributed in the form of a Product Version Update (PVU) within a Product Version.SPRs have access, delivery, and sequence attributes.

software product revision (SPR). The generic term for a distributed software product object. An SPR is distributed in the form of a Product Version Update (PVU) within a Product Version.SPRs have access, delivery, and sequence attributes.

SQL catalog DEFINEs. See DEFINE.

TACL. HP Tandem Advanced Command Language, the user interface to the HP NonStop™ operating system. The TACL product is both a command interpreter and a command language. Users can write TACL programs that perfrorm complex tasks or provide a consistent user interface across independently programmed applications.

tape DEFINEs. See DEFINE.

test bulk submit run. A trial bulk submit run in which the bulk submit program selects jobs but does not submit them to their NetBatch schedulers. The Bulk Submit Predictions report, which the program produces automatically for the run, lists the jobs selected in the run. By running a test, you can determine which jobs are selected in a production run before you execute that run.

wild-card process. A wild-card process is a NetBatch scheduler process or spooler supervisor process specified using the asterisk (*), question mark (?), or both wild-card characters. When the NetBatch-Plus application searches for process names matching the specification, it searches on the wild-card processes list, which you maintain on the Wild-Card Processes screen. The list is a table of fully qualified process names enabling the application to perform quick searches for matching processes rather than time-consuming searches on all systems in your network.

Index

A

Abbreviations, of keywords 6-9
ABORT SCHEDULER command
  description 6-30
  synopsis and syntax summary 6-20
ACTIVATE JOB command
  description 6-32
  synopsis and syntax summary 6-15
ACTIVE executor state
  CPU failure, effect of 7-34
  description 3-31, 6-164
  scheduler warm start, effect of 3-22
  STOP EXECUTOR command, effect of 6-178
ADD ATTACHMENT-SET command
  description 6-34
  synopsis and syntax summary 6-12
ADD CLASS command
  description 6-40
  synopsis and syntax summary 6-13
ADD EXECUTOR command
  description 6-42
  synopsis and syntax summary 6-14
ADD JOB
  See SUBMIT JOB command
ADD SCHEDULER command
  description 6-46
  synopsis and syntax summary 6-20
ADD, alias of SUBMIT 6-11
ADD-ALLOWED
  See SUBMIT-ALLOWED scheduler attribute
ADP (ASSIGNs, DEFINEs, and PARAMs) alias of ATTACHMENT-SET 6-11
AFTER job attribute
  changes AT to AFTER 7-11, 7-17
  description 7-9
  synopsis and syntax summary 7-2
Algorithm (scheduling), description 1-9
Aliases, of keywords and commands 6-11
ALLOW ERRORS command
  description 6-51
  synopsis and syntax summary 6-22
ALTER ATTACHMENT-SET command
  description 6-52
  synopsis and syntax summary 6-12
ALTER CLASS command
  description 6-58
  synopsis and syntax summary 6-13
ALTER EXECUTOR command
  description 6-59
  synopsis and syntax summary 6-14
ALTER JOB command
  description 6-62
  synopsis and syntax summary 6-15
ALTER SCHEDULER command
  description 6-67
  synopsis and syntax summary 6-20
Ampersand (&), line continuation character 6-76
ASSIGN attachment-set attribute
  description 7-12
  synopsis and syntax summary 7-1
ASSIGNs, DEFINEs, and PARAMs (ADP) 6-11
ASSUME ATTACHMENT-SET command
  description 6-69
  synopsis and syntax summary 6-12
ASSUME CLASS command
  description 6-70
  synopsis and syntax summary 6-13
ASSUME EXECUTOR command
  description 6-71
  synopsis and syntax summary 6-14

ASSUME JOB command
description 6-72
synopsis and syntax summary 6-16
ASSUME SCHEDULER command
description 6-73
synopsis and syntax summary 6-20
Asterisk (*) wild-card character
See Wild-card characters
AT job attribute
description 7-15
synopsis and syntax summary 7-2
ATTACH scheduler database file 6-48
ATTACH0 scheduler database file 6-48
Attachment set
adding 1-9, 4-35, 6-34
altering
description 1-9, 4-37, 6-52
See also Attachment set, deleting
attribute default values 6-35
attribute descriptions
ASSIGN 7-12
DEFINE 7-47
PARAM 7-89
SECURITY 7-99
TEMPORARY 7-112
command descriptions
ADD ATTACHMENT-SET 6-34
ALTER ATTACHMENT-SET 6-52
ASSUME ATTACHMENT-SET 6-69
DELETE ATTACHMENT-SET 6-77
INFO ATTACHMENT-SET 6-99
RESET ATTACHMENT-SET 6-116
SHOW ATTACHMENT-SET 6-142
STATUS ATTACHMENT-SET 6-160
CURRENT variable
displaying value of 6-100
how maintained 7-20
setting null value 6-114
using 4-38
Attachment set (continued)
definition 1-3, 1-9
deleting
by DELETE ATTACHMENT-SET command 6-77
by scheduler warm start when TEMPORARY ON 3-22
description 1-9, 4-38
See also Attachment set, altering
displaying 4-36, 6-99, 6-160
named, definition 7-113
names, makeup of 6-35
numbered, definition 7-113
passing of ASSIGNs, DEFINEs, and PARAMs to executor-program process
ASSIGNs 7-12
DEFINEs 7-47
PARAMs 7-89
securing 7-99
See also ATTACHMENT-SET job attribute
specifying for a job 4-34
ATTACHMENT-SET job attribute
description 7-19
synopsis and syntax summary 7-2
Attribute descriptions
AFTER 7-9
ASSIGN 7-12
AT 7-15
ATTACHMENT-SET 7-19
AT-ALLOWED 7-18
BACKUPCPU 7-23
CALENDAR 7-25
CATCHUP 7-28
CLASS (executor) 7-30
CLASS (job) 7-32
CPU 7-34
DEFAULT-CLASS 7-36
DEFAULT-EXECUTOR-PROGRAM 7-38

Attribute descriptions (continued)
DEFAULT-HIGHPIN 7-39
DEFAULT-MAXPRINTLINES 7-40
DEFAULT-MAXPRINTPAGES 7-41
DEFAULT-OUT 7-42
DEFAULT-PRI 7-43
DEFAULT-SELPRI 7-44
DEFAULT-STALL 7-45
DEFAULT-STOP-ON-ABEND 7-46
DEFINE 7-47
DESCRIPTION 7-52
EMS 7-54
EVERY 7-55
EXECUTOR-PROGRAM 7-58
EXTSWAP 7-60
HIGHPIN 7-61
HOLD 7-62
HOLDAFTER 7-63
IFFAILS 7-64
IN 7-66
INITIATION (class) 7-68
INITIATION (scheduler) 7-69
JOBID-ZERO 7-73
JOB-LOG 7-70
LIMIT 7-75
LOCALNAMES 7-77
MAXPRINTLINES 7-82
MAXPRINTPAGES 7-83
MAX-CONCURRENT-JOBS 7-79
MAX-PRI 7-81
MEM 7-85
NAME 7-86
OUT 7-87
PARAM 7-89
PFS 7-91
PRI 7-92
PURGE-IN-FILE 7-93
RESTART 7-95
RUND 7-97

Attribute descriptions (continued)
SAVEABEND 7-98
SECURITY 7-99
SELPRI 7-101
STALL 7-102
STARTUP 7-104
STOP-ON-ABEND 7-105
SUBMIT-ALLOWED 7-107
SWAP 7-108
TAPEDRIVES (job) 7-109
TAPEDRIVES (scheduler) 7-110
TEMPORARY 7-112
TERM 7-114
VOLUME 7-116
WAIT 7-118
WAITON 7-119
Attribute flag keywords
abbreviations 6-9
aliases 6-11
list 6-9
See also Attribute keywords
Attribute keywords
abbreviations 6-9
aliases 6-11
list 6-9
restrictions on use
in ASSIGN names 7-12
in DEFINE names 7-47
in PARAM names 7-89
See also Attribute flag keywords
Attributes, security (RWEP)
for SECURITY attribute 7-99
for VOLUME attribute 7-116
for VOLUME command 6-197
AT-ALLOWED scheduler attribute
description 7-18
synopsis and syntax summary 7-6

A, N, G, C, O, U, - (security codes) descriptions
for SECURITY attribute 7-99
for VOLUME attribute 7-116
for VOLUME command 6-197

B

BACKUPCPU scheduler attribute
description 7-23
synopsis and syntax summary 7-6
BATCHCAL
command descriptions 5-13
definition 1-4
high PIN capabilities 1-12, 5-4
BATCHCOM
attributes
descriptions 7-8
synopses and syntax summaries 7-1
changing keywords and messages C-1
commands
descriptions 6-28
security of 6-23
synopses and syntax summaries 6-11
completion codes returned by 6-51
conversational session, definition 6-1
definition 1-4
help
HELP command 6-95
text file 6-3
high PIN capabilities 1-12, 6-6
interactive session, definition 6-1
keywords
See Keywords
noninteractive session, definition 6-1
running 6-1
session
default scheduler name ($ZBAT) 6-2
session (continued)
definition 6-1
starting 6-1
stopping 6-92
starting 6-1
stopping 6-92
syntax checking, of commands 6-2
BATCHCTL scheduler database file
contents 3-16
created and opened during cold start 3-16
description 6-48
opened during warm start 3-19
See also Scheduler, cold start and Scheduler, warm start
BATCHIMU, help text file 6-3
BATCHLIB, defines NB^JOB^SUBMIT 1-5
BATCHUTL, national-language-support program C-1
BPROC
See NBEXEC

C

CALENDAR job attribute
description 7-25
synopsis and syntax summary 7-2
Catalog DEFINE
See DEFINE, catalog DEFINE
CATCHUP scheduler attribute
description 7-28
synopsis and syntax summary 7-6
CHANGEUSER command
description 6-74
synopsis and syntax summary 6-22
CHKQUE scheduler database file 6-48
CHKQUE0 scheduler database file 6-48
Class
adding 1-6, 6-40
altering 1-6, 3-34, 6-58
assigning to executor 3-6

Class (continued)
command descriptions
ADD CLASS 6-40
ALTER CLASS 6-58
ASSUME CLASS 6-70
DELETE CLASS 6-81
INFO CLASS 6-103
RESET CLASS 6-119
SET CLASS 6-133
SHOW CLASS 6-145
configuration 3-6
definition 3-6
deleting 1-6, 3-37, 6-81
displaying 3-31, 6-103, 6-145
CLASS executor attribute
description 7-30
synopsis and syntax summary 7-2
CLASS job attribute
description 7-32
synopsis and syntax summary 7-2
CLASS keyword, alias of JOBCLASS 6-11
Command descriptions
ABORT SCHEDULER 6-30
ACTIVATE JOB 6-32
ADD ATTACHMENT-SET 6-34
ADD CLASS 6-40
ADD EXECUTOR 6-42
ADD SCHEDULER 6-46
ALLOW ERRORS 6-51
ALTER ATTACHMENT-SET 6-52
ALTER CLASS 6-58
ALTER EXECUTOR 6-59
ALTER JOB 6-62
ALTER SCHEDULER 6-67
ASSUME ATTACHMENT-SET 6-69
ASSUME CLASS 6-70
ASSUME EXECUTOR 6-71
ASSUME JOB 6-72
ASSUME SCHEDULER 6-73
Command descriptions (continued)
CHANGEUSER 6-74
COMMENT 6-76
DELETE ATTACHMENT-SET 6-77
DELETE CLASS 6-81
DELETE EXECUTOR 6-82
DELETE JOB 6-84
DISPLAY-SPI 6-86
EXIT
BATCHCAL 5-13
BATCHCOM 6-92
FC
BATCHCAL 5-13
BATCHCOM 6-92
HELP
BATCHCAL 5-14
BATCHCOM 6-95
HISTORY 6-98
INFO ATTACHMENT-SET 6-99
INFO CLASS 6-103
INFO EXECUTOR 6-104
INFO JOB 6-106
INFO SCHEDULER 6-110
OBEY 6-112
OPEN 6-113
RELEASE-WAITON 6-115
RESET ATTACHMENT-SET 6-116
RESET CLASS 6-119
RESET EXECUTOR 6-120
RESET JOB 6-121
RESET SCHEDULER 6-122
RUN 6-124
RUNNEXT JOB 6-125
RUNNOW JOB 6-128
SET ATTACHMENT-SET 6-130
SET CLASS 6-133
SET EXECUTOR 6-134
SET JOB 6-136
SET SCHEDULER 6-140

Command descriptions (continued)
SHOW ATTACHMENT-SET 6-142
SHOW CLASS 6-145
SHOW EXECUTOR 6-146
SHOW JOB 6-147
SHOW SCHEDULER 6-150
SHUTDOWN SCHEDULER 6-152
START EXECUTOR 6-155
START SCHEDULER 6-157
STATUS ATTACHMENT-SET 6-160
STATUS EXECUTOR 6-163
STATUS JOB 6-165
STATUS SCHEDULER 6-173
STATUS-HISTORY 6-175
STOP EXECUTOR 6-178
STOP JOB 6-180
SUBMIT JOB 6-183
SUSPEND JOB 6-190
SWITCHCPU SCHEDULER 6-192
SWITCHLOG SCHEDULER 6-193
SYSTEM 6-195
VOLUME 6-196
! (exclamation point) 6-198
== (double equals)
BATCHCAL 5-13
BATCHCOM 6-76
? (question mark) 6-200
Commands, date formats in 7-9, 7-15
Commands, security of 6-23
Commands, time formats in 7-10, 7-15
COMMENT command
alias of (==) 6-11
description 6-76
synopsis and syntax summary 6-22
Components (core) of NetBatch product 1-4
Continuation of lines 6-76
Conversational session, definition 6-1
CPU executor attribute
description 7-34
synopsis and syntax summary 7-2
CPU-bound, definition 3-1
CTRL/Y, end BATCHCOM session 6-92

D

Date
displayed by STATUS SCHEDULER command 6-174
format in commands 7-9
Daylight-saving time (DST), effect on jobs of DST transitions 6-187
DEFAULT-CLASS scheduler attribute
description 7-36
synopsis and syntax summary 7-6
DEFAULT-EXECUTOR-PROGRAM scheduler attribute
description 7-38
synopsis and syntax summary 7-6
DEFAULT-HIGHPIN scheduler attribute
description 7-39
synopsis and syntax summary 7-6
DEFAULT-MAXPRINTLINES scheduler attribute
description 7-40
synopsis and syntax summary 7-6
DEFAULT-MAXPRINTPAGES scheduler attribute
description 7-41
synopsis and syntax summary 7-6
DEFAULT-OUT scheduler attribute
description 7-42
synopsis and syntax summary 7-6
DEFAULT-PRI scheduler attribute
description 7-43
synopsis and syntax summary 7-6
DEFAULT-SELPRI scheduler attribute
description 7-44
synopsis and syntax summary 7-7

DEFAULT-STALL scheduler attribute
description 7-45
synopsis and syntax summary 7-7
DEFAULT-STOP-ON-ABEND scheduler attribute
description 7-46
synopsis and syntax summary 7-7
DEFINE
attributes 7-48
catalog DEFINE
attributes 7-48
definition 7-48
default class 7-47
defaults DEFINE
attributes 7-48
definition 7-48
definition 7-47
map DEFINE
attributes 7-48
definition 7-48
search DEFINE
attributes 7-48
definition 7-48
sort DEFINE
attributes 7-49
definition 7-48
spool DEFINE
attributes 7-50
definition 7-48
subsort DEFINE
attributes 7-50
definition 7-48
tape DEFINE
attributes 7-50
definition 7-48
DEFINE attachment-set attribute
description 7-47
synopsis and syntax summary 7-1
DEFINE errors A-1
DELETE ATTACHMENT-SET command
description 6-77
synopsis and syntax summary 6-12
DELETE CLASS command
description 6-81
synopsis and syntax summary 6-13
DELETE EXECUTOR command
description 6-82
synopsis and syntax summary 6-14
DELETE executor state
description 3-31, 6-164
scheduler warm start, effect of 3-22
DELETE JOB command
description 6-84
synopsis and syntax summary 6-16
DESCRIPTION job attribute
description 7-52
synopsis and syntax summary 7-2
DETAIL qualifier, use in commands
DELETE ATTACHMENT-SET 6-78
INFO ATTACHMENT-SET 6-100
STATUS JOB 6-167
DISPLAY-SPI command
description 6-86
synopsis and syntax summary 6-22
Double braces (}}) prompt
in COMMENT command 6-76
in FC command 6-93
Double equals (==)
See == (double equals) 6-11
DOWN executor state
description 3-31, 6-164
scheduler warm start, effect of 3-22
DSM/SCM installation program 2-1

E

EMS parameter, in RUN NETBATCH command 3-12, 3-44, 7-54
EMS scheduler attribute
  description 7-54
  synopsis and syntax summary 7-7
End-of-file condition
  See CTRL/Y 6-92
EOF
  See CTRL/Y
Equals, double (==)
  See == (double equals)
Error 11, on scheduler warm start 6-177
Error 12, on scheduler warm start 3-21
Error 201, on scheduler warm start 3-21
Error messages A-1
EVENT job state, description 4-42, 6-169
Event-message generation, enabling and disabling 3-12, 3-44, 7-54
EVERY job attribute
  description 7-55
  synopsis and syntax summary 7-2
EXECUT00 scheduler database file 6-48
EXECUTING job state, description 4-42, 6-169
Executor
  adding 1-7, 6-42
  altering 1-7, 6-59
  assigning classes 6-60
  attribute descriptions
    CLASS 7-30
    CPU 7-34
  attributes, default values 6-43
  class selection priority 6-43
  command descriptions
    ADD EXECUTOR 6-42
    ALTER EXECUTOR 6-59
    ASSUME EXECUTOR 6-71
    DELETE EXECUTOR 6-82
    INFO EXECUTOR 6-104
  command descriptions (continued)
    RESET EXECUTOR 6-120
    SET EXECUTOR 6-134
    SHOW EXECUTOR 6-146
    START EXECUTOR 6-155
    STATUS EXECUTOR 6-163
    STOP EXECUTOR 6-178
  deleting 1-7
  starting 1-7
  stopping 1-7
  temporary
    creation of 7-15
    names, makeup of 7-16
Executor program, input file 1-1
EXECUTOR scheduler database file 6-48
EXECUTOR-PROGRAM job attribute
  description 7-58
  synopsis and syntax summary 7-3
EXIT command
  in BATCHCOM
    description 6-92
    synopsis and syntax summary 6-22
EXTSWAP job attribute
  description 7-60
  synopsis and syntax summary 7-3

F

FC command
  in BATCHCAL 5-13
  in BATCHCOM
    description 6-92
    synopsis and syntax summary 6-22
  subcommands 6-93
File-system errors A-1

G

GMOM
- definition 1-9
- See also GMOMJOBID

GMOMJOBID
- definition 1-9, 4-10
- displaying 4-10
- dissociating a job from the scheduler 4-12
- dissociating a process from a job 4-11
- process identification, tracking, and control, use in 1-9, 4-10
- See also JOBID-ZERO job attribute
- specifying TERM, IN, OUT, and NOWAIT for a dissociated process 4-12

Guidelines, planning
- See Planning guidelines

H

HELP command
- in BATCHCAL 5-14
- in BATCHCOM
  - description 6-95
  - help text file 6-3
  - synopsis and syntax summary 6-22

High PIN
- BATCHCAL processes, capabilities of 1-12, 5-4
- BATCHCOM processes, capabilities of 1-12, 6-6
- NBEXEC processes, capabilities of 1-12, B-2
- NETBATCH processes, capabilities of 1-12, 3-13

HIGHPIN job attribute
- description 7-61
- synopsis and syntax summary 7-3

HIGHPIN run option, in TACL RUN command
- BATCHCAL 5-4
- BATCHCOM 6-6
- NETBATCH 3-13

History buffer
- commands not stored in 6-75, 6-93, 6-98, 6-199, 6-201
- manipulating contents of 6-98
- See also HISTORY command
- size 6-98

HISTORY command
- description 6-98
- See also History buffer
- synopsis and syntax summary 6-23

HOLD job attribute
- description 7-62
- synopsis and syntax summary 7-3

HOLDAFTER job attribute
- description 7-63
- synopsis and syntax summary 7-3

Home terminal, of executor program 7-58, 7-114

I

IFFAILS job attribute
- description 7-64
- synopsis and syntax summary 7-3

Implicit command, ALLOW ERRORS 6-51

IN job attribute
- description 7-66
- synopsis and syntax summary 7-3

INFO ATTACHMENT-SET command
- description 6-99
- synopsis and syntax summary 6-12

INFO CLASS command
- description 6-103
- synopsis and syntax summary 6-14

INFO EXECUTOR command
description 6-104
synopsis and syntax summary 6-14
INFO JOB command
description 6-106
synopsis and syntax summary 6-16
INFO SCHEDULER command
description 6-110
synopsis and syntax summary 6-20
Information Message Management Utilities (IMMU), definition 6-3
INITIATION class attribute
description 7-68
synopsis and syntax summary 7-1
INITIATION scheduler attribute
description 7-69
synopsis and syntax summary 7-7
Input file
defaults, specifying 4-21
definition 4-7
purging 4-21
specifying 4-21
Interactive session, definition 6-1
I/O errors, sequential A-1
I/O-bound, definition 3-1

J

Job
activating 4-45, 6-32
altering 1-5, 1-8, 4-44, 6-62
attribute descriptions
AFTER 7-9
AT 7-15
ATTACHMENT-SET 7-19
CALENDAR 7-25
DESCRIPTION 7-52
EVERY 7-55
EXECUTOR-PROGRAM 7-58
EXTSWAP 7-60
HIGHPIN 7-61
attribute descriptions (continued)
HOLD 7-62
HOLDAFTER 7-63
IFFAILS 7-64
IN 7-66
JOBID-ZERO 7-73
JOB-LOG 7-70
LIMIT 7-75
MAXPRINTPAGES 7-83
MEM 7-85
NAME 7-86
OUT 7-87
PFS 7-91
PRI 7-92
PURGE-IN-FILE 7-93
RESTART 7-95
RUND 7-97
SAVEABEND 7-98
SELPRI 7-101
STALL 7-102
STARTUP 7-104
STOP-ON-ABEND 7-105
SWAP 7-108
TAPEDRIVES 7-109
TERM 7-114
VOLUME 7-116
WAIT 7-118
WAITON 7-119
attributes, default values 6-185
command descriptions
ACTIVATE JOB 6-32
ALTER JOB 6-62
ASSUME JOB 6-72
DELETE JOB 6-84
INFO JOB 6-106
RELEASE-WAITON 6-115
RESET JOB 6-121
RUNNEXT JOB 6-125
RUNNOW JOB 6-128

command descriptions (continued)
SET JOB 6-136
SHOW JOB 6-147
STATUS JOB 6-165
STATUS-EXECUTING 6-11, 6-166, 6-172
STATUS-HISTORY 6-175
STATUS-READY 6-11, 6-166, 6-172
STATUS-USER 6-11, 6-166, 6-172
STOP JOB 6-180
SUBMIT JOB 6-183
SUSPEND JOB 6-190
Job control language 1-2
JOB scheduler database file
description 6-49
renamed to JOBSAVED 3-21, 3-25
JOBCLAS0 scheduler database file 6-49
JOBCLASS scheduler database file 6-49
JOBCLASS, alias of CLASS 6-11
JOBID-ZERO job attribute
description 7-73
synopsis and syntax summary 7-3
JOB-LOG job attribute
description 7-70
synopsis and syntax summary 7-3

K

Keywords
abbreviations 6-9
aliases 6-11
list 6-7
restrictions on use
in ASSIGN names 7-12
in attachment-set names 6-35
in class names 6-40
in DEFINE names 7-47
in executor names 6-42
in job names 6-183
restrictions on use (continued)
in PARAM names 7-89
types 6-7

L

LIB job attribute
description 7-74
synopsis and syntax summary 7-3
LIKE qualifier, use in commands
ADD ATTACHMENT-SET 6-36, 7-47
ADD EXECUTOR 6-42
ALTER ATTACHMENT-SET 6-55, 7-47
ALTER EXECUTOR 6-59
ALTER JOB 6-62
SET ATTACHMENT-SET 6-131, 7-47
SET EXECUTOR 6-134
SET JOB 6-136
SUBMIT JOB 6-183
LIMIT job attribute
description 7-75
synopsis and syntax summary 7-3
Limits
NETBATCH
ASSIGN names, character in 7-12
attachment-set names, characters in 6-35
class names, characters in 6-40
classes per executor 6-43, 6-60
CPUs per executor 6-43, 6-60
DEFINE names, characters in 7-47
executor names, characters in 6-42
Line continuation character (&) 6-76
LOCALNAMES scheduler attribute
description 7-77
synopsis and syntax summary 7-7
Logon, during BATCHCOM session 6-74
Log-file chaining error, on scheduler warm start 6-177

Low PIN
definition 7-61
running executor-program processes at 7-61
See also High PIN

M

Macros
ZBAT JOBINFO 4-12
ZBAT RELEASE 4-32, 7-120
Management
job 4-42
scheduler 3-29
MAXPRINTLINES job attribute
description 7-82
synopsis and syntax summary 7-3
MAXPRINTPAGES job attribute
description 7-83
synopsis and syntax summary 7-4
MAX-CONCURRENT-JOBS scheduler attribute
description 7-79
synopsis and syntax summary 7-7
MAX-PRI scheduler attribute
description 7-81
synopsis and syntax summary 7-7
MEM job attribute
description 7-85
synopsis and syntax summary 7-4
Messages, descriptions A-1

N

NAME job attribute
description 7-86
synopsis and syntax summary 7-4
Named attachment set, definition 7-113
Names, makeup of
ASSIGN 7-12
attachment set 6-35
Names, makeup of (continued)
class 6-40
DEFINE 7-47
executor 6-42, 6-129, 6-164, 7-16
executor-program processes 7-58, 7-86
job 6-183
log files, scheduler 3-16, 3-19
PARAM 7-89
restrictions on use of attribute keywords
in ASSIGN names 7-12
in DEFINE names 7-47
in PARAM names 7-89
restrictions on use of object keywords
in attachment-set names 6-35
in class names 6-40
in executor names 6-42
in job names 6-183
restrictions on use of =_ZBAT... 6-3, 7-47
National-language support (NLS) C-1
NBATTX scheduler database file 6-49
NBATTX0 scheduler database file 6-49
NBEXEC
BPROC compatibility 1-4, B-1
commands, synopses and syntax summaries B-3
definition 1-4, B-1
high PIN capabilities 1-12, B-2
RUN NBEXEC command B-5
syntax summary B-5
user manual location B-1
variables, synopses, and syntax summaries B-10
NBEXEC errors A-1
NB-LOG qualifier, use in STATUS-HISTORY command 6-176
NB^JOB^SUBMIT 1-5

Nested OBEY commands
definition 6-112
maximum number 6-112
NetBatch product
ASSIGN, DEFINE, and PARAM propagation 1-3
command interpreter 1-4
CPU assignment 1-2
EMS interface 1-3
files
restoring to distribution subvolume 2-1
updating installation subvolumes 2-1
national-language support C-1
programmatic job submission and alteration 1-3
softdoc 2-2
spooler support 1-3
supervisor, definition 6-23
NETBATCH program
core component of NetBatch product 1-4
default log file size 3-20, 3-39, 6-194
definition of 1-4
EMS parameter 3-12, 3-44, 7-54
high PIN capabilities 1-12
limits
See Limits, NETBATCH
program file name, specifying in RUN command 3-10
NetBatch supervisor, definition 6-23
NetBatch-Plus
errors A-1
job submission, use in 3-12, 4-13, 7-78
scheduler opener 3-14, 7-80
Next run time
See Job, run time or Run calendar, run times
NLS (national-language support)
See National-language support (NLS)
Noninteractive session, definition 6-1
Nonrecurrent job, definition 7-25, 7-56
Numbered attachment set, definition 7-113
Numbers, job 6-187

O

OBEY command
description 6-112
nested 6-112
synopsis and syntax summary 6-23
OBEY-FORM qualifier, use in commands
INFO CLASS 6-103
INFO EXECUTOR 6-105
INFO JOB 6-107
INFO SCHEDULER 6-110
INO ATTACHMENT-SET 6-100
SHOW ATTACHMENT-SET 6-142
SHOW CLASS 6-145
SHOW EXECUTOR 6-146
SHOW JOB 6-148
SHOW SCHEDULER 6-151
Object attributes, descriptions 7-8
Object keywords
abbreviations 6-9
aliases 6-11
list 6-7
restrictions on use
in attachment-set names 6-35
in class names 6-40
in executor names 6-42
in job names 6-183
OFF
class state, description 6-103
executor state
CPU failure, effect of 7-34
description 3-31, 6-155, 6-164
scheduler warm start, effect of 3-22
START EXECUTOR command, effect of 6-155

executor state (continued)
STOP EXECUTOR command, effect of 6-178
OFF executor state
scheduler warm start, effect of 3-22
ON
class state, description 6-103
executor state
CPU failure, effect of 7-34
description 3-31, 6-164
scheduler warm start, effect of 3-22
START EXECUTOR command, effect of 6-155
STOP EXECUTOR command, effect of 6-178
ON executor state
scheduler warm start, effect of 3-22
Online help
See HELP command
OPEN command
description 6-113
synopsis and syntax summary 6-21
Openers (scheduler), maximum number of 3-14, 7-79
Options, run
for executor-program processes
See Run options, for executor-program processes
for TACL RUN command
See Run options, TACL RUN command
Order of classes and jobs
See Selection priority
OUT job attribute
description 7-87
in working-attributes set
resetting 6-121
setting 6-137
showing 6-147
See also DEFAULT-OUT scheduler attribute
OUT job attribute (continued)
synopsis and syntax summary 7-4
usage in commands
ALTER JOB 6-62
SET JOB 6-137
Output file, job
See Job, output file
OVER LIMIT job state, description 4-42, 6-169
Ownership of jobs 6-187

P

Pages, maximum number for job output file 7-83
PARAM
definition 7-89
See also Attachment set, or PARAM attachment-set attribute
PARAM attachment-set attribute
description 7-89
in working-attributes set
resetting 6-118
setting 6-131
showing 6-142
See also Attachment set, PARAM
synopsis and syntax summary 7-1
use in commands
ADD ATTACHMENT-SET 6-36
ALTER ATTACHMENT-SET 6-53
SET ATTACHMENT-SET 6-131
Parameter
See ASSIGN attachment-set attribute, or PARAM attachment-set attribute
Period (.) prompt, FC command 6-93
PFS job attribute
description 7-91
in working-attributes set
resetting 6-121
setting 6-136
showing 6-148

PFS job attribute (continued)
synopsis and syntax summary 7-4
use in commands
SET JOB 6-136
PIN
definition 6-6
See also High PIN or Low PIN
Planning guidelines
job 3-1, 4-1
scheduler 3-1
PMSEARCHLIST variable, function
when running BATCHCAL 5-3
when running NETBATCH 3-11
PRI job attribute
description 7-92
synopsis and syntax summary 7-4
Processes
high PIN capabilities of
BATCHCAL 1-12, 5-4
BATCHCOM 1-12, 6-6
executor program 7-61
NBEXEC 1-12, B-2
NETBATCH 1-12, 3-13
Prompts
}} (double braces)
in COMMENT command 6-76
in FC command 6-93
PURGE-IN-FILE job attribute
description 7-93
synopsis and syntax summary 7-4

Q

Qualified keywords, for commands
See Command qualifier keywords
Question mark (?)
command
See ? (question mark), command
wild-card character
See Wild-card characters

R

READY job state, description 4-43, 6-170
Recurrent job, definition 7-25, 7-56
RELEASE-WAITON command
description 6-115
synopsis and syntax summary 6-17
REPORT JOB Command 6-116
description of 6-116
synopsis and syntax summary 6-17
RESET ATTACHMENT-SET command
description 6-116
synopsis and syntax summary 6-13
RESET CLASS command
description 6-119
synopsis and syntax summary 6-14
RESET EXECUTOR command
description 6-120
synopsis and syntax summary 6-15
RESET JOB command
description 6-121
synopsis and syntax summary 6-17
RESET SCHEDULER command
description 6-122
synopsis and syntax summary 6-21
RESTART job attribute
description 7-95
synopsis and syntax summary 7-4
RUN command
description 6-124
synopsis and syntax summary 6-23
Run options
TACL RUN command
when running BATCHCOM 6-1
RUND job attribute
description 7-97
synopsis and syntax summary 7-4
RUNNEXT JOB command
description 6-125
synopsis and syntax summary 6-17

RUNNEXT job state, description 4-43, 6-170
RUNNOW JOB command
description 6-128
synopsis and syntax summary 6-17
RUNNOW job state, description 4-43, 6-170

S

SAVEABEND job attribute
description 7-98
synopsis and syntax summary 7-4
Scheduler
adding 3-9, 6-46
algorithm (scheduling), description 1-11
altering 3-31, 6-67
attachment set
See Attachment set
attribute default values 6-47
attribute descriptions
AT-ALLOWED 7-18
BACKUPCPU 7-23
CATCHUP 7-28
DEFAULT-CLASS 7-36
DEFAULT-EXECUTOR-PROGRAM 7-38
DEFAULT-HIGHPIN 7-39
DEFAULT-MAXPRINTLINES 7-40
DEFAULT-MAXPRINTPAGES 7-41
DEFAULT-OUT 7-42
DEFAULT-PRI 7-43
DEFAULT-SELPRI 7-44
DEFAULT-STALL 7-45
DEFAULT-STOP-ON-ABEND 7-46
EMS 7-54
INITIATION 7-69
LOCALNAMES 7-77
MAX-CONCURRENT-JOBS 7-79
MAX-PRI 7-81
SUBMIT-ALLOWED 7-107
attribute descriptions (continued)
TAPEDRIVES 7-110
command descriptions
ABORT SCHEDULER 6-30
ADD SCHEDULER 6-46
ALTER SCHEDULER 6-67
ASSUME SCHEDULER 6-73
INFO SCHEDULER 6-110
OPEN 6-113
RESET SCHEDULER 6-122
SET SCHEDULER 6-140
SHOW SCHEDULER 6-150
SHUTDOWN SCHEDULER 6-152
STATUS SCHEDULER 6-173
SWITCHCPU SCHEDULER 6-192
SWITCHLOG SCHEDULER 6-193
definition 1-3
determining if running 3-29
displaying 3-25, 3-29
SECURITY attachment-set attribute
description 7-99
synopsis and syntax summary 7-1
SELPRI job attribute
description 7-101
synopsis and syntax summary 7-4
SET ATTACHMENT-SET command
synopsis and syntax summary 6-13
SET CLASS command
description 6-133
synopsis and syntax summary 6-14
SET EXECUTOR command
description 6-134
synopsis and syntax summary 6-15
SET JOB command
description 6-136
synopsis and syntax summary 6-18
SET SCHEDULER command
description 6-140
synopsis and syntax summary 6-21

SHOW ATTACHMENT-SET command
description 6-142
synopsis and syntax summary 6-13
SHOW CLASS command
description 6-145
synopsis and syntax summary 6-14
SHOW EXECUTOR command
description 6-146
synopsis and syntax summary 6-15
SHOW JOB command
description 6-147
synopsis and syntax summary 6-18
SHOW SCHEDULER command
description 6-150
synopsis and syntax summary 6-21
SHUTDOWN SCHEDULER command
description 6-152
synopsis and syntax summary 6-21
SPECIAL-n job state
description 4-43, 6-170
SPI (Subsystem Programmatic Interface)
errors A-1
STALL job attribute
description 7-102
synopsis and syntax summary 7-5
START EXECUTOR command
description 6-155
synopsis and syntax summary 6-15
START SCHEDULER command
synopsis and syntax summary 6-22
STARTUP job attribute
description 7-104
synopsis and syntax summary 7-5
STATUS ATTACHMENT-SET command
description 6-160
synopsis and syntax summary 6-13
STATUS EXECUTOR command
description 6-163
synopsis and syntax summary 6-15
STATUS JOB command
description 6-165
synopsis and syntax summary 6-18
STATUS SCHEDULER command
description 6-173
synopsis and syntax summary 6-22
STATUS-HISTORY command
description 6-175
synopsis and syntax summary 6-18
STOP EXECUTOR command
description 6-178
synopsis and syntax summary 6-15
STOP executor state
description 3-31, 6-164
scheduler warm start, effect of 3-22
STOP JOB command
description 6-180
synopsis and syntax summary 6-18
STOP-ON-ABEND job attribute
description 7-105
synopsis and syntax summary 7-5
SUBMIT JOB command
synopsis and syntax summary 6-18
SUBMIT-ALLOWED scheduler attribute
description 7-107
synopsis and syntax summary 7-7
SUSPEND JOB command
description 6-190
synopsis and syntax summary 6-19
SUSPENDED job state, description 4-43, 6-172
SWAP job attribute
description 7-108
synopsis and syntax summary 7-5
SWITCHCPU SCHEDULER command
description 6-192
synopsis and syntax summary 6-22
SWITCHLOG SCHEDULER command
description 6-193
synopsis and syntax summary 6-22

SYSTEM command
description 6-195
synopsis and syntax summary 6-23

T

TAPE job state, description 4-43, 6-172
TAPEDRIVES job attribute
description 7-109
synopsis and syntax summary 7-5
TAPEDRIVES scheduler attribute
description 7-110
synopsis and syntax summary 7-7
TEMPORARY attachment-set attribute
description 7-112
synopsis and syntax summary 7-1
Temporary executor
See Executor, temporary
TEMP_EXEC_...
See __TEMP_EXEC_..., temporary executor name
TERM job attribute
description 7-114
in working-attributes set
setting 6-137
synopsis and syntax summary 7-5
use in commands
ALTER JOB 6-63
SET JOB 6-137
Terminal, home
See Home terminal
The 7-44
Time
attributes, definition 4-23
displayed by STATUS SCHEDULER command 6-173
format in commands 7-10, 7-15
future, definition 7-26
limit, for job execution 7-75
TIME job state, description 4-43, 6-172
Transitions, DST (daylight-saving time), effect on jobs 6-187
Twenty-four-hour time format, in commands 7-10, 7-15

U

Use 6-70
User ID format 6-74

V

VOLUME command
description 6-196
synopsis and syntax summary 6-23
VOLUME job attribute
description 7-116
synopsis and syntax summary 7-5

W

WAIT job attribute
description 7-118
synopsis and syntax summary 7-5
WAITON job attribute
description 7-119
synopsis and syntax summary 7-5
Wild-card characters 6-52

Z

ZBAT JOBINFO macro, returns job name and number 4-12
ZBAT RELEASE macro, releases dependent jobs 4-32, 7-120
ZBAT-nnnn job names 6-183

Special Characters

! (exclamation point) command
description 6-198
synopsis and syntax summary 6-23
$RELEASE command, releases dependent jobs 4-33
& (ampersand), line continuation character 6-76
? (question mark)
command
description 6-200
synopsis and syntax summary 6-23
wild-card character
See Wild-card characters
__TEMP_EXEC_..., temporary executor name 6-129, 6-164, 7-16
}} (double braces) prompt
in COMMENT command 6-76
in FC command 6-93