SCF Reference Manual for the Kernel Subsystem

HP Part Number: 523406-013
Published: May 2010
Edition: J06.03 and subsequent J-series RVUs, H06.03 and subsequent H-series RVUs, and G06.24 and subsequent G-series RVUs

Legal Notice

Table of Contents

List of Figures

List of Tables

About This Document

This manual describes the Kernel subsystem on NonStop S-series servers, Integrity NonStop NS-series servers, and Integrity NonStop BladeSystems. It also describes the Subsystem Control Facility (SCF) configuration and management tasks that can be performed on Kernel subsystem objects. Some of these tasks are:

- Replace Event Management Service (EMS) templates. (See "Changing EMS Template Files" (page 34).)
- Specify the power-failure-to-shutdown time interval. (See "Changing the Power-Failure-to-Shutdown Time Interval" (page 35).)
- Alter the system name and system number. (See "Changing the System Name or System Number (G-Series RVUs)" (page 36), "Changing the System Name or System Number (H-Series RVUs)" (page 37), or "Changing the System Name or System Number (J-Series RVUs)" (page 40).)
- Alter the system time attributes. (See "Changing the System Time Attributes" (page 41).)
- Add, alter, and delete generic processes. (See Chapter 3 (page 47).)
- Obtain information about a ServerNet X fabric or Y fabric. (See "Obtaining Information About the ServerNet Network" (page 75).)

Supported Release Version Updates (RVUs)

This manual supports J06.03 and all subsequent J-series RVUs, H06.03 and all subsequent H-series RVUs, and G06.24 and all subsequent G-series RVUs, until otherwise indicated in a replacement publication.

Intended Audience

This manual is written for anyone who is responsible for configuring and managing a NonStop S-series or Integrity NonStop NS-series server, including:

- Configuring new systems
- Changing or adding to existing system configurations
- Displaying the status of objects on the system
- Configuring or changing generic processes

New and Changed Information in This Edition

Changes to 523406–013

The TLE_LIMIT attribute is now valid for G-series as of the G06.32.01 RVU. For details on this feature, see "Changing the System TLE Limit Attribute" (page 44).

Changes to 523406–012

- Throughout the manual, clarified statement about pending attributes in INFO SUBSYS displays.
- Updated displays throughout the manual to include the system attribute DESTINATION_CONTROL_LIMIT.
- Updated displays throughout the manual to include new system attributes TLE_LIMIT and AUTO_RETRY_ON_ERROR_654.
- Added "Changing the Destination Control Table Size Limit" (page 43) to describe how to us the DESTINATION_CONTROL_TABLE attribute. See also "DESTINATION_CONTROL_TABLE." [p. 102]

- Added "Changing the System TLE Limit Attribute" (page 44) to describe how to use the new TLE_LIMIT attribute. See also "TLE_LIMIT" [p. 105]. This new attribute is valid with the J06.09 and H06.20 RVUs.
- Added "Changing Software Data Integrity Checking" (page 45) to descirbe how to use the new AUTO_RETRY_ON_ERROR_654 attribute. See also "AUTO_RETRY_ON_ERROR_654" [p. 105]. This new attribute is valid with the J06.09 and H06.20 RVUs.
- Added new SCF error messages 00132, 00133, and 00134. See Appendix B (page 145).
- Throughout the manual, clarified statement about pending attributes in INFO SUBSYS displays.

Changes to 523406–011

- Throughout this manual, added references to H-series manuals and J-series manuals.
- Under "Generic Processes" (page 17), added the $ZZCIP and the $ZCM*nn* CIP generic processes.
- Added detailed information about the CIP manager process and the CIP monitor process under "Displaying Information about the CIP Manager Process " (page 21)

  and "Displaying Information about the CIP Monitor Process" (page 22).
- To clarify information on power failure procedures, changed these sections:
  - "Changing the Power-Failure-to-Shutdown Time Interval" (page 35)
  - POWERFAIL_DELAY_TIME option of the ALTER SUBSYS Command.
- Added the new section, "Changing the System Name or System Number (J-Series RVUs)" (page 40).

Changes to 523406–010

- Supported release statements have been updated to include J-series RVUs.

Changes to 523406–009

- Removed the unsupported attributes, MISALIGNLOG and NATIVEATOMICMISALIGN, in Chapter 2: Configuring System Attributes (page 33) and Chapter 5: SCF Object Types and Object Names (page 79).
- Changed the default value of the TNSMISALIGN attribute to FAIL in Chapter 2: Configuring System Attributes (page 33) and Chapter 5: SCF Object Types and Object Names (page 79).
- Removed the value ROUND from the TNSMISALIGN attribute in Chapter 2: Configuring System Attributes (page 33) and Chapter 5: SCF Object Types and Object Names (page 79).
- Added a new consideration under "Considerations for Adding an ASSIGN, PARAM, or DEFINE" (page 70).
- Updated the USA66 option to change the DST in Chapter 6: SCF Commands for the Kernel Subsystem (page 83).

Changes to 523406–008

Chapter	Change
Chapter 2: Configuring System Attributes (page 33)	Added a note under "Changing Data Misalignment Attribute" (page 42) to clarify that the MISALIGNLOG, TNSMISALIGN, and NATIVEATOMICMISALIGN attributes are supported only on systems running G-series RUVs.
Chapter 3 (page 47)	Added a new subsection, "Configuring and Managing ASSIGNs, PARAMs, and DEFINEs for a Generic Process" (page 68) that explains: • Adding an ASSIGN to a Generic Process • Adding a PARAM to a Generic Process • Adding a DEFINE to a Generic Process • Altering the ASSIGN Attribute of a Generic Process • Altering the PARAM Attribute of a Generic Process • Altering the DEFINE Attribute of a Generic Process • Deleting the ASSIGN Attribute of a Generic Process • Deleting the PARAM Attribute of a Generic Process • Deleting the DEFINE Attribute of a Generic Process
Chapter 6 (page 83)	Added these new subsections: • "ADD Command for Using ASSIGNs" (page 94) • "ADD Command for Using PARAMs" (page 94) • "ADD Command for Using DEFINEs" (page 94) • "ALTER Command for Using ASSIGNs" (page 106) • "ALTER Command for Using PARAMs" (page 107) • "ALTER Command for Using DEFINEs" (page 107) • "DELETE Command for Using ASSIGNs, PARAMs, and DEFINEs" (page 108) • "INFO Command for Using ASSIGNs, PARAMs, and DEFINEs" (page 118)
Appendix A: SCF Command Summary for the Kernel Subsystem (page 143)	Updated the SCF commands to include ASSIGN, PARAM, and DEFINE attributes.
Appendix B: SCF Kernel Subsystem Error Messages (page 145)	Added these new error messages: • 00125 • 00126 • 00127 • 00128 • 00129 • 00130 • 00131

Changes to 523406–007

Chapter	Change
Chapter 1: Kernel Subsystem Overview (page 17)	A subsection entitled "Displaying Information About the FCS Monitor Process" (page 30). The subsection documents the INFO command for the FCS Monitor process, a generic process that monitors disk drive enclosures.
Chapter 2: Configuring System Attributes (page 33)	Information about the RIDETHRUONLY option added to the subsection titled "Changing the Power-Failure-to-Shutdown Time Interval" (page 35). The subsection "Changing the System Name and Number" renamed "Changing the System Name or System Number (G-Series RVUs)" (page 36). A new subsection titled "Changing the System Name or System Number (H-Series RVUs)" (page 37).
Chapter 6 (page 83)	Documentation of the RIDETHRUONLY option of the POWERFAIL_DELAY_TIME attribute for the ALTER SUBSYS command.

Document Organization

Chapter	Contents
Chapter 1: Kernel Subsystem Overview (page 17)	Describes the $ZZKRN Kernel subsystem manager and the SCF interface to the Kernel subsystem.
Chapter 2: Configuring System Attributes (page 33)2	Describes how to view and change system attributes controlled by the $ZZKRN Kernel subsystem manager.
Chapter 3: Configuring and Managing Generic Processes (page 47)	Describes how to configure and manage generic processes controlled by the $ZZKRN Kernel subsystem manager.
Chapter 4: Managing the ServerNet Network (page 75)	Describes how to manage the ServerNet X and Y fabrics.
Chapter 5: SCF Object Types and Object Names (page 79)	Describes SCF object types and object names supported by the Kernel subsystem.
Chapter 6: SCF Commands for the Kernel Subsystem (page 83)	Describes the SCF commands that support the Kernel subsystem.
Appendix A: SCF Command Summary for the Kernel Subsystem (page 143)	Contains a syntax summary of the SCF commands that support the Kernel subsystem.
Appendix B: SCF Kernel Subsystem Error Messages (page 145)	Describes the SCF error messages that apply to the Kernel subsystem.

This manual also contains a glossary of technical terms and abbreviations used throughout the text.

Notation Conventions

General Syntax Notation

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

UPPERCASE LETTERS

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

MAXATTACH

Italic Letters

Italic letters, regardless of font, indicate variable items that you supply. Items not enclosed in brackets are required. For example:

```
file-name
```

Computer Type

Computer type letters indicate:

- C and Open System Services (OSS) keywords, commands, and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

  Use the `cextdecs.h` header file.

- Text displayed by the computer. For example:

  ```
  Last Logon: 14 May 2006, 08:02:23
  ```

- A listing of computer code. For example

  ```
  if (listen(sock, 1) < 0)
  {
  perror("Listen Error");
  exit(-1);
  }
  ```

Bold Text

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

```
ENTER RUN CODE

?123
CODE RECEIVED:      123.00
```

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

[ ] 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 can 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 can 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 typed 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 type 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 this example, no spaces are 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 ]…
```

Notation for Messages

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

Bold Text

Bold text in an example indicates user input typed 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.
```

Italic Text

Italic text indicates 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 can 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 can 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
```

Related Information

Manuals describing the NonStop servers are organized into several sets of manuals, including the *NonStop S-Series Planning and Configuration Guide*, the appropriate NonStop NS-series planning guide, or the *NonStop BladeSystem Planning Guide*.

Publishing History

Part Number	Product Version	Publication Date
523406-009	SYSTEM CONFIG MGR G06 SYSTEM CONFIG MGR H01	August 2006
523406-010	SYSTEM CONFIG MGR G06 SYSTEM CONFIG MGR H01	May 2008

Part Number	Product Version	Publication Date
523406-011	SYSTEM CONFIG MGR G06 SYSTEM CONFIG MGR H01	November 2009
523406-012	SYSTEM CONFIG MGR G06 SYSTEM CONFIG MGR H01	February 2010
523406-013	SYSTEM CONFIG MGR G06 SYSTEM CONFIG MGR H01	May 2010

HP Encourages Your Comments

HP encourages your comments concerning this document. We are committed to providing documentation that meets your needs. Send any errors found, suggestions for improvement, or compliments to **docsfeedback@hp.com**.

Include the document title, part number, and any comment, error found, or suggestion for improvement you have concerning this document.

1 Kernel Subsystem Overview

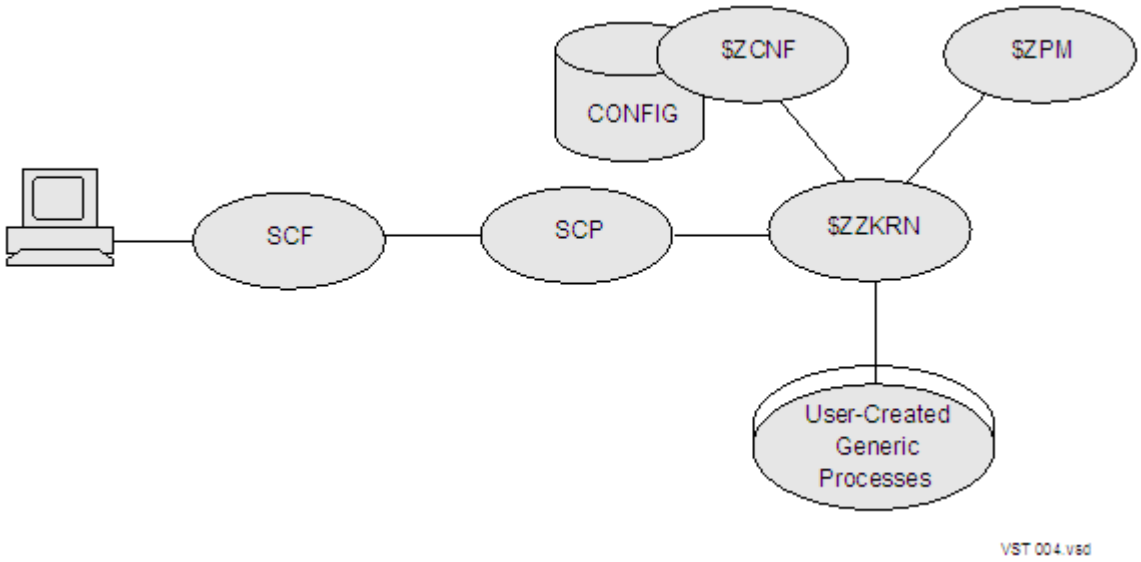
This chapter describes the $ZZKRN Kernel subsystem manager and the SCF interface to the Kernel subsystem.

SCF Interface to the Kernel Subsystem

The Kernel subsystem configures and maintains the subsystem managers for NonStop servers running G-series, H-series, and J-series RVUs. You can use SCF to configure, control, and inquire about the $ZZKRN Kernel subsystem manager process and other generic processes on these systems.

Figure 1-1 shows the SCF interface to the Kernel subsystem.

Figure 1-1 SCF Interface to the Kernel Subsystem



The interfaces between SCF, the Subsystem Control Point (SCP), and $ZZKRN are described in both the *SCF Reference Manual for G-Series RVUs* and the *SCF Reference Manual for J-Series and H-Series RVUs*. The other interfaces with $ZZKRN are described in the subsections that follow.

Components of the Kernel Subsystem

This subsection describes the components of the Kernel subsystem.

Generic Processes

A generic process is a process that is created by the operating system or a user in order to perform a task. Generic processes are different from I/O processes and are described more thoroughly under "Definition of a Generic Process" (page 47).

Generic processes are an important part of the configuration of NonStop systems because the subsystem managers supported by these systems are configured as generic processes. Listed in

the table are subsystem managers that are configured as generic processes, along with the TACL names of the manager or monitor processes:

Subsystem	Manager or Monitor Process Name	Manual
ATM	$ZZATM	*ATM Configuration and Management Manual*
CIP	$ZZCIP and $ZCM*nn*	*Cluster I/O Protocols (CIP) Configuration and Management Manual*
Expand	$ZEXP *	*Expand Configuration and Management Manual*
Kernel	$ZZKRN	This manual
PAM	$ZZPAM *	*PAM Configuration and Management Manual*
QIO	$ZM*nn*	*QIO Configuration and Management Manual*
ServerNet/FX Adapter	$ZZFOX	*ServerNet/FX Adapter Configuration and Management Manual*
SLSA	$ZZLAN	*LAN Configuration and Management Manual*
Storage	$ZZSTO	*SCF Reference Manual for the Storage Subsystem*
WAN	$ZZWAN	*WAN Subsystem Configuration and Management Manual*

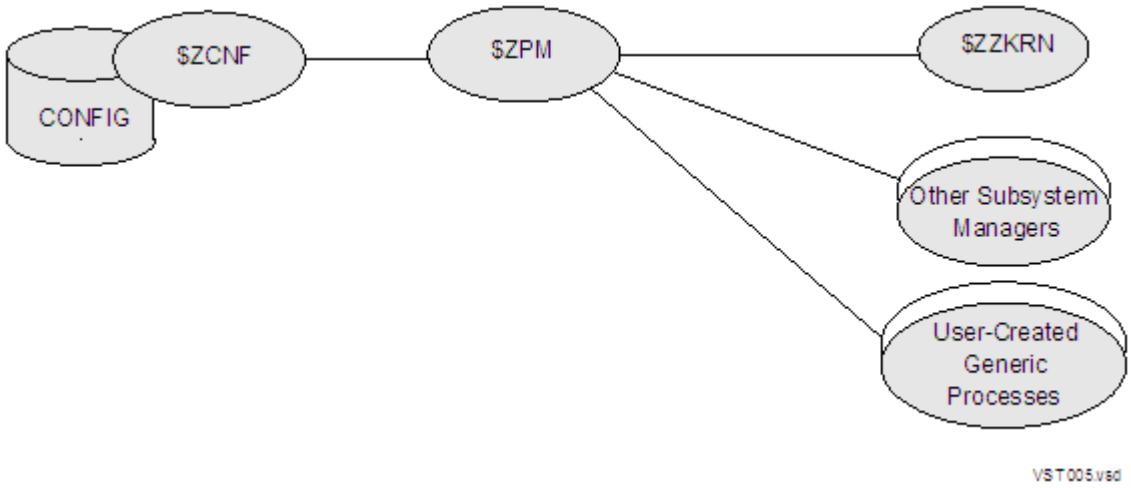
* Users have the option of configuring this process through TACL or SCF.

Although these subsystem managers are automatically configured during system startup, you can change their characteristics by using the SCF ALTER PROCESS command.

The $ZCNF Configuration Utility Process

The $ZCNF configuration utility process is configured to start early during system load in processors 0 and 1. The primary purpose of $ZCNF is to handle access to, and information requests about, the system configuration database file (CONFIG). Figure 1-2 shows the relationship of $ZCNF to the other processes that are started during a system load. The $ZCNF configuration utility process creates and manages the $ZPM persistence manager. Then $ZPM starts the generic processes configured for this system.

Figure 1-2 Process Creation Flow at System Startup



The $ZPM Persistence Manager

The $ZPM persistence manager is started and managed by the $ZCNF configuration utility process. It is the first process started during a system load.

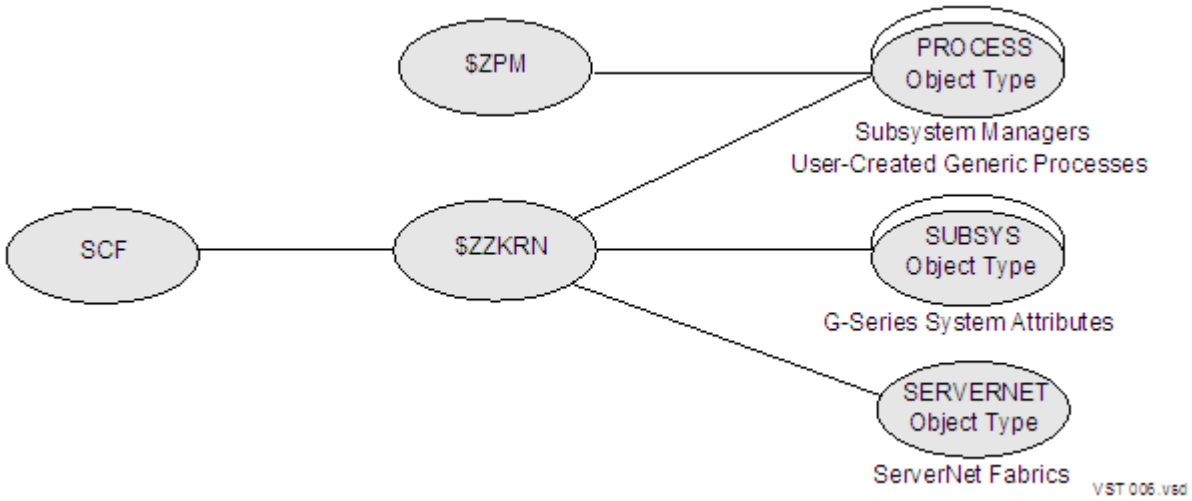
Based on information in the CONFIG system configuration database file located in the $SYSTEM.ZSYSCONF subvolume, $ZPM starts all generic processes and optionally manages their persistence. First it starts the subsystem managers during the KERNEL phase of system load. It also automatically restarts them if they stop while the system is up.

Then, $ZPM starts user-configured generic processes. If these generic processes are configured to automatically restart when stopped, $ZPM also restarts them. For more information, see "Persistence Considerations" (page 54).

The $ZZKRN Kernel Subsystem Manager

Like the other subsystem managers, the $ZZKRN Kernel subsystem manager is a persistent generic process. $ZZKRN configures and maintains the subsystem managers, user-configured generic processes, some system attributes, and (through $ZSNET) the ServerNet X and Y fabrics. Figure 1-3 shows the $ZZKRN Kernel subsystem manager and the objects under its control.

Figure 1-3 SCF Architecture of the Kernel Subsystem



For more information on these object types, see:

Object Type	Chapter
SUBSYS	Chapter 2: Configuring System Attributes (page 33)
PROCESS	Chapter 3: Configuring and Managing Generic Processes (page 47)
SERVERNET	Chapter 4: Managing the ServerNet Network (page 75)

Displaying Information About Subsystems

You can display information about the generic processes in the table. These generic processes are subsystem managers or monitor processes:

Subsystem	Manager or Monitor Process Name	Refer To
ATM	$ZZATM, the Asynchronous Transfer Mode (ATM) subsystem manager	20
CIP	$ZZCIP and $ZCM*nn*	21 and 22
Expand	$ZEXP, the Expand monitor process	23
Kernel	$ZZKRN, the Kernel subsystem manager	24
PAM	$ZZPAM, the Port Access Method (PAM) manager process	25
QIO	$ZM*nn*, the Query I/O (QIO) monitor process	25
ServerNet/FX Adapter	$ZZFOX, the FOX monitor process	26
SLSA	$ZZLAN, the ServerNet LAN Systems Access (SLSA) subsystem manager	27
Storage	$ZZSTO, the Storage subsystem manager	28
WAN	$ZZWAN, the Wide Area Network (WAN) subsystem manager	30

Displaying Information About the ATM Subsystem Manager

Information about the ATM subsystem manager is documented in the *ATM Configuration and Management Manual*. You can use the SCF STATUS and INFO commands to display information about the $ZZATM ATM subsystem manager. (The NAMES command is not supported for $ZZATM in the Kernel subsystem.)

- The STATUS command displays dynamic state information about $ZZATM:

```
-> STATUS PROCESS $ZZATM, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \SUN.$ZZKRN.#ZZATM

Backup PID........ None
Creation Time..... JAN 21,2000 12:41:59
Name.............. $ZZATM
OwnerID........... 255, 255
Primary PID....... 0 , 21
Priority.......... 180
```

```
State.............. STARTED
Substate..........
```

- This INFO command displays the configured attributes of $ZZATM:

```
-> INFO PROCESS $ZZATM, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \SUN.$ZZKRN.#ZZATM

 *AutoRestart..............10
 *BackupCPU................1
 *CPU......................Not Specified
 *DefaultVolume............$SYSTEM.SYSTEM
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$ZHOME
 *InFile...................$YMIOP.#CLCI
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$ZZATM
 *OutFile..................$ZHOME
 *PFSSize..................Not Specified
 *PrimaryCPU...............0
 *Priority.................180
 *Program..................$SYSTEM.SYSTEM.ATMASM
 *SaveAbend................ON
 *StartMode................KERNEL
 *StartupMessage...........<BCKP-CPU>
 *StopMode.................STANDARD
 *Type.....................OTHER
 *UserId...................SUPER.SUPER ( 255,255 )
```

Displaying Information about the CIP Manager Process

Information about the CIP manager process is documented in the *Cluster I/O Protocols (CIP) Configuration and Management Manual*. You can configure this process through SCF. You can use the SCF STATUS and INFO commands to display information about the $ZZCIP CIP manager process. (The NAMES command only shows $ZZCIP in the Kernel subsystem.)

- The STATUS command displays dynamic state information about #ZZCIP:

```
->STATUS PROCESS $ZZKRN.#ZZCIP, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \MYSYS.$ZZKRN.#ZZCIP

Backup PID........ 1 , 324
Creation Time....  SEP 18,2009 09:24:10
Name.............  $ZZCIP
OwnerID..........  255, 255
Primary PID......  0 , 343
Priority.........  200
State............  STARTED
Substate..........
```

- The INFO command displays the configured attributes of #ZZCIP:

```
-> INFO PROCESS $ZZKRN.#ZZCIP, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \MYSYS.$ZZKRN.#ZZCIP

 *AutoRestart..................10
 *BackupCPU....................1
 *CPU..........................Not Specified
 *DefaultVolume................$SYSTEM.JACIPMAN
 *ExtSwap......................Not Specified
 *Highpin......................ON
 *HomeTerminal.................$ZHOME
```

```
*InFile.....................Not Specified
*Library....................Not Specified
*MemPages...................Not Specified
*Name.......................$ZZCIP
*OutFile....................Not Specified
*PFSSize....................Not Specified
*PrimaryCPU.................0
*Priority...................146
*Program....................$SYSTEM.SYSTEM.CIPMAN
*SaveAbend..................OFF
*StartMode..................SYSTEM
*StartupMessage.............<BCKP-CPU>
*StopMode...................SYSMSG
*Type.......................OTHER
*UserId.....................SUPER.SUPER  ( 255,255 )
```

Displaying Information about the CIP Monitor Process

Information about the CIP monitor process is documented in the *Cluster I/O Protocols (CIP) Configuration and Management Manual*. You can configure this process through SCF. You can use the SCF STATUS and INFO commands to display information about the $ZZCIP CIP manager process. (The NAMES command only shows $CIPMON in the Kernel subsystem.)

- The STATUS command displays dynamic state information about #CIPMON processes, named $ZCM*nn*:

```
-> STATUS PROCESS $ZZKRN.#CIPMON, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \MYSYS.$ZZKRN.#CIPMON

Backup PID........ None
Creation Time....  SEP 18,2009 09:24:43
Name.............  $ZCM01
OwnerID..........  255, 255
Primary PID......  1 , 346
Priority.........  200
State............  STARTED
Substate...........

NONSTOP KERNEL - Detailed Status PROCESS \MYSYS.$ZZKRN.#CIPMON

Backup PID........ None
Creation Time....
Name.............  $ZCM04
OwnerID............
Primary PID....... None
Priority...........
State..............STOPPED
Substate...........
.
.
.
```

- The INFO command displays the configured attributes that apply to all monitor processes ($ZCM*nn*):

```
-> INFO PROCESS $ZZKRN.#CIPMON, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \MYSYS.$ZZKRN.#CIPMON

*AutoRestart..................10
*BackupCPU....................Not Specified
*CPU..........................ALL
*DefaultVolume................$SYSTEM.NOSUBVOL
*ExtSwap......................Not Specified
```

```
*Highpin.....................ON
*HomeTerminal................$ZHOME
*InFile......................$ZHOME
*Library.....................Not Specified
*MemPages....................Not Specified
*Name........................$ZCMnn
*OutFile.....................$ZHOME
*PFSSize.....................Not Specified
*PrimaryCPU..................Not Specified
*Priority....................146
*Program.....................$SYSTEM.SYSTEM.CIPMON
*SaveAbend...................OFF
*StartMode...................SYSTEM
*StartupMessage..............Not Specified
*StopMode....................SYSMSG
*Type........................OTHER
*UserId......................SUPER.SUPER ( 255,255 )
```

Displaying Information About the Expand Monitor Process

Information about the Expand monitor process is documented in the *Expand Configuration and Management Manual*. You can configure this process through TACL or SCF. You can use the SCF STATUS and INFO commands to display information about the $ZEXP Expand monitor process. (The NAMES command is not supported for $ZEXP in the Kernel subsystem.)

- The STATUS command displays dynamic state information about $ZEXP:

```
-> STATUS PROCESS $ZEXP, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \SUN.$ZZKRN.#ZEXP

Backup PID........ None
Creation Time..... JAN 21,2000 10:41:59
Name.............. $ZEXP
OwnerID........... 255, 255
Primary PID....... 0 , 229
Priority.......... 180
State............. STARTED
Substate..........
```

- This INFO command displays the configured attributes of $ZEXP:

```
-> INFO PROCESS $ZEXP, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \SUN.$ZZKRN.#ZEXP

 *AutoRestart..............10
 *BackupCPU................1
 *CPU......................Not Specified
 *DefaultVolume............$SYSTEM.NOSUBVOL
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$ZHOME
 *InFile...................Not Specified
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$ZEXP
 *OutFile..................$ZHOME
 *PFSSize..................Not Specified
 *PrimaryCPU...............0
 *Priority.................180
 *Program..................$SYSTEM.SYSTEM.OZEXP
 *SaveAbend................OFF
 *StartMode................SYSTEM
 *StartupMessage...........<BCKP-CPU>
 *StopMode.................STANDARD
```

```
*Type.....................OTHER
*UserId...................SUPER.SUPER ( 255,255 )
```

Displaying Information About the Kernel Subsystem Manager

You can use the SCF STATUS, NAMES, and INFO commands to display information about the $ZZKRN Kernel subsystem manager:

- The STATUS command displays dynamic state information about $ZZKRN:

```
-> STATUS PROCESS $ZZKRN, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \SUN.$ZZKRN.#ZZKRN

Backup PID........ 1 , 20
Creation Time..... JAN 17,2000 11:41:59
Name.............. $ZZKRN
OwnerID........... 255, 255
Primary PID....... 0 , 11
Priority.......... 180
State............. STARTED
Substate..........
```

- This NAMES command displays the names of all generic processes created by the Kernel subsystem:

```
-> NAMES PROCESS $ZZKRN

NONSTOP KERNEL - Names PROCESS \SUN.$ZZKRN

Process
$ZZKRN.#CEV-SERVER-MANAGER-P0     $ZZKRN.#CEV-SERVER-MANAGER-P1
$ZZKRN.#CLCI-TACL                 $ZZKRN.#CHK
$ZZKRN.#OSM-APPSRVR               $ZZKRN.#OSM-CIMOM
$ZZKRN.#OSM-CONFLH-RD             $ZZKRN.#OSM-OEV
$ZZKRN.#QIOMON                    $ZZKRN.#ROUTING-DIST
$ZZKRN.#TCPIP-ZTC02               $ZZKRN.#TSM-SNMP
$ZZKRN.#SP-EVENT                  $ZZKRN.#TSM-SRM
$ZZKRN.#ZLOG                      $ZZKRN.#ZTCP0
$ZZKRN.#ZTCP1                     $ZZKRN.#ZHOME
$ZZKRN.#ZZKRN                     $ZZKRN.#ZZLAN
$ZZKRN.#ZZSTO                     $ZZKRN.#ZZWAN
```

- This INFO command displays the configured attributes of $ZZKRN:

```
-> INFO PROCESS $ZZKRN, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \SUN.$ZZKRN.#ZZKRN

 *AutoRestart..............10
 *BackupCPU................1
 *CPU......................Not Specified
 *DefaultVolume............$SYSTEM.SYSTEM
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$ZHOME
 *InFile...................$YMIOP.#CLCI
 *Library..................Not Specified
 *MemPages.................0
 *Name.....................$ZZKRN
 *OutFile..................$ZHOME
 *PFSSize..................Not Specified
 *PrimaryCPU...............0
 *Priority.................180
 *Program..................$SYSTEM.SYSTEM.OZKRN
 *SaveAbend................ON
```

```
*StartMode.................KERNEL
*StartupMessage............<BCKP-CPU>
*StopMode..................STANDARD
*Type......................SUBSYSTEM-MANAGER
*UserId....................SUPER.SUPER ( 255,255 )
```

Displaying Information About the PAM Manager Process

Information about the PAM manager process is documented in the *PAM Configuration and Management Manual*. You can configure this process through TACL or SCF. If you configure this process using SCF, you can use the SCF STATUS and INFO commands to display information about the PAM manager process. (The NAMES command is not supported for $ZZPAM in the Kernel subsystem.)

- The STATUS command displays dynamic state information about $ZZPAM:

```
-> STATUS PROCESS $ZZKRN.#ZZPAM, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \SUN.$ZZKRN.#ZZPAM

Backup PID........ 1 , 277
Creation Time..... JAN 21,2000 10:41:59
Name.............. $ZZPAM
OwnerID........... 255, 255
Primary PID....... 0 , 319
Priority.......... 167
State............. STARTED
Substate..........
```

- This INFO command displays the configured attributes of $ZZPAM:

```
-> INFO PROCESS $ZZKRN.#ZZPAM, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \SUN.$ZZKRN.#ZZPAM

 *AutoRestart..............10
 *BackupCPU................1
 *CPU......................Not Specified
 *DefaultVolume............$SYSTEM.NOSUBVOL
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............Not Specified
 *InFile...................Not Specified
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$PAM1
 *OutFile..................Not Specified
 *PFSSize..................Not Specified
 *PrimaryCPU...............0
 *Priority.................167
 *Program..................$SYSTEM.SYSTEM.PAM
 *SaveAbend................OFF
 *StartMode................APPLICATION
 *StartupMessage............"1 -LIF LAN01 -STARTDOWN -MSAPSTARTDOWN"
 *StopMode.................STANDARD
 *Type.....................OTHER
 *UserId...................SUPER.SUPER ( 255,255 )
```

Displaying Information About the QIO Monitor Process

Information about the QIO monitor process is documented in the *QIO Configuration and Management Manual*. You can use the SCF STATUS and INFO commands to display information

about the $ZM*nn* QIO monitor process (known to the Kernel subsystem manager as QIOMON). (The NAMES command is not supported for QIOMON in the Kernel subsystem.)

- The STATUS command displays dynamic state information about QIOMON:

```
-> STATUS PROCESS $ZZKRN.#QIOMON, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \SUN.$ZZKRN.#QIOMON

Backup PID........ None
Creation Time..... JAN 17,2000 11:41:59
Name.............. $ZM01
OwnerID........... 255, 255
Primary PID....... 1 , 8
Priority.......... 201
State............. STARTED
Substate..........
```

- This INFO command displays the configured attributes of QIOMON:

```
-> INFO PROCESS $ZZKRN.#QIOMON, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \SUN.$ZZKRN.#QIOMON

 *AutoRestart..............10
 *BackupCPU................Not Specified
 *CPU......................All
 *DefaultVolume............$SYSTEM.SYSTEM
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$ZHOME
 *InFile...................$YMIOP.#CLCI
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$ZM01
 *OutFile..................$ZHOME
 *PFSSize..................Not Specified
 *PrimaryCPU...............0
 *Priority.................199
 *Program..................$SYSTEM.SYSTEM.QIOMON
 *SaveAbend................OFF
 *StartMode................KERNEL
 *StartupMessage...........Not Specified
 *StopMode.................STANDARD
 *Type.....................OTHER
 *UserId...................SUPER.SUPER    (  255,255  )
```

Displaying Information About the FOX Monitor Process

Information about the FOX monitor process, $ZZFOX, is documented in the *ServerNet/FX Adapter Configuration and Management Manual*. You can use the SCF STATUS and INFO commands to display information about $ZZFOX. (The NAMES command is not supported for $ZZFOX in the Kernel subsystem.)

- The STATUS command displays dynamic state information about $ZZFOX:

```
-> STATUS PROCESS $ZZKRN.#ZZFOX, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \SUN.$ZZKRN.#ZZFOX

Backup PID........ 1 , 317
Creation Time..... JAN 17,2000 11:42:00
Name.............. $ZZFOX
OwnerID........... 255, 255
Primary PID....... 0 , 273
Priority.......... 199
```

```
State.............. STARTED
Substate..........
```

- This INFO command displays the configured attributes of $ZZFOX:

```
-> INFO PROCESS $ZZKRN.#ZZFOX, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#ZZFOX

 *AutoRestart..............0
 *BackupCPU................1
 *CPU......................Not Specified
 *DefaultVolume............$SYSTEM.SYSTEM
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$ZHOME
 *InFile...................Not Specified
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$ZZFOX
 *OutFile..................Not Specified
 *PFSSize..................Not Specified
 *PrimaryCPU...............0
 *Priority.................199
 *Program..................$SYSTEM.SYSTEM.FOXMON
 *SaveAbend................OFF
 *StartMode................MANUAL
 *StartupMessage...........Not Specified
 *StopMode.................STANDARD
 *Type.....................FOXMON
 *UserId...................SUPER.SUPER    (  255,255  )
```

Displaying Information About the SLSA Subsystem Manager

Information about the SLSA subsystem manager is documented in the *LAN Configuration and Management Manual*. You can use the SCF STATUS, NAMES, and INFO commands to display information about the $ZZLAN ServerNet SLSA subsystem manager.

- The STATUS command displays dynamic state information about $ZZLAN:

```
-> STATUS PROCESS $ZZKRN.#ZZLAN, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \EAST.$ZZKRN.#ZZLAN

Backup PID........ 1 , 10
Creation Time..... JAN 17,2000 16:19:30
Name.............. $ZZLAN
OwnerID........... 255, 255
Primary PID....... 0 , 10
Priority.......... 180
State............. STARTED
Substate..........
```

- The NAMES command displays the names of all processes controlled by the SLSA subsystem:

```
-> NAMES PROCESS $ZZLAN

SLSA Names PROCESS \SUN.$ZZLAN

PROCESS
$ZZLAN

LIF
$ZZLAN.LAN010 $ZZLAN.LAN011 $ZZLAN.LAN021 $ZZLAN.LAN022 $ZZLAN.LAN023
$ZZLAN.LAN012

ADAPTER
$ZZLAN.E0153
```

```
SAC
$ZZLAN.E0153.0

PIF
$ZZLAN.E0153.0.A

…
ADAPTER
$ZZLAN.MIOE01

SAC
$ZZLAN.MIOE0.0

PIF
$ZZLAN.MIOE0.0.A

MON
$ZZLAN.#ZLM00  $ZZLAN.#ZLM01  $ZZLAN.#ZLM02  $ZZLAN.#ZLM03  $ZZLAN.#ZLM04
$ZZLAN.#ZLM05  $ZZLAN.#ZLM06  $ZZLAN.#ZLM07  $ZZLAN.#ZLM08  $ZZLAN.#ZLM09
$ZZLAN.#ZLM10  $ZZLAN.#ZLM11  $ZZLAN.#ZLM12  $ZZLAN.#ZLM13  $ZZLAN.#ZLM14
$ZZLAN.#ZLM15
```

- This INFO command displays the configured attributes of $ZZLAN:

```
-> INFO PROCESS $ZZKRN.#ZZLAN, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \SUN.$ZZKRN.#ZZLAN

 *AutoRestart..............10
 *BackupCPU................1
 *CPU......................Not Specified
 *DefaultVolume............$SYSTEM.SYSTEM
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$ZHOME
 *InFile...................$YMIOP.#CLCI
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$ZZLAN
 *OutFile..................$ZHOME
 *PFSSize..................Not Specified
 *PrimaryCPU...............0
 *Priority.................180
 *Program..................$SYSTEM.SYSTEM.LANMAN
 *SaveAbend................ON
 *StartMode................KERNEL
 *StartupMessage...........<BCKP-CPU>
 *StopMode.................SYSMSG
 *Type.....................OTHER
 *UserId...................SUPER.SUPER ( 255,255 )
```

Displaying Information About the Storage Subsystem Manager

Information about the storage subsystem manager is documented in the *SCF Reference Manual for the Storage Subsystem*. You can use the SCF STATUS, NAMES, and INFO commands to display information about the $ZZSTO storage subsystem manager.

- The STATUS command displays dynamic state information about $ZZSTO:

```
-> STATUS PROCESS $ZZKRN.#ZZSTO, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \SUN.$ZZKRN.#ZZSTO

Backup PID........ 1 , 317
Creation Time..... JAN 17,2000 11:42:00
Name.............. $ZZSTO
```

```
OwnerID.......... 255, 255
Primary PID...... 0 , 273
Priority......... 180
State............ STARTED
Substate.........
```

- Because the storage subsystem has no PROCESS objects, you can use the NAMES command to display the names of all objects controlled by the storage subsystem:

```
-> NAMES $ZZSTO

STORAGE Names SUBSYS \SUN.$ZZSTO

SUBSYS
$ZZSTO

ADAPTER
$ZZSTO.#SNDA.GRP-1.MOD-1.SLOT-53  $ZZSTO.#SNDA.GRP-1.MOD-1.SLOT-54

DISK
$SYSTEM  $D0101   $D0103   $D0105   $D0107   $DSMSCM  $AUDIT

MON
$ZSMS

POOL
$POOL6  $POOL5  $POOL4  $POOL3  $POOL2  $POOL1

PROFILE
$ZZSTO.INTERNAL-DISK

SCSI
$SCZT0

TAPE
$TAPE1
```

- This INFO command displays the configured attributes of $ZZSTO:

```
-> INFO PROCESS $ZZKRN.#ZZSTO, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \SUN.$ZZKRN.#ZZSTO

 *AutoRestart..............10
 *BackupCPU................1
 *CPU......................Not Specified
 *DefaultVolume............$SYSTEM.SYSTEM
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$ZHOME
 *InFile...................$YMIOP.#CLCI
 *Library..................Not Specified
 *MemPages.................0
 *Name.....................$ZZSTO
 *OutFile..................$ZHOME
 *PFSSize..................Not Specified
 *PrimaryCPU...............0
 *Priority.................180
 *Program..................$SYSTEM.SYSTEM.TZSTO
 *SaveAbend................ON
 *StartMode................KERNEL
 *StartupMessage...........<BCKP-CPU>
 *StopMode.................STANDARD
 *Type.....................SUBSYSTEM-MANAGER
 *UserId...................SUPER.SUPER ( 255,255 )
```

Displaying Information About the FCS Monitor Process

For disk drive enclosures containing M8xxx fibre channel disks, the Fibre Channel Storage (FCS) Monitor process monitors environmental limits, tracks and reports hardware changes, and issues control commands to the Environmental Monitoring Unit (EMU). You can use the SCF STATUS and INFO commands to display information about the FCS Monitor process.

- The STATUS command displays dynamic state information about FCSMON.

```
-> STATUS PROCESS $ZZKRN.#FCSMON

NONSTOP KERNEL - Status PROCESS \IO.$ZZKRN.#FCSMON

Symbolic Name             Name   State    Sub Primary  Backup   Owner
                                                PID      PID      ID
FCSMON                    $ZFC00 STARTED      0 ,277   None     255,255
FCSMON                    $ZFC01 STARTED      1 ,427   None     255,255
FCSMON                    $ZFC02 STARTED      2 ,263   None     255,255
FCSMON                    $ZFC03 STARTED      3 ,398   None     255,255
FCSMON                    $ZFC04 STOPPED      None     None
FCSMON                    $ZFC05 STOPPED      None     None
FCSMON                    $ZFC06 STOPPED      None     None
FCSMON                    $ZFC07 STOPPED      None     None
FCSMON                    $ZFC08 STOPPED      None     None
FCSMON                    $ZFC09 STOPPED      None     None
FCSMON                    $ZFC10 STOPPED      None     None
FCSMON                    $ZFC11 STOPPED      None     None
FCSMON                    $ZFC12 STOPPED      None     None
FCSMON                    $ZFC13 STOPPED      None     None
FCSMON                    $ZFC14 STOPPED      None     None
FCSMON                    $ZFC15 STOPPED      None     None
```

- The INFO PROCESS command displays attributes of the FCSMON process.

```
-> STATUS PROCESS $ZZKRN.#FCSMON

NONSTOP KERNEL - Info PROCESS \IO.$ZZKRN.#FCSMON

Symbolic Name             *Name  *Autorestart *Program
FCSMON                    $ZFCnn  10           $SYSTEM.SYSTEM.FCSMON
```

Displaying Information About the WAN Subsystem Manager

Information about the WAN subsystem manager is documented in the *WAN Subsystem Configuration and Management Manual*. You can use the SCF STATUS, NAMES, and INFO commands to display information about the $ZZWAN WAN subsystem manager.

- The STATUS command displays dynamic state information about $ZZWAN:

```
-> STATUS PROCESS $ZZKRN.#ZZWAN, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \SUN.$ZZKRN.#ZZWAN

Backup PID........ 1 , 285
Creation Time..... JAN 17,2000 11:41:59
Name.............. $ZZWAN
OwnerID........... 255, 255
Primary PID....... 0 , 272
Priority.......... 180
State............. STARTED
Substate..........
```

- The NAMES command can display the names of all objects controlled by the WAN PROCESS object type; that is, it displays the names of the WAN configuration manager, TFTP server, SNMP trap multiplexer, and WANboot processes:

```
-> NAMES PROCESS $ZZWAN.#*

WANMgr Names  PROCESS $ZZWAN.*

PROCESS
$ZZWAN.#0
$ZZWAN.#1
$ZZWAN.#ZF018
$ZZWAN.#ZF01C
$ZZWAN.#ZTMX1
$ZZWAN.#ZW018
$ZZWAN.#ZW01C
```

- This INFO command displays the configured attributes of $ZZWAN:

```
-> INFO PROCESS $ZZKRN.#ZZWAN, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \SUN.$ZZKRN.#ZZWAN

 *AutoRestart..............10
 *BackupCPU................1
 *CPU......................Not Specified
 *DefaultVolume............$SYSTEM.SYSTEM
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$ZHOME
 *InFile...................$YMIOP.#CLCI
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$ZZWAN
 *OutFile..................$ZHOME
 *PFSSize..................Not Specified
 *PrimaryCPU...............0
 *Priority.................180
 *Program..................$SYSTEM.SYSTEM.WANMAN
 *SaveAbend................OFF
 *StartMode................KERNEL
 *StartupMessage...........<BCKP-CPU>
 *StopMode.................STANDARD
 *Type.....................SUBSYSTEM-MANAGER
 *UserId...................SUPER.SUPER    (  255,255  )
```

2 Configuring System Attributes

This chapter describes how to use SCF to view and reconfigure system attributes for systems running G-series, H-series, and J-series RVUs. If you are logged on as a super group user (255,*n*), you can use SCF to reconfigure these attributes.

Attribute	Topic	Page
DAYLIGHT_SAVING_TIME*	"Changing the System Time Attributes"	41
NONRESIDENT_TEMPLATES**	"Changing EMS Template Files"	34
POWERFAIL_DELAY_TIME**	"Changing the Power-Failure-to-Shutdown Time Interval"	35
RESIDENT_TEMPLATES**	"Changing EMS Template Files"	34
SYSTEM_NAME (G-series)*	"Changing the System Name or System Number (G-Series RVUs)"	36
SYSTEM_NUMBER (G-series)*	"Changing the System Name or System Number (G-Series RVUs)"	36
SYSTEM_NAME (H-series)*	"Changing the System Name or System Number (H-Series RVUs)"	37
SYSTEM_NUMBER (H-series)*	"Changing the System Name or System Number (H-Series RVUs)"	37
SYSTEM_NAME (J-series)*	"Changing the System Name or System Number (J-Series RVUs)"	40
SYSTEM_NUMBER (J-series)*	"Changing the System Name or System Number (J-Series RVUs)"	40
TIME_ZONE_OFFSET*	"Changing the System Time Attributes"	41
TNSMISALIGN**	"Changing Data Misalignment Attribute"	42
DESTINATION_CONTROL_TABLE**	"Changing the Destination Control Table Size Limit"	43
TLE_LIMIT**	"Changing the System TLE Limit Attribute"	44
AUTO_RETRY_ON_ERROR_654**	"Changing Software Data Integrity Checking"	45

* These changes take effect at the next Manual Reload or Hard Reset of the system.

** These changes take effect immediately.

Displaying System Attributes

The INFO SUBSYS $ZZKRN command displays these system attributes; for example:

```
-> INFO SUBSYS $ZZKRN

NONSTOP KERNEL - Info SUBSYS  \EAST.$ZZKRN

Current Settings

*DAYLIGHT_SAVING_TIME ............... USA66
*NONRESIDENT_TEMPLATES............... $SYSTEM.SYSTEM.TEMPLATE
*POWERFAIL_DELAY_TIME................ 30
*RESIDENT_TEMPLATES.................. $SYSTEM.SYSTEM.RTMPLATE
 SUPER_SUPER_IS_UNDENIABLE........... OFF
*SYSTEM_NAME......................... \EAST
*SYSTEM_NUMBER....................... 254
 SYSTEM_PROCESSOR_TYPE ............... NSR-W
*TIME_ZONE_OFFSET.................... -8:00
```

```
*TNSMISALIGN.......................... FAIL
*DESTINATION_CONTROL_TABLE............ SMALL
*TLE_LIMIT............................ 2000
*AUTO_RETRY_ON_ERROR_654.............. OFF
```

Attributes marked with an asterisk (*) in the INFO display are changed using the SCF ALTER command. Attributes without an asterisk are set in the CONFTEXT file during system generation. Other system attributes and how they are configured are listed in the *NonStop S-Series Planning and Configuration Guide*, the appropriate NonStop NS-series planning guide, or the *NonStop BladeSystem Planning Guide*. See the *DSM/SCM User's Guide* for more information on system generation and the CONFTEXT file.

Changing EMS Template Files

This procedure lets you specify the template object files for the Event Management Service (EMS) to use. If you perform normal system installation (running DSM/SCM on the target system), this procedure is not necessary.

NOTE: If you change the location of the EMS template files using this procedure, the INSTALL^TEMPLATES program permanently changes the location of the EMS template files. As a result, when you next run DSM/SCM, even though the Build and Apply creates new EMS templates, the subsequent system load invokes the EMS templates previously specified to the INSTALL^TEMPLATES program. To use the RTMPLATE and TEMPLATE EMS template files installed in the new SYS*nn* by DSM/SCM, you must use this procedure with the command:

```
-> ALTER, RESIDENT_TEMPLATES $SYSTEM.SYSTEM.RTEMPLATE, &
   NONRESIDENT_TEMPLATES $SYSTEM.SYSTEM.TMPLATE
```

1. As a precaution, save the current configuration file with an SCF SAVE command (documented in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*). For example, use this command to save the current configuration file in a file at the location $SYSTEM.ZSYSCONF.CONF0104:

   ```
   -> SAVE CONFIGURATION 1.4
   ```

2. View the current EMS template file names (shown here in bold type) with an INFO command:

   ```
   -> ASSUME SUBSYS $ZZKRN
   -> INFO

   NONSTOP KERNEL - Info SUBSYS  \EAST.$ZZKRN

   Current Settings

   *DAYLIGHT_SAVING_TIME ................ USA66
   *NONRESIDENT_TEMPLATES................ $SYSTEM.SYSTEM.TEMPLATE
   *POWERFAIL_DELAY_TIME................. 30
   *RESIDENT_TEMPLATES................... $SYSTEM.SYSTEM.RTMPLATE
    SUPER_SUPER_IS_UNDENIABLE............ OFF
   *SYSTEM_NAME.......................... \EAST
   *SYSTEM_NUMBER........................ 254
    SYSTEM_PROCESSOR_TYPE ............... NSR-W
   *TIME_ZONE_OFFSET..................... -8:00
   *TNSMISALIGN.......................... FAIL
   *DESTINATION_CONTROL_TABLE............ SMALL
   *TLE_LIMIT............................ 2000
   *AUTO_RETRY_ON_ERROR_654.............. OFF
   ```

NOTE: If $ZZKRN cannot find an EMS template file on $SYSTEM.SYSTEM, it then searches for the file on the current $SYSTEM.SYS*nn*.

3. Change the configuration, using the ALTER command to specify different files for the resident and nonresident EMS templates; for example:

   ```
   -> ALTER, RESIDENT_TEMPLATES $SYSTEM.TEMPLATE.TNEW, &
      NONRESIDENT_TEMPLATES $SYSTEM.TEMPLATE.NRTNEW
   ```

   These changes take effect immediately.

4. Confirm the changed EMS template file names with another INFO command.

If you need to reverse this change, repeat the ALTER command with the original values.

Changing the Power-Failure-to-Shutdown Time Interval

This procedure lets you specify the power-failure-to-shutdown time interval.

For NonStop S-series servers, the power-failure-to-shutdown time interval is the maximum time the system continues operation after the power has failed before entering a memory hold-up mode.

For Integrity NonStop NS-series servers and for Integrity NonStop BladeSystems, the power-failure-to-shutdown time interval is the length of time the system continues before powering off.

NonStop S-series servers recover automatically from a power failure if batteries are installed. The batteries maintain power to the processor memory only for as long as they can, usually 45 minutes. The actual amount of time the batteries can maintain memory depends on the system configuration, power-fail delay time, and the charge state of the batteries. If power is restored before the batteries are drained, the system begins processing at the point it was interrupted. However, if the power failure lasts long enough to drain the batteries, the system is shut down. At this point, when power is restored the system must be started by an operator.

On Integrity NonStop NS-series servers and Integrity NonStop BladeSystems, there is no automatic recovery after the system is powered off, even when battery capacity remains in the Uninterruptible Power Supply (UPS). Once AC power is restored, a manual restart of the system is necessary. If the system is configured to use a UPS, the system can be configured for a ride-through time before executing an orderly shut-down. For more information, see the POWERFAIL_DELAY_TIME attribute of the ALTER SUBSYS command (POWERFAIL_DELAY_TIME { n | RIDETHRUONLY }).

For detailed information about power-failure-to-shutdown time, see the *NonStop S-Series Planning and Configuration Guide*, the appropriate NonStop NS-series planning guide, or the *NonStop BladeSystem Planning Guide*.

1. As a precaution, save the current configuration file with an SCF SAVE command (documented in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*). For example, use this command to save the current configuration file in a file at the location $SYSTEM.ZSYSCONF.CONF0105:

   ```
   -> SAVE CONFIGURATION 1.5
   ```

2. View the current power failure time interval (shown here in bold type) with an INFO command:

   ```
   -> ASSUME SUBSYS $ZZKRN
   -> INFO

   NONSTOP KERNEL - Info SUBSYS  \EAST.$ZZKRN

   Current Settings

   *DAYLIGHT_SAVING_TIME ..............  USA66
   ```

```
*NONRESIDENT_TEMPLATES.............. $SYSTEM.SYSTEM.NRTNEW
*POWERFAIL_DELAY_TIME................ 30
*RESIDENT_TEMPLATES.................. $SYSTEM.SYSTEM.TNEW
 SUPER_SUPER_IS_UNDENIABLE........... OFF
*SYSTEM_NAME......................... \EAST
*SYSTEM_NUMBER....................... 254
 SYSTEM_PROCESSOR_TYPE .............. NSR-W
*TIME_ZONE_OFFSET.................... -8:00
*TNSMISALIGN......................... FAIL
*DESTINATION_CONTROL_TABLE........... SMALL
*TLE_LIMIT........................... 2000
*AUTO_RETRY_ON_ERROR_654............. OFF
```

3. Change the power failure time interval. This example uses the ALTER command to change the time interval to 40 seconds:

   ```
   -> ALTER, POWERFAIL_DELAY_TIME 40
   ```

   This change takes effect immediately.

4. Confirm the changed power failure time interval with another INFO command.

If you need to reverse this operation, repeat the ALTER command with the original value.

On Integrity NonStop NS-series servers and Integrity NonStop BladeSystems, you can specify RIDETHRUONLY instead of specifying a specific value. Specifying RIDETHRUONLY causes the operating system to wait indefinitely.

NOTE: Do not specify power failure delay time longer than the capacity of your uninterruptible power supply (UPS). When you specify a value for POWERFAIL_DELAY_TIME, you are controlling only the amount of time the operating system waits. You are not controlling how long the UPS lasts, nor are you limiting how much heat is generated. In determining an appropriate value, make sure you take into account the capacity of the UPS and the thermal attributes of the room. See the appropriate planning guide for more information.

Changing the System Name or System Number (G-Series RVUs)

On systems running G-series RVUs, changes to the system name and system number require a hard reset of the cold-load processor, followed by a system load.

To avoid introducing duplicate system names into the Network Routing Table (NRT) when you change the system name or number, read the section on managing the network in the *Expand Configuration and Management Manual*

CAUTION: Changing the system name or system number as documented in the *NonStop S-Series Hardware Installation and FastPath Guide* is intended to be used only when setting up a new NonStop S-series server.

Before changing the system name or system number on an existing system, contact your database administrator or service provider. To avoid losing data, you must modify database catalogs and file labels if you are running any of these complex applications:

- DSM/SCM
- DSM/Tape Catalog (MEDIACOM)
- Enscribe
- ODBC/NOS
- Open System Services
- RDF
- ServerNet/FX
- HP Storage Management Facility (SMF)
- HP NonStop SQL/MP
- HP NonStop Transaction Management Facility (TM)

Other applications, as well as application control files such as the Pathway PATHCTL file, might also be affected.

Changing the System Name or System Number (H-Series RVUs)

This section describes the steps you must take to change the system name of an existing H-series system. While similar to the procedure used with G-series systems, specifying a system name for an H-series system requires additional steps because the H-series system name is used in additional ways:

- Maintenance Entities (MEs) in ServerNet switch boards use the system name to register with the Domain Name Server (DNS) Server.
- ME firmware and HSS and STD millicode use the system name to perform connectivity checks between different ServerNet Switch Boards and between the Processor ServerNet Switch Boards and the processors

A system name change requires system power cycle. It might also require reconfiguration of some of the software running on the NonStop system. To avoid introducing duplicate system names into Network Routing Tables or into the Domain Name Server, make sure the assigned

name is unique in your network environment and follow the guidelines in the *Expand Configuration and Management Manual*.

CAUTION: Changing the system name or system number of an existing system may require changes in your software application configuration and labeling. Some of these applications are:

- DSM/SCM
- DSM/Tape Catalog (MEDIACOM)
- Enscribe databases
- ODBC/NOS
- Open System Services
- RDF
- ServerNet/FX
- HP Storage Management Facility (SMF)
- HP NonStop SQL/MP
- HP NonStop Transaction Management Facility (TM)
- Application configuration files
- SCFCSTM file

Other applications, as well as application control files such as the Pathway PATHCTL file, might also be affected and require a change in configuration. Please contact your database administrator or service provider or software application manual before making system name change.

System Number Change

For information about changing the system number, see the *Expand Configuration and Management Manual*.

System Name Change Procedure (NonStop Not Running)

Load the system using OSM Low Level Link (OSM LLL) and follow the procedure in the next subsection.

System Name Change Procedure (NonStop Running)

CAUTION: This procedure should be performed only by, or under direction of, your service provider, in conjunction with your database administrator.

Use SCF to change the actual system name:

1. Use the SCF ALTER command to change the system name attribute:

```
->ASSUME SUBSYS $ZZKRN
->ALTER, system_name, new_system_name
```

2. Use the SCF ABORT command to stop OSM Service application:

```
->ABORT PROCESS $ZZKRN.#OSM-CIMOM
```

3. Use TACL PURGE commands to purge OSM Configuration files:

```
>PURGE $SYSTEM.ZSERVICE.IAREPO
>PURGE $SYSTEM.ZSERVICE.SUPPREPO
```

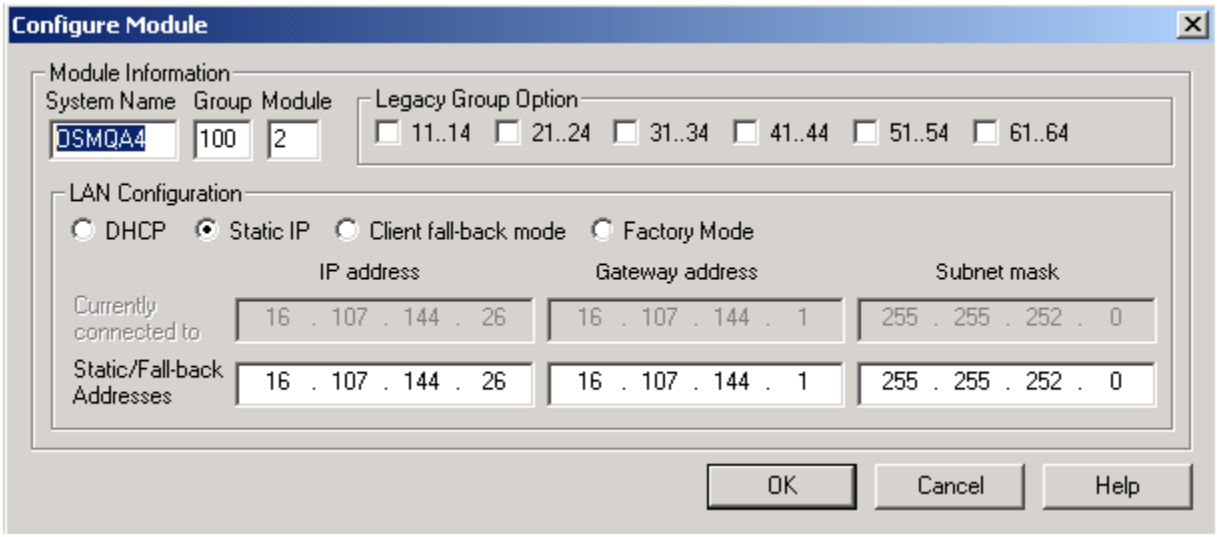
4. Change and reconfigure any software application as needed.

Use the OSM Low-Level Link to change all MEs to the new system name:

1. Log on to the OSM Low-Level Link.
2. Click the Processor Status button. In the Processor Status dialog box, perform the Halt action on all processors

3. Click System Discovery, expand the tree pane to select one of the I/O adapter module (IOAM) objects (located within all group objects numbered >100 and <200).
4. Perform the Configure Module action. In the Configure Module dialog box, change the System Name to match the new name assigned in SCF, then click OK.

NOTE: The LAN Configuration setting must be something other than Client fall-back mode in order to change the System Name.



5. A dialog box informs you that a reset is required to make the action take effect (which will result in loss of the current OSM Low-Level Link session), and will give you the option of performing the action. Click OK.
6. Log on to the OSM Low-Level Link (log on through the same p-switch ME every time during this procedure). Repeat steps 3 through 5 to change the system name for each I/O adapter module (IOAM) in your system; then proceed to Step 7.

NOTE: As you log on to the OSM Low-Level Link to change the system name for the various IOAM and p-switch modules, you continue to log on using the old system name until Step 12 of this procedure.

The modules you perform the system name change procedure on will not appear in the OSM Low-Level Link tree view during subsequent logons, until you complete the process and log on once more (Step 17).

7. Log on to the OSM Low-Level Link and click System Discovery.
8. Expand the tree pane and Group 100 object and select the p-switch module that you are not currently logged on to.

NOTE: P-switch modules are GRP-100.MOD-2 and GRP-100.MOD-3. If you are logged on to GRP-100.MOD-2, select GRP-100.MOD-3

If you are using the System List option to log on, you can determine the IP address of the p-switch ME you are logged on to by selecting Network Settings from the File menu of the OSM LLL menu bar.

9. With the p-switch module selected, repeat Step 4 and Step 5 (changing the system name in the Configure Module dialog box); then proceed to Step 10.
10. Log on to the OSM Low-Level Link and click System Discovery.

11. Expand the tree pane and Group 100 object and select the p-switch module that you are logged on to (it should be the only one visible at this point).
12. With the p-switch module selected, repeat Step 4 and Step 5; then proceed to Step 13.
13. Log on to the OSM Low-Level Link. This time, depending on the log on method used:
    a. If you log on using the System List, you will still see the old system name (and not the new name). Select the old name and log on as usual. During logon, a dialog box will inform you that the system name has been changed. In subsequent logon attempts, you will see the new system name in the System List (and not the old system name).
    b. If you log on using the Host Name option, you must now enter the new host name for the p-switch.
14. Click System Discovery, then perform the Power Cycle All Processors action on the Group 400 object.
15. Log off the OSM Low-Level Link and wait five minutes
16. Log on to the OSM Low-Level Link. All enclosures, including processors, should be discovered. Load the system.
17. Reconfigure and start any software applications that were modified (as discussed in Overview/Preparation section).
18. Configure the system for dial-out using OSM Notification Director.

Changing the System Name or System Number (J-Series RVUs)

On systems running J-series RVUs, changes to the system name require a shut down of the system and a subsequent system load.

CAUTION: Changing the system name or system number of an existing system may require changes in your software application configuration and labeling. Some of these applications are:

- DSM/SCM
- DSM/Tape Catalog (MEDIACOM)
- Enscribe databases
- ODBC/NOS
- Open System Services
- RDF
- ServerNet/FX
- HP Storage Management Facility (SMF)
- HP NonStop SQL/MP
- HP NonStop Transaction Management Facility (TM)
- Application configuration files
- SCFCSTM file

Other applications, as well as application control files such as the Pathway PATHCTL file, might also be affected and require a change in configuration. Please contact your database administrator or service provider or software application manual before making system name change.

System Number Change

For information about changing the system number, see the *Expand Configuration and Management Manual*.

System Name Change Service Procedures

For complete instructions on changing the system name of a J-series system, refer to the appropriate service procedure:

- *Changing the System Name of a NonStop BladeSystem*
- *Changing the System Name of a NonStop NS2000 Series System*

Changing the System Time Attributes

Changes to them require a system load to take effect.

To reconfigure the DAYLIGHT_SAVING_TIME value, the TIME_ZONE_OFFSET value, or both values, use this procedure.

1. As a precaution, save the current configuration file with an SCF SAVE command (documented in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*). For example, use this command to save the current configuration file in a file at the location $SYSTEM.ZSYSCONF.CONF0106:

   ```
   -> SAVE CONFIGURATION 1.6
   ```

2. View the current settings for the system time attributes (shown below in bold type) with an SCF INFO command. The INFO command displays both the current settings and the changed values of the system name and number attributes, which take effect at the next system load.

   ```
   -> ASSUME SUBSYS $ZZKRN
   -> INFO

   NONSTOP KERNEL - Info SUBSYS  \NONAME.$ZZKRN

   Current Settings

   *DAYLIGHT_SAVING_TIME ..............  NONE
   *NONRESIDENT_TEMPLATES..............  $SYSTEM.SYSTEM.TEMPLATE
   *POWERFAIL_DELAY_TIME...............  40
   *RESIDENT_TEMPLATES.................  $SYSTEM.SYSTEM.RTMPLATE
    SUPER_SUPER_IS_UNDENIABLE..........  OFF
   *SYSTEM_NAME........................  \NONAME
   *SYSTEM_NUMBER......................  44
    SYSTEM_PROCESSOR_TYPE .............  NSR-W
   *TIME_ZONE_OFFSET...................  0:00
   *TNSMISALIGN........................  FAIL
   *DESTINATION_CONTROL_TABLE..........  SMALL
   *TLE_LIMIT..........................  2000
   *AUTO_RETRY_ON_ERROR_654............  OFF

   Pending Changes will take effect at next Manual Reload or Hard Reset of the system

   *SYSTEM_NAME........................  \EAST
   *SYSTEM_NUMBER......................  44
   ```

3. You can use the SCF ALTER command to change either of these attributes; for example:

   ```
   -> ALTER, DAYLIGHT_SAVING_TIME USA66
   -> ALTER, TIME_ZONE_OFFSET -5:00
   ```

   If you are changing both attributes, you can more efficiently use system resources (because these attributes are stored in a SEEPROM in the NonStop S-series server backplane) by grouping them into one command rather than by entering each separately; for example:

   ```
   -> ALTER, DAYLIGHT_SAVING_TIME USA66, TIME_ZONE_OFFSET -5:00
   ```

4. Confirm the change with another INFO command (shown below in bold type). The INFO command displays both the current value and the changed (pending) value, which takes effect at the next system load.

   ```
   -> INFO

   NONSTOP KERNEL - Info SUBSYS  \NONAME.$ZZKRN

   Current Settings

   *DAYLIGHT_SAVING_TIME ..............  NONE
   *NONRESIDENT_TEMPLATES..............  $SYSTEM.SYSTEM.TEMPLATE
   *POWERFAIL_DELAY_TIME...............  40
   *RESIDENT_TEMPLATES.................  $SYSTEM.SYSTEM.RTMPLATE
    SUPER_SUPER_IS_UNDENIABLE..........  OFF
   *SYSTEM_NAME........................  \NONAME
   *SYSTEM_NUMBER......................  254
   ```

```
 SYSTEM_PROCESSOR_TYPE ............... NSR-W
*TIME_ZONE_OFFSET..................... 0:00
*TNSMISALIGN.......................... FAIL
*DESTINATION_CONTROL_TABLE............ SMALL
*TLE_LIMIT............................ 2000
*AUTO_RETRY_ON_ERROR_654.............. OFF

Pending Changes will take effect at next Manual Reload or Hard Reset of the system

*DAYLIGHT_SAVING_TIME ................ USA66
*SYSTEM_NAME.......................... \EAST
*SYSTEM_NUMBER........................ 44
*TIME_ZONE_OFFSET..................... -05:00
```

5. Halt the system from the OSM or TSM Low-Level Link.
6. Reload the system using the Start System dialog box from the OSM or TSM Low-Level Link. This action loads the millicode and puts the changes into effect.

NOTE: TSM is not supported in H-series and J-series systems.

7. If you changed the TIME_ZONE_OFFSET value, enter a TACL SETTIME command to compensate for the change in the system time, for example:

   ```
   > SETTIME MARCH 7 2000, 8:10
   ```

   If you have not reloaded the system and you need to reverse this change, repeat the ALTER command with the original values. Then enter another SETTIME command (if needed).

   If you have reloaded the system and you need to reverse this change, repeat the ALTER command with the original values. Then use the OSM or TSM Low-Level Link to reset and reload the system to restore the original values. Finally, enter another SETTIME command (if needed).

Changing Data Misalignment Attribute

TNS programs and accelerated TNS programs must follow the data alignment rules of the TNS architecture, which require that all non-byte data must begin and end at even-byte memory boundaries for correct execution. TNS compilers automatically ensure this for compiler-managed variables. But odd-byte misalignments can occur if the programmer uses incorrect pointer conversions on byte arrays that contain non-byte data. These errors can escape detection at compile time and can silently cause data corruption at run time.

The TNSMISALIGN attribute controls how TNS and accelerated TNS programs behave when odd-byte data misalignments occur. This applies to all TNS and accelerated TNS programs on the system; it has no effect on native-mode programs.

Recommendation for Production Systems

To avoid any change in the behavior of existing TNS programs on a production system, leave the TNSMISALIGN attribute at its default setting, FAIL.

This attribute is maintained system wide; the settings affect all the processors in the node.

Changing Attribute on a Nonproduction System

To change these data misalignment values on a nonproduction system:

1. View the current data misalignment attribute values (shown here in bold type):

```
-> INFO SUBSYS $ZZKRN

NONSTOP KERNEL - Info SUBSYS  \EAST.$ZZKRN

Current Settings

*DAYLIGHT_SAVING_TIME ............... USA66
*NONRESIDENT_TEMPLATES............... $SYSTEM.SYSTEM.TEMPLATE
*POWERFAIL_DELAY_TIME................ 30
*RESIDENT_TEMPLATES.................. $SYSTEM.SYSTEM.RTMPLATE
 SUPER_SUPER_IS_UNDENIABLE........... OFF
*SYSTEM_NAME......................... \EAST
*SYSTEM_NUMBER....................... 254
 SYSTEM_PROCESSOR_TYPE ............... NSR-W
*TIME_ZONE_OFFSET.................... -8:00
*TNSMISALIGN......................... FAIL
*DESTINATION_CONTROL_TABLE........... SMALL
*TLE_LIMIT........................... 2000
*AUTO_RETRY_ON_ERROR_654............. OFF
```

2. Change one or more data misalignment attribute values, for example:

```
-> ALTER SUBSYS $ZZKRN, TNSMISALIGN NOROUND.
```

TNSMISALIGN NOROUND	If a data misalignment occurs, address used by the native process will not be rounded off

   This change takes effect immediately.

3. Confirm the change:

```
-> INFO SUBSYS $ZZKRN

NONSTOP KERNEL - Info SUBSYS  \EAST.$ZZKRN

Current Settings

*DAYLIGHT_SAVING_TIME ............... USA66
*NONRESIDENT_TEMPLATES............... $SYSTEM.SYSTEM.TEMPLATE
*POWERFAIL_DELAY_TIME................ 30
*RESIDENT_TEMPLATES.................. $SYSTEM.SYSTEM.RTMPLATE
 SUPER_SUPER_IS_UNDENIABLE........... OFF
*SYSTEM_NAME......................... \EAST
*SYSTEM_NUMBER....................... 254
 SYSTEM_PROCESSOR_TYPE ............... NSR-W
*TIME_ZONE_OFFSET.................... -8:00
*TNSMISALIGN......................... NOROUND
*DESTINATION_CONTROL_TABLE........... SMALL
*TLE_LIMIT........................... 2000*
*AUTO_RETRY_ON_ERROR_654............. OFF
```

If you need to reverse this operation, repeat the ALTER command with the original values under "View the current data misalignment attribute."

Changing the Destination Control Table Size Limit

This procedure lets you specify the size limit of the Destination Control Table.

For more information about the DESTINATION_CONTROL_TABLE attribute of the ALTER SUBSYS command, see "DESTINATION_CONTROL_TABLE" [p. 102].

1. As a precaution, save the current configuration file with an SCF SAVE command (documented in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*.

   For example, use this command to save the current configuration file in a file at the location $SYSTEM.ZSYSCONF.CONF0106:

   ```
   -> SAVE CONFIGURATION 1.6
   ```

2. View the current settings for the attribute TLE Limit with an SCF INFO command:

   ```
   -> ASSUME SUBSYS $ZZKRN
   -> INFO

   NONSTOP KERNEL - Info SUBSYS \EAST.$ZZKRN

   Current Settings
   *DAYLIGHT_SAVING_TIME ............... USA66
   *NONRESIDENT_TEMPLATES............... $SYSTEM.SYSTEM.NRTNEW
   *POWERFAIL_DELAY_TIME................ 30
   *RESIDENT_TEMPLATES.................. $SYSTEM.SYSTEM.TNEW
    SUPER_SUPER_IS_UNDENIABLE........... OFF
   *SYSTEM_NAME......................... \EAST
   *SYSTEM_NUMBER....................... 254
    SYSTEM_PROCESSOR_TYPE ............... NSR-W
   *TIME_ZONE_OFFSET.................... -8:00
   *TNSMISALIGN......................... FAIL
   *DESTINATION_CONTROL_TABLE........... SMALL
   *TLE_LIMIT........................... 2000
   *AUTO_RETRY_ON_ERROR_654............. OFF
   ```

3. Change the DESTINATION_CONTROL_TABLE attribute using ALTER command. For example, to change the limit to LARGE:

   ```
   -> ALTER, DESTINATION_CONTROL_TABLE MEDIUM
   ```

   This change takes effect immediately.
4. Confirm the change with another INFO command. For example:

   ```
   -> INFO
   ```

   If you need to reverse this operation, repeat the ALTER command using the original value.

Changing the System TLE Limit Attribute

This procedure lets you specify the TLE limit attribute. As of the J06.09, H06.20, and G06.32.01 RVUs, you can configure a limit on the number of TLEs (Time List Elements) that can be allocated by a process. For previous G, H, and J-series RVUs, there was no such limit and an errant application could consume all, or nearly all, the TLEs.

The default TLE limit is the maximum number of TLEs per CPU:

- For G-series systems, the maximum number of TLEs per CPU is 3600.
- For H and J-series systems, the maximum number of TLEs per CPU is 20000.

For more information about the TLE_LIMIT attribute of the ALTER SUBSYS command, see TLE_LIMIT [p. 105].

1. As a precaution, save the current configuration file with an SCF SAVE command (documented in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*.

   For example, use this command to save the current configuration file in a file at the location $SYSTEM.ZSYSCONF.CONF0106:

   ```
   -> SAVE CONFIGURATION 1.6
   ```

2. View the current settings for the attribute TLE Limit with an SCF INFO command:

```
-> ASSUME SUBSYS $ZZKRN
-> INFO

NONSTOP KERNEL - Info SUBSYS \EAST.$ZZKRN

Current Settings
*DAYLIGHT_SAVING_TIME ............... USA66
*NONRESIDENT_TEMPLATES............... $SYSTEM.SYSTEM.NRTNEW
*POWERFAIL_DELAY_TIME................ 30
*RESIDENT_TEMPLATES.................. $SYSTEM.SYSTEM.TNEW
 SUPER_SUPER_IS_UNDENIABLE........... OFF
*SYSTEM_NAME......................... \EAST
*SYSTEM_NUMBER....................... 254
 SYSTEM_PROCESSOR_TYPE ............... NSR-W
*TIME_ZONE_OFFSET.................... -8:00
*TNSMISALIGN......................... FAIL
*DESTINATION_CONTROL_TABLE........... SMALL
*TLE_LIMIT........................... 2000
*AUTO_RETRY_ON_ERROR_654............. OFF
```

3. Change the TLE Limit attribute using ALTER command. For example, to change the TLE limit to 3000:

```
-> ALTER, TLE_LIMIT 3000
```

   This change takes effect immediately.
4. Confirm the change with another INFO command. For example:

```
-> INFO
```

   If you need to reverse this operation, repeat the ALTER command using the original value.

Changing Software Data Integrity Checking

Beginning with the H06.20 and J06.09 RVUs, the system can detect situations in which a message request buffer is modified due to a programming error or inadvertent data corruption. Depending on the value of the AUTO_RETRY_ON_ERROR_654 attribute, the system can either immediately report file system error 654 or retry sending the request up to three times:

- A value of ON causes the system to retry the request up to three times.
- A value of OFF causes the system to immediately report file-system error 654 without performing retries.

For more information, see AUTO_RETRY_ON_ERROR_654 [p. 105].

To turn on auto retries:

1. As a precaution, save the current configuration file with an SCF SAVE command (documented in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*.

   For example, use this command to save the current configuration file in a file at the location $SYSTEM.ZSYSCONF.CONF0106:

```
-> SAVE CONFIGURATION 1.6
```

2. View the current settings for the AUTO_RETRY_ON_ERROR_654 attribute with an SCF INFO command:

```
-> ASSUME SUBSYS $ZZKRN
-> INFO

NONSTOP KERNEL - Info SUBSYS \EAST.$ZZKRN

Current Settings
*DAYLIGHT_SAVING_TIME ............... USA66
```

```
*NONRESIDENT_TEMPLATES.............. $SYSTEM.SYSTEM.NRTNEW
*POWERFAIL_DELAY_TIME................ 30
*RESIDENT_TEMPLATES.................. $SYSTEM.SYSTEM.TNEW
 SUPER_SUPER_IS_UNDENIABLE........... OFF
*SYSTEM_NAME......................... \EAST
*SYSTEM_NUMBER....................... 254
 SYSTEM_PROCESSOR_TYPE .............. NSR-W
*TIME_ZONE_OFFSET.................... -8:00
*TNSMISALIGN......................... FAIL
*DESTINATION_CONTROL_TABLE........... SMALL
*TLE_LIMIT........................... 2000
*AUTO_RETRY_ON_ERROR_654............. OFF
```

3. To enable automatic retries, change the AUTO_RETRY_ON_ERROR_654 attribute using ALTER command. For example:

   ```
   -> ALTER, AUTO_RETRY_ON_ERROR_654 ON
   ```

   This change takes effect immediately.
4. Confirm the change with another INFO command. For example:

   ```
   -> INFO
   ```

   If you need to reverse this operation, repeat the ALTER command using the original value.

3 Configuring and Managing Generic Processes

This chapter describes how to configure and manage a generic process. A generic process can be a copy of a program, a program written by a third-party supplier, or a user-written program that can be started by an explicit or implicit RUN command.

Definition of a Generic Process

A process is a running entity that is managed by the operating system, as opposed to a program, which is a collection of code and data.

An I/O process (IOP) is a specialized process that performs input and output transfer from one device to another; for example, between a ServerNet addressable controller and a specific device type. Other characteristics of an I/O process are:

- An I/O process usually uses privileged code (processes with a nonzero device type).
- An I/O process manages communication with I/O devices, such as disks, printers, and communication lines.
- An I/O process pair logically owns one or more I/O devices or communication lines.

A generic process is a device-type 0 process that has fewer limitations than an IOP. HP, a third-party company, or any user can configure a generic process to start in one or more processors. A generic process:

- Is configured by entering an ADD command that specifies attributes to customize the process.
- Is started and maintained by the $ZPM persistence manager.
- Once it is started, has its persistence managed by the Kernel subsystem; hence, generic processes are sometimes called system-managed processes.

You can create a generic process to replace any system process that can be started from TACL and that does not require a PARAM or ASSIGN.

Examples of generic processes created by HP are:

- Subsystem manager processes like $ZZKRN, $ZZSTO, $ZZLAN, and $ZZWAN.
- The $ZHOME process, which allows user processes to survive errors received from the $YMIOP system console (for example, error 140). This feature is described in the *NonStop S-Series Planning and Configuration Guide*, the *NonStop S-Series Operations Guide*, or the *NonStop Operations Guide* (for J- and H-series RVUs). See the INFO display of the $ZHOME process.

The commands for control and management of generic processes are described in Chapter 6: SCF Commands for the Kernel Subsystem (page 83). They are:

Control commands:	ABORT, ADD, ALTER, DELETE, and START
Management commands:	INFO, NAMES, STATUS, and VERSION

Characteristics of a Generic Process

Generic processes have these characteristics:

- You can configure a generic process like a Pathway server class, to start in one processor, more than one processor, or in each processor in the system. See "Controlling Where a Generic Process Starts" (page 52).
- You can control when a generic process should start and, optionally, restart. See "Controlling When a Generic Process Starts" (page 52).
- You can permanently change an attribute of a generic process by using the SCF ALTER command. This change takes effect the next time the process is started. See "Altering a Generic Process" (page 64).

- You can create or alter a generic process to run at a high PIN or a low PIN (see HIGHPIN). The default is to run at a high PIN.
- You use the SCF ADD and ALTER commands to create or change the behavior of a generic process. You cannot use TACL ASSIGNs, DEFINEs, or PARAMs on a generic process.
- When you add a generic process to the system, the $ZPM persistence manager reserves the generic process name in the $SYSTEM.ZSYSCONF.CONFIG system configuration database and in the destination control table.
- The default user ID for a generic process is the same as the user ID for the SCF session in which it was created. If you are the super ID (255,255) and want the process to be run by a specific user, you can specify that user ID (using the USERID attribute rather than using the default value.

Uses for a Generic Process

You might create a generic process if you want a process that:

- Is automatically started up by a means other than an SCF command file.
- Acts as a subsystem manager process, rather than a D-series program started by a RUN command.
- Tracks outages by writing a record to a database at regular intervals.
- Runs in multiple processors in the system rather than just a pair of I/O processes (like LINKMON and QIOMON).
- Increases performance by using a database-in-memory-type process (rather than a database-on-disk-type process).
- Creates a TACL macro that is run every time a processor is reloaded (for example, to automatically correct LAN settings that were overwritten by the processor reload).
- Is persistent; that is, restarts automatically if it stops abnormally.

Examples of Generic Processes

Here are examples of generic processes that you, as a user, can create:

1. You can create a data-collecting generic process that runs only when a specific processor is reloaded by specifying that processor as the primary processor and specifying its STARTMODE to be APPLICATION. For example:

```
-> ASSUME PROCESS $ZZKRN
-> ADD PROCESS $ZZKRN.#CPU3-PROCESS,  &
       NAME $CPU3,                    &
       PROGRAM $SYSTEM.SYSTEM.NULL,   &
       AUTORESTART 0,                 &
       HOMETERM $ZHOME,               &
       CPU 3,                         &
       STARTMODE APPLICATION
```

   If you do not enter a START command now, the process is not started until and unless processor 3 is reloaded. When the program has finished collecting its data and terminates in an orderly fashion, it is not restarted (because AUTORESTART is 0) until and unless processor 3 is reloaded again.

```
-> INFO #CPU3-PROCESS, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#CPU3-PROCESS

 *AutoRestart..............0
 *BackupCPU................1
 *CPU......................3
 *DefaultVolume............$SYSTEM.NOSUBVOL
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$ZHOME
```

```
 *InFile...................Not Specified
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$CPU3
 *OutFile..................Not Specified
 *PFSSize..................Not Specified
 *PrimaryCPU...............0
 *Priority.................148
 *Program..................$SYSTEM.SYSTEM.NULL
 *SaveAbend................OFF
 *StartMode................APPLICATION
 *StartupMessage...........Not Specified
 *StopMode.................Standard
 *Type.....................OTHER
 *UserId...................SUPER.SUPER ( 255,255 )
```

2. You can create your own Subsystem Control Point (SCP) process as a generic process. For example:

```
-> ASSUME PROCESS $ZZKRN
-> ADD PROCESS $ZZKRN.#SCP,          &
       NAME $ZNET, PRIORITY 175,     &
       PROGRAM $SYSTEM.SYSTEM.SCP,   &
       PRIMARYCPU 0, BACKUPCPU 1,    &
       AUTORESTART 10, TYPE OTHER,   &
       STARTMODE SYSTEM,             &
       HOMETERM $ZHOME,              &
       OUTFILE $ZHOME,               &
       STARTUPMSG "<BCKP-CPU>",      &
       STOPMODE STANDARD
-> START #SCP
-> INFO #SCP, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#SCP

 *AutoRestart..............10
 *BackupCPU................1
 *CPU......................Not Specified
 *DefaultVolume............$SYSTEM.NOSUBVOL
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$ZHOME
 *InFile...................Not Specified
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$ZNET
 *OutFile..................ZHOME
 *PFSSize..................Not Specified
 *PrimaryCPU...............0
 *Priority.................175
 *Program..................$SYSTEM.SYSTEM.SCP
 *SaveAbend................OFF
 *StartMode................SYSTEM
 *StartupMessage...........<BCKP-CPU>
 *StopMode.................Standard
 *Type.....................OTHER
 *UserId...................SUPER.SUPER ( 255,255 )
```

3. You can create your own Expand monitor process as a generic process. For example:

```
-> ASSUME PROCESS $ZZKRN
-> ADD PROCESS $ZZKRN.#ZEXP,         &
       NAME $ZEXP, PRIORITY 180,     &
       PROGRAM $SYSTEM.SYSTEM.OZEXP, &
       PRIMARYCPU 0,                 &
       BACKUPCPU 1,                  &
```

```
       AUTORESTART 10, TYPE OTHER,   &
       HOMETERM $ZHOME,              &
       OUTFILE $ZHOME,               &
       STARTMODE SYSTEM,             &
       STARTUPMSG "<BCKP-CPU>"
-> START #ZEXP
-> INFO #ZEXP, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#ZEXP

 *AutoRestart..............10
 *BackupCPU................1
 *CPU......................Not Specified
 *DefaultVolume............$SYSTEM.NOSUBVOL
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$ZHOME
 *InFile...................Not Specified
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$ZEXP
 *OutFile..................$ZHOME
 *PFSSize..................Not Specified
 *PrimaryCPU...............0
 *Priority.................180
 *Program..................$SYSTEM.SYSTEM.OZEXP
 *SaveAbend................OFF
 *StartMode................SYSTEM
 *StartupMessage...........<BCKP-CPU>
 *StopMode.................Standard
 *Type.....................OTHER
 *UserId...................SUPER.SUPER ( 255,255 )
```

4. You can create a SPOOLCOM generic process if you want an operator to run SPOOLCOM but not have access to TACL. To create a SPOOLCOM generic process, you enter an ADD command that specifies the SPOOLCOM program file along with the other attributes you want to specify. For example:

```
-> ASSUME PROCESS $ZZKRN
-> ADD #PTR-OP,                        &
       NAME $PTROP,                    &
       PROGRAM $SYSTEM.SYSTEM.SPOOLCOM, &
       AUTORESTART 2,                  &
       HOMETERM $TERM.#T1,             &
       INFILE $TERM.#T1,               &
       OUTFILE $TERM.#T1,              &
       CPU 2,                          &
       STARTMODE APPLICATION
-> START #PTR-OP
-> INFO #PTR-OP, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#PTR-OP

 *AutoRestart..............2
 *BackupCPU................Not Specified
 *CPU......................02
 *DefaultVolume............$SYSTEM.NOSUBVOL
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$TERM.#T1
 *InFile...................$TERM.#T1
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$PTROP
 *OutFile..................$TERM.#T1
```

```
 *PFSSize..................Not Specified
 *PrimaryCPU...............Not Specified
 *Priority.................169
 *Program..................$SYSTEM.SYSTEM.SPOOLCOM
 *SaveAbend................OFF
 *StartMode................APPLICATION
 *StartupMessage...........Not Specified
 *StopMode.................Standard
 *Type.....................OTHER
 *UserId...................SUPER.SUPER ( 255,255 )
```

5. You can create a TACL generic process with a symbolic name that describes the person using it. If the TACL generic process stops, the operator need not know the true terminal name. Instead the operator can restart the TACL generic process by entering the person's name at the TACL prompt. For example:

```
-> ASSUME PROCESS $ZZKRN
-> ADD #PNAME,                              &
       NAME $PNAME,                         &
       PROGRAM $SYSTEM.SYSTEM.TACL,         &
       HOMETERM $TERM.#T1,                  &
       INFILE $TERM.#T1,                    &
       OUTFILE $TERM.#T1,                   &
       CPU 3,                               &
       STARTMODE MANUAL
-> START #PNAME
-> INFO #PNAME, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#PNAME

 *AutoRestart..............0
 *BackupCPU................Not Specified
 *CPU......................03
 *DefaultVolume............$SYSTEM.NOSUBVOL
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$TERM.#T1
 *InFile...................$TERM.#T1
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$PNAME
 *OutFile..................$TERM.#T1
 *PFSSize..................Not Specified
 *PrimaryCPU...............Not Specified
 *Priority.................169
 *Program..................$SYSTEM.SYSTEM.TACL
 *SaveAbend................OFF
 *StartMode................MANUAL
 *StartupMessage...........Not Specified
 *StopMode.................Standard
 *Type.....................OTHER
 *UserId...................SUPER.SUPER ( 255,255 )
```

6. You can also create the TACL generic process in the preceding example to be persistent (by adding the AUTORESTART attribute). For example:

```
-> ASSUME PROCESS $ZZKRN
-> ADD #PNAME,                             &
       NAME $PNAME,                        &
       PROGRAM $SYSTEM.SYSTEM.TACL,        &
       AUTORESTART 2,                      &
       HOMETERM $TERM.#T1,                 &
       INFILE $TERM.#T1,                   &
       OUTFILE $TERM.#T1,                  &
       CPU 1,                              &
       STARTMODE MANUAL
```

```
-> START #PNAME
-> INFO #PNAME, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#PNAME
 *AutoRestart..............2
 *BackupCPU................Not Specified
 *CPU......................01
 *DefaultVolume............$SYSTEM.NOSUBVOL
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$TERM.#T1
 *InFile...................$TERM.#T1
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$PNAME
 *OutFile..................$TERM.#T1
 *PFSSize..................Not Specified
 *PrimaryCPU...............Not Specified
 *Priority.................169
 *Program..................$SYSTEM.SYSTEM.TACL
 *SaveAbend................OFF
 *StartMode................MANUAL
 *StartupMessage...........Not Specified
 *StopMode.................Standard
 *Type.....................OTHER
 *UserId...................SUPER.SUPER ( 255,255 )
```

Controlling Where a Generic Process Starts

Specifying the processors in which a generic process starts depends on the configuration of the CPU, PRIMARYCPU, and BACKUPCPU attributes for the generic process. Table 3-1 describes where the $ZPM persistence manager can start the process.

Table 3-1 Controlling Where a Generic Process Starts

If the processor configuration is…	The generic process starts in…
CPU *n*	The designated processor, when it is available.
CPU ALL	All processors that are up; the remaining processors when they come up.
CPU (*n, n1, …*)	All designated processors that are up; the remaining designated processors when they come up.
CPU FIRST	The first available processor in the system.
CPU FIRSTOF (*n, n1, …*)	The first available processor in the designated group.
PRIMARYCPU with BACKUPCPU	The first available processor in the pair (unless you configure it to start in a designated processor, as described under "Starting in a Designated Processor" (page 60)).
PRIMARYCPU without BACKUPCPU	The designated primary processor when it comes up.

Controlling When a Generic Process Starts

G-series systems use a phased load process, a sequence of steps for system load or processor reload. This allows the operating system to start its subsystems in an automated and orderly manner. The STARTMODE attribute of each subsystem manager (and of other generic processes) determines when the $ZPM persistence manager starts the process, as described in Table 3-2.

Table 3-2 Start Modes for Generic Processes

If the value of the STARTMODE attribute is…	The Generic Process…
KERNEL	Is started early during system load.
SYSTEM	Is started during the final stage of system load.
APPLICATION	Is started after system load is finished.
MANUAL	Can be started by the user any time after system load is finished.
DISABLED	Cannot be started unless the user changes the STARTMODE attribute to one of the preceding values.

Table 3-3 describes when a generic process can start.

Table 3-3 Controlling When a Generic Process Starts

Condition or Event	Determining Attribute	See Page
Start mode	STARTMODE	53
Persistence	AUTORESTART	54
System load (and initial processor reload)	STARTMODE	55
Later processor reload	Object state and STARTMODE	56
Abnormal event	Object state	56

Start Mode Considerations

When specifying a STARTMODE attribute value for a generic process, consider:

- The start mode of a generic process governs the startup of a generic process as part of system load (and initial reload of the processor in which the generic process is configured). This is in contrast to the persistence of a generic process, which governs restart after system load.
- At system load or processor reload, generic processes are started when they are configured to do so. You specify the start mode of a generic process when you use the STARTMODE attribute of the "ADD Command (Sensitive Command)" (page 86) or "ALTER Command (Sensitive Command)" (page 95). The STARTMODE attribute lets you specify when a generic process should be started, as listed in Table 3-4.

Table 3-4 STARTMODE Attribute Values for Generic Processes

STARTMODE	Description
KERNEL	The process is among the first started during system load, usually because it must be running before other processes are created. $ZPM starts subsystem manager generic processes and Kernel service processes such as QIOMON at this time.
SYSTEM	$ZPM starts the process as the final stage of the system load.
APPLICATION	$ZPM starts the process after the system load is finished and the CIIN file has been read and executed. Only then does $ZPM start user-configured generic processes. Processes such as $ZTSM (for TSM) and $ZOSM and $ZCMOM (for OSM) are started at this time.
MANUAL	The user can start a generic process any time after system load is finished. You use this start mode for a generic process that depends on an application that starts after system load.
DISABLED	The process cannot be started unless you change its STARTMODE attribute to one of the preceding values. This is the default. You can configure a generic process with this value if you want to ensure that it is not started inadvertently through SCF.

- The STARTMODE value takes effect at the next system load or processor reload. For example, if you configure a process with a STARTMODE value of SYSTEM, it is started during the SYSTEM phase when the system is next loaded or when the processor in which the process is configured is next reloaded.

  When the system is loaded, a generic process is automatically started at the first reload of its processor or processors unless its start mode is MANUAL or DISABLED. Or you can use the START command on a process with any start mode except DISABLED.

- When configuring the STARTMODE attribute of a generic process, it is important to understand that a generic process started in a specific phase must be able to resolve dependencies on resources not yet available. This understanding is especially important with regard to generic processes configured to start in the KERNEL or SYSTEM phase of the system load or processor reload.

  For example, the storage subsystem is initialized during the KERNEL phase. Disks supported by a process in the loading processor might not be available until the storage subsystem has completed its initialization in that processor. If you configure a generic process to start during the KERNEL phase, that process must be able to handle the situation where a device is not available until after that generic process is started; that is, it must continue running without the necessary resources being available. That process must also be capable of recognizing when those resources have become available and take necessary steps to use them.

  For this reason, HP suggests that you configure a generic process using the APPLICATION or MANUAL start mode.

- When the system is loaded (and the processor in which a generic process is configured is reloaded), the configured value for the STARTMODE attribute takes effect, regardless of the current state of the process. For example, if you configure a process with a STARTMODE value of APPLICATION, it will be started in the APPLICATION phase when the system is next loaded, even if the process was in the STOPPED object state when the system was loaded.

  When the processor in which a generic process is configured is later reloaded, the persistence of the process depends on its last object state (unless the STARTMODE is DISABLED). For example, if you abort a generic process, a processor reload does not restart the process. You must restart it manually.

Persistence Considerations

When specifying an AUTORESTART value for a generic process, consider that:

- The persistence of a generic process governs the restart of a generic process on a running system; that is, after the system is up (and the processor in which the generic process is configured and loaded). This is in contrast to the start mode of a generic process, which governs its startup during system load (and initial processor reload).
- A generic process is configured to be persistent if it is configured with a nonzero AUTORESTART value (described on page 6-6). This AUTORESTART value is known as the persistence count. A generic process with an AUTORESTART value of 10 (the maximum) is said to have a persistence count of 10. That is, the process can be restarted up to 10 times in a 10-minute interval. After 10 minutes of uninterrupted operation, the persistence count is restored to its original value.
- A generic process that is configured with a nonzero AUTORESTART value is known as a persistent generic process. If a persistent generic process abends due to a processor failure, the $ZPM persistence manager restarts it but does not decrement the persistence count.

  If a persistent generic process abends or is stopped outside of SCF (but not by a processor failure), the $ZPM persistence manager restarts it and decrements the persistence count by

1. $ZPM responds this way until the persistence count is reduced to zero (or until the user stops the generic process with an SCF ABORT command).

- If you create a process that stops or abends frequently, and you specify a high AUTORESTART value, system performance is negatively affected. To reduce the impact on system performance, you can use the "ALTER Command (Sensitive Command)" (page 95) to give the process a lower AUTORESTART value.

  Table 3-5 lists the reasons a generic process can be stopped and, for each reason, whether the persistence count is decremented.

Table 3-5 Effect of Stopping on the Persistence of a Generic Process

Reasons a Generic Process Is Stopped	Persistence Count Is
It was never started (the start mode is MANUAL or DISABLED).	Not decremented
It was started, but its processor was down at the time.	Not decremented
It was started, but its processor went down later.	Not decremented
It was stopped outside SCF (for example, by a TACL STOP command).	Decremented
It was stopped inside SCF by an ABORT command.	Reset to 0
It abended.	Decremented
Its persistence count was decremented to 0.	0

- The ABORT command effectively sets the persistence count to zero. A processor reload does not restart an aborted process. To restart a generic process in the STOPPED object state, ABORTED substate, you must either issue an SCF START command or load the system. See "Restarting an Aborted Generic Process" (page 57).
- The START command sets or resets the persistence count to its configured value.
- The SCF INFO command displays the configured (not the current) persistence count.

System Load Considerations

The effect on a generic process of a system load (and initial reload of the processor in which the generic process is configured) depends on the configured start mode. Any process with a start mode other than MANUAL or DISABLED is started when its processor comes up, regardless of its configured or current persistence count. After the system load finishes, the persistence count is reset to its configured value.

Table 3-6 lists these system load consideration.

Table 3-6 System Load Considerations for Persistence

	Consideration	Situation
If…	Generic process object state prior to system load…	Has any value
and if…	Start mode…	Is KERNEL, SYSTEM, or APPLICATION
Then…	Persistence count…	Is reset to its configured value
and…	$ZPM…	Restarts the generic process regardless of configured persistence count. ($ZPM might restart the generic process in the backup processor, if that processor comes up before the primary processor.)

Processor Reload Considerations

The effect on a generic process of a later processor reload (not the first reload of the processor after a system load) depends on all of these circumstances:

- The object state the process was in before the processor reload

  A process in the STOPPED state, ABORTED substate is not restarted.

- The configured start mode

  A prosthetic has a start mode of KERNEL, SYSTEM, or APPLICATION and that is not in the ABORTING state or ABORTED substate is automatically started when its processor comes up regardless of the autoconfigured AUTORESTART value.

  A process with a start mode of MANUAL is automatically restarted when its processor comes up, if it is not in the ABORTING state or ABORTED substate and if one of these is true:

  — You entered a START command after the processor was down.
  — It had been running when the processor went down, and its configured AUTORESTART value is greater than 0.

As a result of a processor reload, both of these conditions are true:

- The persistence count is not decremented.
- The $ZPM persistence manager starts the generic process.

Table 3-7 lists these processor reload considerations.

Table 3-7 Later Processor Reload Considerations for Persistence

	Consideration	Situation 1	Situation 2
If…	Generic process object state prior to processor load…	Is either STARTED state or STOPPED state, STOPPED substate	Is either ABORTING state or STOPPED state, ABORTED substate
and if…	Start mode…	Is KERNEL, SYSTEM, APPLICATION, or (if it had been started) MANUAL	Has any value
Then…	Persistence count…	Is not decremented	Is reduced to 0 by ABORT
and…	$ZPM…	Restarts the process regardless of the current persistence count. ($ZPM might restart the generic process in the backup processor if that processor comes up before the primary processor.)	Doesn't restart the process.

Abnormal Event Considerations

An abnormal event is defined as an event that stops a process other than an SCF ABORT command or a processor failure. For example:

- The process abends and its object state becomes STOPPED, while its substate is not ABORTED (as shown by an SCF STATUS command display for the process).
- The process is stopped outside SCF, perhaps by a TACL STOP command.

After an abnormal event stops a generic process:

- The $ZPM persistence manager restarts the process if its persistence is greater than zero, even if its start mode is MANUAL.
- $ZPM decrements the persistence count.

Table 3-8 lists these abnormal event considerations.

Table 3-8 Abnormal Event Considerations for Persistence

	Consideration	Situation
If…	Generic process object state prior to system load…	Is STOPPED
and if…	Start mode…	Is KERNEL, SYSTEM, APPLICATION, or MANUAL
then…	Persistence count…	Is decremented
and…	$ZPM…	If persistence count > 0, restarts the process. ($ZPM might restart the generic process in the backup processor, if that processor comes up before the primary processor.) If persistence count = 0, does not restart the process.

Restarting an Aborted Generic Process

Only an SCF ABORT command can place a generic process in the STOPPED object state, ABORTED substate.

You can restart an aborted generic process by using the SCF START command or by loading the system. When the generic process restarts:

- The configured start mode cannot be DISABLED, or the process will not start.
- The persistence count is reset to its configured value.
- If the processor in which the generic process is configured is available, the $ZPM persistence manager starts the generic process and changes its object state to STARTED.
- If its processor is down, the generic process object state changes from the ABORTED substate to the STOPPED substate. When the processor comes up, the $ZPM persistence manager starts the generic process and changes its state from STOPPED to STARTED.

To prevent a generic process (whether aborted or not) from restarting after the system is loaded, you must use either the SCF ALTER PROCESS command to change its STARTMODE attribute value or the SCF DELETE PROCESS command to remove the process from the system configuration database.

Displaying Information About a Generic Process

You can use the SCF STATUS, INFO, and NAMES commands to display different kinds of information about generic processes created by the $ZZKRN Kernel subsystem manager:

- Use the NAMES command to display the names of all generic processes managed by the Kernel subsystem manager. For example:

```
-> NAMES PROCESS $ZZKRN

NONSTOP KERNEL - Names PROCESS \EAST.$ZZKRN
Process
$ZZKRN.#CEV-SERVER-MANAGER-P0      $ZZKRN.#CEV-SERVER-MANAGER-P1
$ZZKRN.#CLCI-TACL                  $ZZKRN.#CHK
$ZZKRN.#OSM-APPSRVR                $ZZKRN.#OSM-CIMOM
$ZZKRN.#OSM-CONFLH-RD              $ZZKRN.#OSM-OEV
$ZZKRN.#QIOMON                     $ZZKRN.#ROUTING-DIST
$ZZKRN.#TCPIP-ZTC02                $ZZKRN.#TSM-SNMP
$ZZKRN.#SP-EVENT                   $ZZKRN.#TSM-SRM
$ZZKRN.#ZLOG                       $ZZKRN.#ZTCP0
$ZZKRN.#ZTCP1                      $ZZKRN.#ZHOME
$ZZKRN.#ZZKRN                      $ZZKRN.#ZZLAN
$ZZKRN.#ZZSTO                      $ZZKRN.#ZZWAN
```

- Use the INFO, DETAIL command to display the configured attributes of a specific generic process. For example:

```
-> INFO PROCESS $ZZKRN.#TEMP, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#TEMP

 *AutoRestart..............2
 *BackupCPU................Not Specified
 *CPU......................01
 *DefaultVolume............$SYSTEM.NOSUBVOL
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$TERM.#T1
 *InFile...................$TERM.#T1
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$PNAME
 *OutFile..................$TERM.#T1
 *PFSSize..................Not Specified
 *PrimaryCPU...............Not Specified
 *Priority.................169
 *Program..................$SYSTEM.SYSTEM.TACL
 *SaveAbend................OFF
 *StartMode................MANUAL
 *StartupMessage...........Not Specified
 *StopMode.................Standard
 *Type.....................OTHER
 *UserId...................SUPER.SUPER ( 255,255 )
```

- Use a STATUS command to display current status information about a specific generic process. For example:

```
-> STATUS PROCESS $ZZKRN.#PNAME, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \EAST.$ZZKRN.#PNAME

Backup PID........ None
Creation Time..... NOV 04,1996 15:57:21
Name.............. $PNAME
OwnerID........... 255, 255
Primary PID....... 1 , 20
Priority.......... 169
State............. STARTED
Substate..........
```

Adding a Generic Process

To configure a generic process, you enter an SCF ADD command that specifies these attributes:

- The process name (see the NAME attribute)
- The program name (see the PROGRAM attribute)
- A processor number entry (see the CPU attribute or PRIMARYCPU attribute, and see Table 3-1 (page 52))
- Other optional attributes (as described for the "ADD Command (Sensitive Command)" (page 86))

Example

This example shows how to use SCF to configure a single generic process in processor 3.

```
-> ASSUME PROCESS $ZZKRN
-> ADD #GP3, &
       CPU 3, &
       NAME $GP3, &
       PROGRAM $SYSTEM.SYSTEM.QIOMON, &
```

```
        LIBRARY $SYSTEM.SYS00.QIOLIB, &
        INFILE $TERM.#T1, &
        AUTORESTART 10
-> START #GP3
```

Creating a Generic Process in More Than One Processor

To configure a generic process in more than one processor, enter an SCF ADD command that specifies the preceding attributes, except:

- You specify the processor numbers of all processors in which the process is to run, either by number or by using ALL. (See the CPU attribute.)
- The length of the NAME attribute must be limited to three alphanumeric characters. When each generic process is started, SCF completes its name by adding the processor number to the alphanumeric prefix; for example, the name $ZM becomes $ZM00.

Example

This example shows how to use SCF to configure multiple QIOMON generic processes. For more information about QIO, see the *QIO Configuration and Management Manual*.

```
-> ASSUME PROCESS $ZZKRN
-> ADD #QIOMON, &
          AUTORESTART 10, &
          CPU ALL, &
          HOMETERM $ZHOME, &
          NAME $ZM, &
          OUTFILE $ZHOME, &
          PRIORITY 199, &
          PROGRAM $SYSTEM.SYS00.QIOMON
-> START #QIOMON
-> INFO #QIOMON, DETAIL
NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#QIOMON

*AutoRestart..............10
*BackupCPU................Not Specified
*CPU......................ALL
*DefaultVolume............$SYSTEM.SYSTEM
*ExtSwap..................Not Specified
*Highpin..................ON
*HomeTerminal.............$ZHOME
*InFile...................$Not Specified
*Library..................$Not Specified
*MemPages.................0
*Name.....................$ZMnn
*OutFile..................$ZHOME
*PFSSize..................Not Specified
*PrimaryCPU...............Not Specified
*Priority.................199
*Program..................$SYSTEM.SYSTEM.QIOMON
*SaveAbend................OFF
*StartMode................KERNEL
*StartupMessage...........Not Specified
*StopMode.................Standard
*Type.....................OTHER
*UserId...................SUPER.SUPER ( 255,255 )
```

Creating a Generic Process as a Process Pair

You must provide extra configuration information if you want your process to run as a process pair. Specifically, you must decide whether the process must always start in the primary processor or can start in either processor, as listed in Table 3-9: Using the PRIMARYCPU and STARTUPMSG Attributes.

Table 3-9 Using the PRIMARYCPU and STARTUPMSG Attributes

Configuration Attributes When...	If Processor 0 Is Up...	If Processor 0 Is Down...
Primary Processor Must Start:		
PRIMARYCPU 0 STARTUPMSG "1"	`RUN prog /CPU 0/1` (Both processors run)	Process does not start.
Either Processor Can Start:		
PRIMARYCPU 0 BACKUPCPU 1 STARTUPMSG "<BCKP-CPU>"	`RUN prog /CPU 0/1` (Both processors run)	`RUN prog /CPU 1/0` (Backup processor runs)

The assumptions for creating a generic process as a process pair are:

- The program is configured to get its backup processor from the startup message.
- The program's primary process starts its own backup process when that processor becomes available.
- Once started, $ZPM manages the persistence of the process or process pair.

Starting in a Designated Processor

Configuring a generic process to always start in the designated primary processor is also known as fixed processor configuration. To create a fixed processor configuration, specify these attributes in the SCF ADD command:

- The PRIMARYCPU attribute with the primary processor number
- The STARTUPMSG attribute with the backup processor number

You should specify a fixed processor configuration if the process must start with its primary process in the primary processor; that is, you do not want the process to start in the backup processor.

Do not specify the BACKUPCPU attribute.

Example

This example illustrates configuring a generic process that must start with the primary process in processor 0:

```
-> ADD PROCESS $ZZKRN.#PROC-A, &
       NAME $PROCA, &
       PROGRAM $SYSTEM.SYSTEM.TACL, &
       HOMETERM $TERM.#T1, &
       INFILE $TERM.#T1, &
       OUTFILE $TERM.#T1, &
       PRIMARYCPU 0, &
       STARTUPMSG "1", &
       STARTMODE MANUAL
-> START PROCESS $ZZKRN.#PROC-A
-> INFO PROCESS $ZZKRN.#PROC-A, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#PROC-A

*AutoRestart..............0
*BackupCPU................Not Specified
*CPU......................Not Specified
*DefaultVolume............$SYSTEM.NOSUBVOL
*ExtSwap..................Not Specified
*Highpin..................ON
*HomeTerminal.............$TERM.#T1
*InFile...................$TERM.#T1
*Library..................Not Specified
*MemPages.................Not Specified
```

```
*Name.....................$PROCA
*OutFile..................$TERM.#T1
*PFSSize..................Not Specified
*PrimaryCPU...............0
*Priority.................167
*Program..................$SYSTEM.SYSTEM.TACL
*SaveAbend................OFF
*StartMode................MANUAL
*StartupMessage...........1
*StopMode.................Standard
*Type.....................OTHER
*UserId...................SUPER.SUPER ( 255,255 )
```

In this example, processor 0 must be up for the generic process to start, and the backup process starts in processor 1 when it is available.

Starting in Either Processor

Configuring a generic process to start in either of the specified primary and backup processors (that is, to be fault tolerant) is also known as dynamic processor configuration. The SCF Kernel, storage, SLSA, and WAN subsystem managers are configured this way. To configure a process to start in either processor, specify these attributes in the SCF ADD command:

- The PRIMARYCPU attribute with the primary processor number
- The BACKUPCPU attribute with the backup processor number
- The STARTUPMSG attribute with this text, including the less than (<) and greater than (>) symbols:

  ```
  <BCKP-CPU>
  ```

When the process is launched, the $ZPM persistence manager substitutes the backup processor number in place of the <BCKP-CPU> text, depending on which processor the primary process is started in. You can use the STATUS PROCESS command to display the primary and backup processor numbers (as shown in this example).

Example

This example illustrates configuring a generic process to start in the first available processor configured for this process:

```
-> ADD PROCESS $ZZKRN.#PROC-B, &
NAME $PROCB, &
PROGRAM $SYSTEM.SYSTEM.TACL, &
HOMETERM $TERM.#T1, &
INFILE $TERM.#T1, &
OUTFILE $ TERM.#T1, &
PRIMARYCPU 0, &
BACKUPCPU 1, &
STARTUPMSG "<BCKP-CPU>", &
STARTMODE MANUAL
-> START PROCESS $ZZKRN.#PROC-B
-> INFO PROCESS $ZZKRN.#PROC-B, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#PROC-B

*AutoRestart..............0
*BackupCPU................1
*CPU......................Not Specified
*DefaultVolume............$SYSTEM.NOSUBVOL
*ExtSwap..................Not Specified
*Highpin..................ON
*HomeTerminal.............$TERM.#T1
*InFile...................$TERM.#T1
*Library..................Not Specified
*MemPages.................Not Specified
```

```
*Name.....................$PROCB
*OutFile..................$TERM.#T1
*PFSSize..................Not Specified
*PrimaryCPU...............0
*Priority.................167
*Program..................$SYSTEM.SYSTEM.TACL
*SaveAbend................OFF
*StartMode................MANUAL
*StartupMessage...........<BCKP-CPU>
*StopMode.................Standard
*Type.....................OTHER
*UserId...................SUPER.SUPER ( 255,255 )
```

In this example, if processor 0 is up, the primary process starts there and starts the backup process in processor 1 when processor 1 is available. If processor 0 is down, the primary process starts in processor 1 and starts the backup process in processor 0 when processor 0 is available.

This STATUS display shows that the primary process was started in processor 0 and the backup process was started in processor 1 with this command:

```
-> STATUS PROCESS $ZZKRN.#PROC-B, DETAIL

NONSTOP KERNEL - Detailed Status Process \EAST.$ZZKRN.#PROC-B
Backup PID........ 1 , 21
Creation Time..... JAN 18,2000 16:48:50
Name.............. $PROCB
OwnerID........... 255, 255
Primary PID....... 0 , 32
Priority.......... 169
State............. STARTED
Substate..........
```

Starting a Generic Process

To start a generic process configured in one or more processors, you use the commands listed in this table in the order indicated by the numerals in first column.

Table 3-10 Starting a Generic Process

	SCF Command	Purpose
1.	INFO, DETAIL	To view the configured attributes and values
2.	STATUS	To display the current object state of each instance of the generic process
3.	TIMEOUT	(optional) To allow time for SCF to receive complete results on the starting of a process group
4.	START	To place the generic process into operation
5.	STATUS	To verify that each instance of the generic process is in the STARTED object state

Example

This example shows how to check the attributes of the $ZZKRN.#GP generic process. It also shows how to start the process and display its status.

1. After adding a generic process configured to run in four processors, verify the attributes by examining an INFO, DETAIL display. For example:

```
-> INFO PROCESS $ZZKRN.#GP, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#GP

*AutoRestart..............0
*BackupCPU................Not Specified
```

```
*CPU.....................(00,01,02,03)
*DefaultVolume.............$SYSTEM.NOSUBVOL
*ExtSwap...................Not Specified
*Highpin...................ON
*HomeTerminal..............$TERM.#T1
*InFile....................$TERM.#T1
*Library...................Not Specified
*MemPages..................Not Specified
*Name......................$GP4nn
*OutFile...................$TERM.#T1
*PFSSize...................Not Specified
*PrimaryCPU................Not Specified
*Priority..................167
*Program...................$SYSTEM.SYSTEM.TACL
*SaveAbend.................OFF
*StartMode.................MANUAL
*StartupMessage............Not Specified
*StopMode..................Standard
*Type......................OTHER
*UserId....................SUPER.SUPER ( 255,255 )
```

2. A STATUS command shows that all instances of the object state are STOPPED:

```
-> STATUS PROCESS $ZZKRN.#GP

NONSTOP KERNEL - Status PROCESS \EAST.$ZZKRN.#PROCESS-A

Symbolic Name  Name   State    Sub Primary   Backup Owner
                                   PID       PID    ID
GP             $GP400 STOPPED    None      None
GP             $GP401 STOPPED    None      None
GP             $GP402 STOPPED    None      None
GP             $GP403 STOPPED    None      None
```

3. If necessary, use a TIMEOUT command (described in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*) to specify a timeout value that is larger than the default value of 90 seconds.

   When you start a generic process that has been configured in multiple processors (by, for example, the CPU ALL attribute), SCF might need more time to finish starting and reporting about all instances of the generic process. This command allows time for the START command that follows to start all instances of the generic process:

```
-> TIMEOUT
```

4. Start the process with a START command, For example:

```
KERNEL W00030 Process \EAST.$GP400 started
successfully.
KERNEL W00030 Process \EAST.GP401 started
successfully.
KERNEL W00106 One or more processes of \EAST.$ZZKRN.#GP did
not start due to cpu down. They will start when their CPU is
reloaded.
```

   Two instances of the process did not start because their associated processors are down.

   The TIMEOUT value stays in effect until the session ends or until you issue another TIMEOUT command.

5. Verify that the status is now STARTED and that processes are running in the configured and up processors, for example:

```
-> STATUS PROCESS $ZZKRN.#GP

NONSTOP KERNEL - Status PROCESS \EAST.$ZZKRN.#GP

Symbolic Name  Name   State       Sub Primary  Backup  Owner
```

```
                                          PID      PID      ID
GP              $GP400 STARTED            0 ,34    None     0 ,0
GP              $GP401 STARTED            1 ,23    None     0 ,0
GP              $GP402 STOPPED            None     None
GP              $GP403 STOPPED            None     None
```

Altering a Generic Process

To alter a generic process configured in one or more processors, use these SCF commands in the order indicated by the numerals in the first column of the table.

Table 3-11 Altering a Generic Process

	SCF Command	Purpose
1.	INFO, DETAIL	To view the configured attributes and values
2.	STATUS	To display the current object state of each instance of the generic process
3.	TIMEOUT	(optional) To allow time for SCF to receive complete results on the starting of a process group
4.	ABORT	To put all instances of the process into the STOPPED object state (if not already in that state)
5.	STATUS	To verify that all instances of the process are in the STOPPED object state
6.	ALTER	To change the configured attributes and values
7.	INFO, DETAIL	To verify that the change was made as requested
8.	START	To place the generic process into operation
9.	STATUS	To verify that each instance of the generic process is in the STARTED object state

Considerations

- The ABORT command stops the generic process in each processor in which it is running. Before altering a generic process, all instances of it must be in the STOPPED object state, ABORTED or STOPPED substate.
- If necessary, repeat the ABORT command.
- If a STATUS command shows an object state to be ABORTING or STOPPING, wait until the object state becomes STOPPED before entering the ALTER command.
- An unsuccessful ABORT command can cause the object state to remain as ABORTING. If a repeated ABORT command is not successful in changing the state from ABORTING to STOPPED, other commands will be unable to alter or delete the object. If so, you can alter the STARTMODE (to DISABLED) to prevent the process from being restarted after the next system load.

Example

This example shows how to stop the $ZZKRN.#GP generic process. Then it shows how to change the AUTORESTART value and restart the process.

1. View the process configuration with an INFO, DETAIL command:

```
-> ASSUME PROCESS $ZZKRN.#GP
-> INFO, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#GP

*AutoRestart..............0
*BackupCPU................Not Specified
*CPU......................(00,01,02,03)
```

```
*DefaultVolume.............$SYSTEM.NOSUBVOL
*ExtSwap...................Not Specified
*Highpin...................ON
*HomeTerminal..............$TERM.#T1
*InFile....................$TERM.#T1
*Library...................Not Specified
*MemPages..................Not Specified
*Name......................$GP4nn
*OutFile...................$TERM.#T1
*PFSSize...................Not Specified
*PrimaryCPU................Not Specified
*Priority..................167
*Program...................$SYSTEM.SYSTEM.TACL
*SaveAbend.................OFF
*StartMode.................MANUAL
*StartupMessage............Not Specified
*StopMode..................Standard
*Type......................OTHER
*UserId....................SUPER.SUPER ( 255,255 )
```

2. Display the current object state and substate of each instance of the generic process in each processor in which it is configured, using the STATUS command. In the sample command and display, processor 0 and 1 are up, but processors 2 and 3 are not. As a result, only $GP400 and $GP401 are started and can display a primary PID.

```
-> STATUS

NONSTOP KERNEL - Status PROCESS \EAST.$ZZKRN.#GP

Symbolic Name     Name    State    Sub Primary   Backup    Owner
                                       PID        PID       ID
GP                $GP400 STARTED     0 ,34      None      0 ,0
GP                $GP401 STARTED     1 ,23      None      0 ,0
GP                $GP402 STOPPED     None       None
GP                $GP403 STOPPED     None       None
```

3. If necessary, use a TIMEOUT command (described in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*) to specify a timeout value that is larger than the default value of 90 seconds.

   When you abort a generic process that has been configured in multiple processors (for example, by the CPU ALL attribute), SCF might need more time to finish stopping and reporting about all instances of the generic process. This command allows time for the ABORT command that follows to stop all instances of the generic process:

```
-> TIMEOUT
```

4. Stop all instances of the generic process with an ABORT command:

```
-> ABORT
KERNEL W00028 Process \EAST.$GP400 aborted successfully.
KERNEL W00028 Process \EAST.$GP401 aborted successfully.
KERNEL W-00016 Object \EAST.$ZZKRN.#$GP402 is already in STOPPED state
KERNEL W-00016 Object \EAST.$ZZKRN.#$GP403 is already in STOPPED state
```

   When the last process has been stopped, the TIMEOUT command ceases having an effect on SCF processing.

5. Verify that the process object state is STOPPED and substate is ABT (ABORTED). This substate is shown here in boldface.

```
-> STATUS

NONSTOP KERNEL - Status PROCESS \EAST.$ZZKRN.#GP

Symbolic Name   Name    State    Sub Primary   Backup    Owner
```

```
                                        PID       PID      ID
GP              $GP400 STOPPED ABT 0 ,34      None     0 ,0
GP              $GP401 STOPPED ABT 1 ,23      None     0 ,0
GP              $GP402 STOPPED ABT None       None
GP              $GP403 STOPPED ABT None       None
```

6. Change the AUTORESTART value, using the ALTER command:

```
-> ALTER, AUTORESTART 8
```

7. Confirm the changed AUTORESTART value with another INFO, DETAIL command. The changed line is shown here in boldface type.

```
-> INFO, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#GP

*AutoRestart..............8
*BackupCPU.................Not Specified
*CPU.......................(00,01,02,03)
*DefaultVolume.............$SYSTEM.NOSUBVOL
*ExtSwap...................Not Specified
*Highpin...................ON
*HomeTerminal..............$TERM.#T1
*InFile....................$TERM.#T1
*Library...................Not Specified
*MemPages..................Not Specified
*Name......................$GP4nn
*OutFile...................$TERM.#T1
*PFSSize...................Not Specified
*PrimaryCPU................Not Specified
*Priority..................167
*Program...................$SYSTEM.SYSTEM.TACL
*SaveAbend.................OFF
*StartMode.................MANUAL
*StartupMessage............Not Specified
*StopMode..................Standard
*Type......................OTHER
*UserId....................SUPER.SUPER ( 255,255 )
```

This change takes effect the next time the process is started.

8. Restart the process with the START command:

```
-> START
KERNEL W00030 Process \EAST.$GP400 started successfully.
KERNEL W00030 Process \EAST.$GP401 started successfully.
KERNEL W00106 One or more processes of \EAST.$ZZKRN.#GP did not start due
to cpu down. They will start when their CPU is reloaded.
```

9. Verify that the process is restarted with another STATUS command. The STARTED object state is shown here in bold. Because processors 2 and 3 are still down, $GP402 and $GP403 remain in the STOPPED object state.

```
-> STATUS

NONSTOP KERNEL - Status PROCESS \EAST.$ZZKRN.#GP

Symbolic Name           Name   State     Sub Primary   Backup  Owner
                                             PID       PID     ID
GP                      $GP400 STARTED       0 ,34     None    0 ,0
GP                      $GP401 STARTED       1 ,23     None    0 ,0
GP                      $GP402 STOPPED       None      None
GP                      $GP403 STOPPED       None      None
```

Deleting a Generic Process

To delete a generic process configured in one or more processors, use these SCF commands in the order indicated by the numeral in the first column of the table.

Table 3-12 Deleting a Generic Process

	SCF Command	Purpose
1.	STATUS	To display the current object state of each instance of the generic process
2.	STOP	To stop any processes communicating with the generic process
3.	TIMEOUT	(optional) To allow time for SCF to receive complete results on the starting of a process group
4.	ABORT	To put all instances of the process into the STOPPED object state (if not already in that state)
5.	STATUS	To verify that all instances of the process are in the STOPPED object state
6.	DELETE	To remove all instances of the generic process from the system configuration
7.	INFO	To verify that the generic process has been removed from the system configuration database

NOTE: You cannot stop or delete the $ZZKRN subsystem manager itself.

Example

This example shows how to delete the $ZZKRN.#GP generic process from the system.

1. Display the object state of each instance of the generic process, using the STATUS command:

```
-> ASSUME PROCESS $ZZKRN.#GP
-> STATUS

NONSTOP KERNEL - Status PROCESS \EAST.$ZZKRN.#GP

Symbolic Name          Name   State     Sub Primary  Backup  Owner
                                              PID      PID     ID
GP                     $GP400 STARTED       0 ,34    None    0 ,0
GP                     $GP401 STARTED       1 ,23    None    0 ,0
GP                     $GP402 STOPPED       None     None
GP                     $GP403 STOPPED       None     None
```

2. Stop any processes communicating with an instance of the generic process.
3. When you stop a generic process that has been configured as a group in multiple processors (by, for example, the CPU ALL attribute), or when you stop multiple generic processes (by using a wild card in the ABORT command), SCF might need more time to finish aborting and reporting about all instances of the generic process. If you are starting such a generic process group and if you have not already done so, enter a TIMEOUT command (described in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*) to specify a timeout value that is larger than the default value of 90 seconds. This command sets an unlimited timeout value:

```
-> TIMEOUT
```

4. Stop all instances of the generic process with an ABORT command.

```
-> ABORT
KERNEL W00028 Process \EAST.$GP400 aborted successfully.
KERNEL W00028 Process \EAST.$GP401 aborted successfully.
KERNEL W-00016 Object \EAST.$ZZKRN.#$GP402 is already in STOPPED state
KERNEL W-00016 Object \EAST.$ZZKRN.#$GP403 is already in STOPPED state
```

5. Use a STATUS command to verify that each instance of the generic process is in the STOPPED object state, ABORTED substate. The state and substate are shown here in bold.

```
-> STATUS

NONSTOP KERNEL - Status PROCESS \EAST.$ZZKRN.#GP

Symbolic Name          Name    State     Sub Primary  Backup  Owner
                                                PID      PID     ID
GP                     $GP400 STOPPED   ABT None     None
GP                     $GP401 STOPPED   ABT None     None
GP                     $GP402 STOPPED   ABT None     None
GP                     $GP403 STOPPED   ABT None     None
```

6. Use a DELETE command to remove the definition of the generic process from the configuration file:

```
-> DELETE
```

   If you are deleting many instances of a generic process (configured, for example, with the CPU ALL attribute), it may take a few moments for the DELETE operation to finish.

7. Use an INFO command to verify (by the absence of an entry or by a "not found" message) that all instances of the generic process have been removed from the system configuration database:

```
-> INFO
KERNEL E-00017 Object \EAST.$ZZKRN.#GP Not Found
```

Configuring and Managing ASSIGNs, PARAMs, and DEFINEs for a Generic Process

This section describes the process to be followed to pass the ASSIGN, PARAM, and DEFINE messages to a generic process through SCF.

Passing the ASSIGN, PARAM, and DEFINE attributes to a generic process is similar to the way these messages are passed to a process started on TACL.

The considerations at the end of each subsection, describe the limitations in using these attributes.

Adding an ASSIGN to a Generic Process

To add an ASSIGN to a generic process, enter an SCF ADD ASSIGN command that specifies these attributes:

- ASSIGN name
- ASSIGN file attribute (for details, see the *TACL Reference Manual*)

Example: Adding an ASSIGN

This example shows how to add the ASSIGN attribute to a generic process.

```
-> ASSUME PROCESS $ZZKRN
-> ADD #GP, &
       (ASSIGN ABC, ABC,EXT (16,16), REC 16,BLOCK 16, &
        CODE 16,OUTPUT,EXCLUSIVE)
-> START #GP
-> INFO, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \MANGO.$ZZKRN.#GP

*AutoRestart..............0
*BackupCPU................Not Specified
*CPU......................Not Specified
*DefaultVolume............$SYSTEM.NOSUBVOL
*ExtSwap..................Not Specified
```

```
*Highpin..................ON
*HomeTerminal.............Not Specified
*InFile...................Not Specified
*Library..................Not Specified
*MemPages.................Not Specified
*Name.....................$GP
*OutFile..................Not Specified
*PFSSize..................Not Specified
*PrimaryCPU...............1
*Priority.................167
*Program..................$SYSTEM.SYSTEM.TACL
*SaveAbend................OFF
*StartMode................MANUAL
*StartupMessage...........Not Specified
*StopMode.................STANDARD
*Type.....................OTHER
*UserId...................SUPER.SUPER ( 255,255 )

*Assigns:
   ABC........................ $SYSTEM.SYSTEM.ABC, EXT (16,16),
                               EXCLUSIVE, OUTPUT, CODE 16, REC16,
                               BLOCK 16
```

Adding a PARAM to a Generic Process

To add a PARAM to a generic process, enter an SCF ADD PARAM command that specifies these attributes:

- PARAM name
- PARAM value (for details, see the *TACL Reference Manual*)

Example: Adding a PARAM

This example shows how to add the PARAM attribute to a generic process.

```
-> ASSUME PROCESS $ZZKRN
-> ADD #GP,                &
       (PARAM ABC ABC)
-> START #GP
-> INFO,DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \MANGO.$ZZKRN.#GP

*AutoRestart..............0
*BackupCPU................Not Specified
*CPU......................Not Specified
*DefaultVolume............$SYSTEM.NOSUBVOL
*ExtSwap..................Not Specified
*Highpin..................ON
*HomeTerminal.............Not Specified
*InFile...................Not Specified
*Library..................Not Specified
*MemPages.................Not Specified
*Name.....................$GP
*OutFile..................Not Specified
*PFSSize..................Not Specified
*PrimaryCPU...............1
*Priority.................167
*Program..................$SYSTEM.SYSTEM.TACL
*SaveAbend................OFF
*StartMode................MANUAL
*StartupMessage...........Not Specified
*StopMode.................STANDARD
*Type.....................OTHER
*UserId...................SUPER.SUPER ( 255,255 )
```

```
*Params:
   ABC................................ ABC
```

Adding a DEFINE to a Generic Process

To add a DEFINE to a generic process, enter an SCF ADD DEFINE command that specifies these attributes:

- DEFINE name
- Class name
- DEFINE attributes

For more details, see the *Guardian Programmer's Guide*.

Example: Adding a DEFINE

This example shows how to add the DEFINE attribute to a generic process.

```
-> ASSUME PROCESS $ZZKRN
-> ADD #GP,                  &
       (DEFINE =ABC, CLASS MAP, FILE ABC)
-> START #GP
-> INFO,DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \MANGO.$ZZKRN.#GP

*AutoRestart..............0
*BackupCPU................Not Specified
*CPU......................Not Specified
*DefaultVolume............$SYSTEM.NOSUBVOL
*ExtSwap..................Not Specified
*Highpin..................ON
*HomeTerminal.............Not Specified
*InFile...................Not Specified
*Library..................Not Specified
*MemPages.................Not Specified
*Name.....................$GP
*OutFile..................Not Specified
*PFSSize..................Not Specified
*PrimaryCPU...............1
*Priority.................167
*Program..................$SYSTEM.SYSTEM.TACL
*SaveAbend................OFF
*StartMode................MANUAL
*StartupMessage...........Not Specified
*StopMode.................STANDARD
*Type.....................OTHER
*UserId...................SUPER.SUPER ( 255,255 )

*Defines :
   =ABC..............................CLASS MAP, FILE
                                      \MANGO.$SYSTEM.SYSTEM.ABC
```

Considerations for Adding an ASSIGN, PARAM, or DEFINE

- Before adding ASSIGN, PARAM or DEFINE, ensure that a generic process exists.
- ASSIGN, PARAM, or DEFINE can be added separately; that is, one at a time.
- You cannot use the same ADD PROCESS command to add both an ASSIGN and a PARAM.
- You cannot specify multiple DEFINEs in the same ADD command.
- For the FILE attribute of CLASS MAP DEFINEs, if you specify the SYSTEM subvolume, the SCF searches for the file on the SYSTEM subvolume first and then on the current SYSnn subvolume.

Altering the ASSIGN Attribute of a Generic Process

Altering the ASSIGN attribute of a generic process is similar to altering any generic process attributes. For details, see the "ALTER Command for Using ASSIGNs, PARAMs, and DEFINEs" (page 106).

Example: Altering an ASSIGN

This example shows how to alter the ASSIGN attribute of a generic process.

```
-> ASSUME PROCESS $ZZKRN.#GP
-> ALTER, (ASSIGN ABC,XYZ)
-> INFO, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \MANGO.$ZZKRN.#PAG

*AutoRestart..............0
*BackupCPU................Not Specified
*CPU......................Not Specified
*DefaultVolume............$SYSTEM.NOSUBVOL
*ExtSwap..................Not Specified
*Highpin..................ON
*HomeTerminal.............Not Specified
*InFile...................Not Specified
*Library..................Not Specified
*MemPages.................Not Specified
*Name.....................$PAG
*OutFile..................Not Specified1
*PFSSize..................Not Specified
*PrimaryCPU...............1
*Priority.................167
*Program..................$SYSTEM.SYSTEM.TACL
*SaveAbend................OFF
*StartMode................MANUAL
*StartupMessage...........Not Specified
*StopMode.................STANDARD
*Type.....................OTHER
*UserId...................SUPER.SUPER ( 255,255 )

*Assigns:
  ABC........................ $SYSTEM.SYSTEM.XYZ, EXT (16,16),
                               EXCLUSIVE, OUTPUT, CODE 16, REC 16,
```

Altering the PARAM Attribute of a Generic Process

Altering the PARAM attribute of a generic process is similar to altering any generic process attributes. For details, see the "ALTER Command for Using ASSIGNs, PARAMs, and DEFINEs" (page 106).

Example: Altering a PARAM

This example shows how to alter the PARAM attribute of a generic process.

```
-> ASSUME PROCESS $ZZKRN.#GP
-> ALTER, (PARAM ABC XYZ)
-> INFO, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \MANGO.$ZZKRN.#PAG

*AutoRestart..............0
*BackupCPU................Not Specified
*CPU......................Not Specified
*DefaultVolume............$SYSTEM.NOSUBVOL
*ExtSwap..................Not Specified
*Highpin..................ON
```

```
*HomeTerminal.............Not Specified
*InFile...................Not Specified
*Library..................Not Specified
*MemPages.................Not Specified
*Name.....................$PAG
*OutFile..................Not Specified
*PFSSize..................Not Specified
*PrimaryCPU...............1
*Priority.................167
*Program..................$SYSTEM.SYSTEM.TACL
*SaveAbend................OFF
*StartMode................MANUAL
*StartupMessage...........Not Specified
*StopMode.................STANDARD
*Type.....................OTHER
*UserId...................SUPER.SUPER ( 255,255 )

*Params:
   ABC..............................XYZ
```

Altering the DEFINE Attribute of a Generic Process

Altering the DEFINE attribute of a generic process is similar to altering any generic process attributes. For details, see the "ALTER Command for Using ASSIGNs, PARAMs, and DEFINEs" (page 106).

Example: Altering a DEFINE

This example shows how to alter the DEFINE attribute of a generic process.

```
-> ASSUME PROCESS $ZZKRN.#GP
-> ALTER, (DEFINE =ABC, FILE XYZ)
-> INFO, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \MANGO.$ZZKRN.#GP

*AutoRestart..............0
*BackupCPU................Not Specified
*CPU......................Not Specified
*DefaultVolume............$SYSTEM.NOSUBVOL
*ExtSwap..................Not Specified
*Highpin..................ON
*HomeTerminal.............Not Specified
*InFile...................Not Specified
*Library..................Not Specified
*MemPages.................Not Specified
*Name.....................$GP
*OutFile..................Not Specified
*PFSSize..................Not Specified
*PrimaryCPU...............1
*Priority.................167
*Program..................$SYSTEM.SYSTEM.TACL
*SaveAbend................OFF
*StartMode................MANUAL
*StartupMessage...........Not Specified
*StopMode.................STANDARD
*Type.....................OTHER
*UserId...................SUPER.SUPER ( 255,255 )

*Defines:
   =ABC..............................CLASS MAP, FILE
                                      \MANGO.$SYSTEM.SYSTEM.XYZ
```

Considerations for Altering ASSIGNs, PARAMs, and DEFINEs

- For an ALTER command, if the associated physical file name must be altered, it should be the first attribute after the assign name itself.
- Unlike the TACL ASSIGN, PARAM, and DEFINE commands, which replace the previous parameters with new definitions, the ALTER ASSIGN, PARAM, and DEFINE commands related to a generic process alter only the specified attribute.
- Other considerations are same as those for a generic process. For more details, see considerations under "Altering a Generic Process" (page 64).

Deleting the ASSIGN Attribute of a Generic Process

Deleting the ASSIGN attribute of a generic process is similar to deleting any generic process attribute. For details, see "DELETE Command for Using ASSIGNs, PARAMs, and DEFINEs" (page 108).

Example

This example shows how to delete the ASSIGN attribute of a generic process.

```
-> ASSUME PROCESS $ZZKRN.#GP
-> DELETE, ASSIGN ABC
```

Deleting the PARAM Attribute of a Generic Process

Deleting the PARAM attribute of a generic process is similar to deleting any generic process attribute. For details, see "DELETE Command for Using ASSIGNs, PARAMs, and DEFINEs" (page 108).

Example

This example shows how to delete the PARAM attribute of a generic process.

```
-> ASSUME PROCESS $ZZKRN.#GP
-> DELETE, PARAM ABC
```

Deleting the DEFINE Attribute of a Generic Process

Deleting the DEFINE attribute of a generic process is similar to deleting any generic process attribute. For details, see "DELETE Command for Using ASSIGNs, PARAMs, and DEFINEs" (page 108).

Example

This example shows how to delete the DEFINE attribute of a generic process.

```
-> ASSUME PROCESS $ZZKRN.#GP
-> DELETE, DEFINE =ABC
```

4 Managing the ServerNet Network

A ServerNet fabric is the communication path for interprocessor messages and processor-to-I/O device communication. This chapter describes how you use the Kernel subsystem SERVERNET object type to query the ServerNet X and Y fabrics.

To monitor a ServerNet fabric or connection, refer to the OSM Service Connection or TSM Service Application. The OSM Service Connection or TSM Service Application lets you view alarms associated with a fabric and also describes repair actions. For more information, see the OSM Service Connection or TSM online help.

Obtaining Information About the ServerNet Network

You can use the NAMES, STATUS, and VERSION commands to display different kinds of information about the ServerNet network. For complete information about the status of ServerNet fabrics, refer to the OSM Service Connection or TSM Service Application.

- First use the NAMES command to display the names of all ServerNet links to processors on the system, even if the connecting processors do not exist. For example:

```
-> NAMES SERVERNET $ZSNET

NONSTOP KERNEL Names \EAST.$ZSNET
SERVERNET
$ZSNET
$ZSNET.X.0            $ZSNET.Y.0
$ZSNET.X.1            $ZSNET.Y.1
$ZSNET.X.2            $ZSNET.Y.2
$ZSNET.X.3            $ZSNET.Y.3
$ZSNET.X.4            $ZSNET.Y.4
$ZSNET.X.5            $ZSNET.Y.5
$ZSNET.X.6            $ZSNET.Y.6
$ZSNET.X.7            $ZSNET.Y.7
$ZSNET.X.8            $ZSNET.Y.8
$ZSNET.X.9            $ZSNET.Y.9
$ZSNET.X.10           $ZSNET.Y.10
$ZSNET.X.11           $ZSNET.Y.11
$ZSNET.X.12           $ZSNET.Y.12
$ZSNET.X.13           $ZSNET.Y.13
$ZSNET.X.14           $ZSNET.Y.14
$ZSNET.X.15           $ZSNET.Y.15
```

- Then use the STATUS command to display current information about the ServerNet network. In the sample display, all ServerNet connections on processors 0 through 3 are up. Because the system consists of four processors, the status of the connections from processors 0 through 3 to processors 4 through 15 is unavailable (UNA), and processors 4 through 15 are down. ServerNet status terms are defined on page 6-71.

```
-> STATUS SERVERNET $ZSNET

NONSTOP KERNEL - Status SERVERNET
X-FABRIC
   TO    0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
FROM
    0    UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    1    UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    2    UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    3    UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    4  <-DOWN
    5  <-DOWN
    6  <-DOWN
    7  <-DOWN
    8  <-DOWN
    9  <-DOWN
```

```
   10  <-DOWN
   11  <-DOWN
   12  <-DOWN
   13  <-DOWN
   15  <-DOWN
Y-FABRIC
   T0    0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
FROM
    0    UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    1    UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    2    UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    3    UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    4  <-DOWN
    5  <-DOWN
    6  <-DOWN
    7  <-DOWN
    8  <-DOWN
    9  <-DOWN
   10  <-DOWN
   11  <-DOWN
   12  <-DOWN
   13  <-DOWN
   14  <-DOWN
   15  <-DOWN
```

- You can also use the VERSION command to display the operating system version of the ServerNet network. For example

```
-> VERSION SERVERNET $ZSNET

VERSION SERVERNET \DELUX.$ZSNET: SERVERNET (MGR) - T1085F40 - (31AUG99) - (28MAY99)
```

Identifying ServerNet Hardware Failures

These hardware failures can affect a ServerNet X or Y fabric:

- There is no access to a processor multifunction (PMF) customer-replaceable unit (CRU).
- A ServerNet expansion board (SEB) or modular ServerNet expansion board (MSEB) is failing.

Before repairing such a failure, use the STATUS SERVERNET command to verify that all other ServerNet connections are accessible.

Example 1

If the STATUS SERVERNET display looks like this example, there is no access to the PMF CRU containing processor 1. The status conditions indicating that there are access problems are shown in bold.

```
-> STATUS SERVERNET $ZSNET
```

```
NONSTOP KERNEL - Status SERVERNET
X-FABRIC
   TO    0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
FROM
    0     UP  UNA  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    1   <-DOWN
    2     UP  UNA  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    3     UP  UNA  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    4   <-DOWN
    5   <-DOWN
    6   <-DOWN
    7   <-DOWN
    8   <-DOWN
    9   <-DOWN
   10   <-DOWN
   11   <-DOWN
   12   <-DOWN
   13   <-DOWN
   15   <-DOWN
Y-FABRIC
   TO    0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
FROM
    0     UP  UNA  UP  UP UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    1   <-DOWN
    2     UP  UNA  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    3     UP  UNA  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    4   <-DOWN
    5   <-DOWN
    6   <-DOWN
    7   <-DOWN
    8   <-DOWN
    9   <-DOWN
   10   <-DOWN
   11   <-DOWN
   12   <-DOWN
   13   <-DOWN
   14   <-DOWN
   15   <-DOWN
```

Because the STATUS display shows that all other connections are up, you can replace the failing PMF CRU. To replace the PMF/SEB or MSEB using TSM, launch the guided procedure from **Start > Programs > Compaq TSM > Guided Replacement Tools > Replace PMF/SEB or MSEB**.

Example 2

If the STATUS SERVERNET display looks like this example, a ServerNet expansion board (SEB) or modular ServerNet expansion board (MSEB) is failing. You can also determine the status of your SEB or MSEB by viewing related alarms and attributes in the OSM Service Connection or TSM Service Application. The status conditions indicating that there are access problems are shown in bold.

```
-> STATUS SERVERNET $ZSNET
```

```
NONSTOP KERNEL - Status SERVERNET
X-FABRIC
   T0     0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
FROM
    0     UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    1     UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    2     UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    3     UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    4   <-DOWN
    5   <-DOWN
    6   <-DOWN
    7   <-DOWN
    8   <-DOWN
    9   <-DOWN
   10   <-DOWN
   11   <-DOWN
   12   <-DOWN
   13   <-DOWN
   15   <-DOWN
Y-FABRIC
   T0     0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
FROM
    0     UP  UP  DN  DN  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    1     UP  UP  DN  DN  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    2     DN  DN  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    3     DN  DN  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    4   <-DOWN
    5   <-DOWN
    6   <-DOWN
    7   <-DOWN
    8   <-DOWN
    9   <-DOWN
   10   <-DOWN
   11   <-DOWN
   12   <-DOWN
   13   <-DOWN
   14   <-DOWN
   15   <-DOWN
```

In the preceding example, the SEB or MSEB between group 01 and group 02 for the Y fabric is not accessible. To replace the PMF/SEB or MSEB using TSM, launch the guided procedure from **Start > Programs > Compaq TSM > Guided Replacement Tools > Replace PMF/SEB or MSEB**.

5 SCF Object Types and Object Names

Many SCF commands operate on the objects belonging to each subsystem. Each object has an object type and an object name. For information about SCF object naming conventions and typology, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*.

This chapter documents the SCF object types and object names that support the Kernel subsystem:

The null Object Type

The null object type is not an actual object type; the null object type represents the lack of a specified object type. A command can support the null object type if an object type is irrelevant (as is the case with the VERSION command), or if it refers to a collection of objects (as is the case with the NAMES command).

You can use the null object type with the NAMES command to identify all objects supported by the Kernel subsystem manager. Use the null object type with the VERSION command to obtain version information.

The value of *object-name* for the null object is:

$process

$*process*
is either $ZZKRN, the name of the Kernel subsystem manager process, or $ZSNET, the name of the ServerNet manager process.

Supported Commands

These commands use the null object type:

```
NAMES
VERSION
```

Consideration

Wild cards are not supported for the null object type.

Examples

- These object names are the valid for the null object type:

  ```
  $ZSNET
  $ZZKRN
  ```

- These commands use the null object type:

  ```
  NAMES $ZZKRN
  NAMES $ZSNET
  VERSION $ZZKRN
  VERSION $ZSNET
  ```

The PROCESS Object Type

The PROCESS object type identifies a specific process name, either the Kernel subsystem manager process or a generic process owned by the Kernel subsystem manager. The value of object-name for the PROCESS object type is:

```
{ $ZZKRN | $ZZKRN.#gpname }
```

$ZZKRN

is the name of the Kernel subsystem manager process. You can use this form of *object-name* in commands that operate on the Kernel subsystem manager.

$ZZKRN.#*gpname*

is the name of the Kernel subsystem manager process followed by a descriptive name of one or more generic processes managed by the Kernel subsystem. The name $ZZKRN.#*gpname* must form a unique name on the system. The name *gpname* can be up to 32 characters long. The first character must be a letter; the other can be any of these characters:

```
At sign (@)    Back slash (\)    Caret (^)         Colon (:)
Dash (-)       Equal sign (=)    Underscore (_)    Any alphanumeric character
```

Supported Commands

These commands use the PROCESS object type:

```
ABORT    ALTER     INFO     START     VERSION
ADD      DELETE    NAMES    STATUS
```

Considerations

- Wild-card support for the PROCESS object type is limited to use of the trailing asterisk (*) for #*gpname* in these commands:

  ```
  ABORT    DELETE    INFO    NAMES    START    STATUS
  ```

- You must fully specify #*gpname* for the ADD, ALTER, and VERSION commands.

Supported Object States

Supported object states for the PROCESS object type are:

ABORTING	Has these meanings: • The process is in transition to the STOPPED state, ABORTED substate, because the ABORT command was issued. The process is running but does not accept new user requests. • An abort operation began on the process, and although the process did not stop as requested, the process will not be started again until its processor is reloaded.	
STARTED	The process is running and can accept user requests.	
STOPPED	The process definition has been added to the subsystem configuration, but the process is not running. Use the START command to put the process in the STARTED state.	
	ABORTED substate	The process was stopped by an ABORT command. It is not running but still exists in the subsystem configuration. Use the DELETE command to remove the process from the subsystem configuration.
	STOPPED substate	The process is not running for one of these reasons: • It was stopped outside of SCF. • It was stopped because its processor went down. • It was never started (or it was started while its processor was down). • It abended.

Examples

These object names are valid for the PROCESS object type:

```
$ZZKRN
$ZZKRN.#MY-PROCESS
$ZZKRN.#*
$ZZKRN.#M*
```

The SERVERNET Object Type

You can use the SERVERNET object type to query and control the ServerNet X or Y fabric. The value of object-name for the SERVERNET object type is:

```
{ $ZSNET | $ZSNET.{X|Y}.cpu }
```

$ZSNET
: is the name of the ServerNet process. You use this form of object-name in the STATUS, VERSION, and NAMES commands.

$ZSNET.{X|Y}.*cpu*
: is the name of the ServerNet process, followed by the name of a ServerNet fabric (either X or Y), and the one-digit or two-digit processor number of the processor logically connected to the network. Use this form of the *object-name* when you want to specify a specific fabric and processor.

Supported Commands

These commands use the SERVERNET object type:

```
NAMES    START    STATUS    STOP    VERSION
```

Consideration

Wild-card support for the SERVERNET object type is limited to use of an asterisk (*) in place of the processor number.

Object Status

Object status for the SERVERNET object can be:

DIS	(disabled) A ServerNet fabric is down at the TO location. As a result, the path from the FROM processor to the TO processor is down for receiving, which means that the TO processor cannot receive from any other processor or from I/O devices on that fabric. DIS overrides UP and DN.
DN	(down) The path from the FROM processor to the "TO" processor is down because the path is failing. The FROM processor cannot communicate with the TO processor on that fabric.
<-DOWN	(across a row) The FROM processor is down or nonexistent.
E*nnn*	The ServerNet fabric unexpectedly returned a file-system error regarding the link from the FROM processor to the TO processor.
ERROR *nnn*	(across a row) The FROM processor unexpectedly returned file-system error *nnn* to the ServerNet fabric. For information about the file-system error, see the *Guardian Procedure Errors and Messages Manual*.
UNA	(unavailable) The link from the FROM processor to the "TO" processor is down because the TO processor is down or nonexistent. UNA overrides all other values.
UP	The path from the FROM processor to the TO processor is up.
UP*	The path from the FROM processor to the TO processor was left up in order not to bring down the last path between these two processors.

Examples

These are valid object names for the SERVERNET object type:

```
$ZSNET.X.0          == X fabric on processor 0
$ZSNET.Y.3          == Y fabric on processor 3
$ZSNET.Y.*          == Entire Y fabric
```

The SUBSYS Object Type

You can use the SUBSYS object type to query and control the configuration of the Kernel subsystem. The value of object-name for the SUBSYS object type is:

```
$ZZKRN
```

$ZZKRN
is the name of the Kernel subsystem manager process.

Supported Commands

These commands use the SUBSYS object type:

```
ALTER    CONTROL    INFO    NAMES    STATUS    VERSION
```

Consideration

Wild cards are not supported for the SUBSYS object type.

Supported Object State

The supported object state for the SUBSYS object type is:

STARTED	The process is running and can accept user requests.

Example

This is the valid object name for the SUBSYS object type:

```
$ZZKRN
```

6 SCF Commands for the Kernel Subsystem

This chapter describes the SCF commands that support the Kernel subsystem:

Other commands that are generally supported by SCF, such as the ASSUME and ENV commands, are not documented in this manual. The *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs* provides general information about SCF commands. You should be familiar with general SCF usage before using this manual.

Supported Commands and Object Types

Table 6-1 lists the SCF commands supported by the Kernel subsystem and the object types to which they apply. The object types are described in Chapter 5: SCF Object Types and Object Names (page 79).

Table 6-1 SCF Commands and Object Types for the Kernel Subsystem

	Object Types			
Command	**null**	**PROCESS**	**SERVERNET**	**SUBSYS**
ABORT	--	X	--	--
ADD	--	X	--	--
ALTER	--	X	--	X
CONTROL	--	--	--	X
DELETE	--	X	--	--
INFO	--	X	--	X

Table 6-1 SCF Commands and Object Types for the Kernel Subsystem *(continued)*

	Object Types			
Command	**null**	**PROCESS**	**SERVERNET**	**SUBSYS**
NAMES	X	X	X	X
START	--	X	X	--
STATUS	--	X	X	X
STOP	--	--	X	--
VERSION	X	X	X	X

X = The command currently supports this object type.
-- = The command does not support this object type.

Sensitive and Nonsensitive Commands

SCF commands are either sensitive or nonsensitive.

Sensitive SCF commands can cause communications to cease if the commands are improperly used. Only a super-group user (255,*n*), the owner of the subsystem, or a member of the group of the owner of the subsystem can issue a sensitive command.

Nonsensitive SCF commands request information or status but do not affect operation. SCP does not perform any security checking on these commands.

Table 6-2 lists the sensitive and nonsensitive SCF commands for the Kernel subsystem.

Table 6-2 Sensitive and Nonsensitive SCF Commands

Sensitive Commands	Nonsensitive Commands
ABORT	INFO
ADD	NAMES
ALTER	STATUS
CONTROL	VERSION
DELETE	
START	
STOP	

If you do not specify an object type or object name in the command, SCF uses the assumed object type and object name (set by the user with a previous ASSUME command) to expand any missing or partially qualified portions of these arguments. Based on the device type of the object named in the command, SCF selects the appropriate subsystem to finish processing the command.

Wild-Card Support

SCF commands for the Kernel subsystem allow you to make these wild-card character substitutions:

- A trailing asterisk (*) for `#gpname` in a PROCESS object name. For more specific information, see the considerations for each command.
- An asterisk in place of the *cpu* value in a SERVERNET object name.

ABORT Command (Sensitive Command)

The ABORT command terminates the operation of one or more PROCESS objects as quickly as possible. The object (or objects if the process is running in more than one processor) are left in the STOPPED object state, substate ABORTED, but remain configured in the system configuration database (CONFIG).

```
ABORT [ /OUT file-spec / ] PROCESS $ZZKRN.gpname
```

PROCESS $ZZKRN.*gpname*

is a generic process controlled by the Kernel subsystem manager process. You can omit PROCESS, $ZZKRN, and #*gpname* if you have specified them with an ASSUME command.

Considerations

- When aborting a generic process configured in multiple processors, you should consider entering a TIMEOUT command (described in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*) to specify a timeout value that is longer than the default 90 seconds. Several processes can be aborted within 90 seconds, but if you abort a generic process that has been configured as a group (for example, by the CPU ALL attribute) or if you abort multiple generic processes (by using a wild card in the ABORT command), more time may be needed.
- If a generic process does not seem to enter the STOPPED state, but instead appears to be ABORTING or STOPPING, repeat the ABORT command.
- The ABORT command effectively sets the persistence count to zero.
- After you have entered an ABORT command on a generic process, $ZPM does not restart the process until the system is loaded. $ZPM does not restart the process if the processor is reloaded. This is always true regardless of the start mode or persistence settings.
- To restart a generic process without loading the system, use the START PROCESS command, as described under "Restarting an Aborted Generic Process" (page 57).
- If the ABORT command fails with an error, the process stays in the ABORTING object state and a second ABORT command is unable to change the state to STOPPED. Although the process may still be running, it is treated like an aborted process and is not restarted when the processor is reloaded. If you want to ensure the process does not restart when the system is loaded, use the ALTER command to change the start mode to MANUAL or DISABLED.
- You cannot use the ABORT command on $ZZKRN (the Kernel subsystem manager process).
- You should not use the ABORT command on $ZZFOX (the FOX monitor process). Unpredictable results can occur.
- You should not use the ABORT command on $ZM*nn* (a QIO monitor process) while a client is active. Processor failure can occur.
- Wild-card support is limited to the trailing asterisk (*) for #*gpname*.

Consideration for OSS Persistent Processes

When you issue an ABORT command to an OSS persistent process object, you abort the processes specified in its NAME and ASSOCPROC attributes.

Examples

- To abort the #TEMP generic process, type:

  ```
  -> ABORT PROCESS $ZZKRN.TEMP
  ```

  Or type:

  ```
  -> ASSUME PROCESS $ZZKRN.TEMP
  -> ABORT
  ```

- To abort multiple instances of a generic process configured in multiple processors, type:

```
-> ASSUME PROCESS $ZZKRN.GP
-> ABORT

KERNEL W00028 Process \EAST.$GP00 aborted successfully.
KERNEL W00028 Process \EAST.$GP01 aborted successfully.
KERNEL W-00016 Object \EAST.$ZZKRN.#$GP02 is already in STOPPED state.
KERNEL W-00016 Object \EAST.$ZZKRN.#$GP03 is already in STOPPED state.
```

ADD Command (Sensitive Command)

Use the ADD command to define a process object in the Kernel subsystem and add it to the system configuration database (CONFIG).

```
ADD [ / OUT file-spec / ] PROCESS $ZZKRN.gpname
   [ , attribute-spec ]...
```

PROCESS $ZZKRN.#*gpname*

is a generic process controlled by the Kernel subsystem manager process. You can omit PROCESS, $ZZKRN, and #*gpname* if you have specified them with an ASSUME command.

attribute-spec

identifies the attribute names and values for the object specified in the command. The attributes are:

```
ASSOCPROC $name
AUTORESTART n
BACKUPCPU n
CPU { ALL | FIRST | FIRSTOF (n,n1,...) | n | (n,n1,...) }
DEFAULTVOL $vol[.subvol ]
EXTSWAP { $vol | [[$vol.]subvol.]filename }
HIGHPIN { ON | OFF }
HOMETERM $device[.#subdevice]
INFILE { $device | [[$vol.]subvol.]filename }
LIBRARY [[$vol.]subvol.]filename
MEMPAGES n
NAME $name
OUTFILE { $device | [[$vol.]subvol.]filename }
PFSSIZE n
PRIMARYCPU n
PRIORITY n
PROGRAM [[$vol.]subvol.]filename
SAVEABEND { ON | OFF }
STARTMODE { KERNEL | SYSTEM | APPLICATION | MANUAL | DISABLED }
STARTUPMSG "text"
STOPMODE { SPI | STANDARD | SYSMSG }
TYPE { FOXMON | SUBSYSTEM-MANAGER | OTHER }
USERID { groupname.username | groupnum,usernum }
```

ASSOCPROC *name*

specifies the name of an OSS application to be run as part of a named generic process, whose persistence is monitored by the Kernel subsystem's persistence-manager process ($ZPM). The OSH that launches the process must also be made persistent. Thus, when you specify the ASSOCPROC attribute, you must also:

- Specify the name of an OSH process in the NAME attribute
- Specify the OSH object file in the PROGRAM attribute.
- Use the STARTUPMSG attribute to specify the startup message that is sent to the OSS application.

Thus, the persistent process object created through an ADD PROCESS command consists of two distinct persistent processes: the OSH process and the OSS application process itself. An ABORT, DELETE, or START command issued to this object affects both processes.

AUTORESTART *n*

specifies the number of times that the $ZPM persistence manager attempts to restart this process within a 10-minute interval after an abnormal termination. The process abends or stops by a means other than the ABORT command.

If *n* is 0 (the default), the process is not automatically restarted.

If *n* is 1 through 10, the process is restarted as many as *n* times in 10 minutes.

If *n* is greater than 0 when a processor fails, the processor is reloaded, its previously running generic processes are restarted, but the value of *n* is not decremented. For more information about what conditions decrement the count, see Table 3-5: Effect of Stopping on the Persistence of a Generic Process (page 55).

For more information about using the AUTORESTART attribute, see "Persistence Considerations" (page 54).

BACKUPCPU *n*

specifies the processor in which this process should start its backup process. For more information, see "Controlling Where a Generic Process Starts" (page 52) and "Controlling When a Generic Process Starts" (page 52).

To specify this attribute, you must also specify (or have previously specified) the PRIMARYCPU attribute, but you must not specify the CPU attribute.

The variable *n* can be from 0 through the maximum number of processors (the BACKUPCPU value specified cannot be the same as the PRIMARYCPU value).

You should not specify BACKUPCPU for any OSS persistent process.

CPU { ALL | FIRST | FIRSTOF (*n*,*n*1,...) | *n* | (*n*,*n*1,...) }

specifies one or more processors in which to start this process. For more information, see "Controlling Where a Generic Process Starts" (page 52).

The CPU attribute is required if you do not specify the PRIMARYCPU attribute.

ALL

specifies that an instance of the process be started in all processors, even if a processor is currently not up. If you specify ALL, you must limit the process name specified in this ADD command to one, two, or three alphanumeric characters. A two-digit processor number is appended to the process name.

FIRST

specifies that the process be started in the first available processor.

Because processors 0 and 1 are always the first processors to be loaded, avoid configuring too many generic processes with CPU FIRST because this can lead to an uneven load balance among the processors in the system. OSS persistent processes running in multiple processors cannot have a specification of FIRST.

FIRSTOF (*n*,*n1*,...)

specifies that the process be started in the first available processor in the designated group. If the processor in which a process is configured fails (for example, processor 2), the process automatically starts in the next available processor (for example, processor 3). OSS persistent processes running in multiple processors cannot have a specification of FIRST OF.

n

specifies that the process be started in processor *n*.

(*n*,*n*1,...)

specifies that an instance of the process be started in each specified processor even if a processor is currently not up. You must limit the process name specified in this ADD

command to one, two, or three alphanumeric characters A two-digit processor number is appended to the process name.

DEFAULTVOL $*vol*[.*subvol*]

specifies the default volume and subvolume information sent to this process (in the startup message) when it is started.

If this attribute is not specified, the startup default volume and subvolume is $SYSTEM.NOSUBVOL.

EXTSWAP { $*vol* | [[$*vol*.]*subvol*.]*filename* }

specifies the volume for or name of the swap file for the default extended data segment of this process. This option applies only to TNS objects.

If this attribute is not specified, the operating system selects a swap file or disk location if the process needs it.

HIGHPIN { ON | OFF }

specifies the desired PIN range for the process.

ON (the default) specifies that the process run at a high PIN if both these statements are true:

- The high-PIN bit is enabled in the program file (and in the library file or files, if any).
- A high PIN is available.

The $ZPM persistence manager tries to create this process as a high-PIN process. If no high PIN is available in the specified processor, it tries to create this process as a low-PIN process. If no low PIN is available, the process creation fails.

OFF specifies that the process run at a low PIN, regardless of any other considerations. The $ZPM persistence manager tries to create this process as a low-PIN process. If no low PIN is available, the process creation fails.

HOMETERM $*device*[.#*subdevice*]

specifies the home terminal to use when starting this process.

If this attribute is not specified, the process has the same home terminal ($YMIOP.#CLCI) that $ZPM has when it starts the process.

If you configure a generic process to use the system console ($OSP on D-series RVUs; $YMIOP.#CLCI or $YMIOP.#CNSL on earlier G-series RVUs), you probably should specify HOMETERM $ZHOME. See Table 6-3.

Table 6-3 Guidelines for Configuring a HOMETERM Value

If the generic process is…	The HOMETERM value should be…
Able to handle errors returned from the home terminal	Set to the default value, $YMIOP.#CLCI
Not able to handle errors returned from the home terminal	Set to $ZHOME
Configured to be STARTMODE KERNEL or SYSTEM	Set to $ZHOME
Configured to be noninteractive	Set to $ZHOME
Configured to be interactive	Set to anything but $ZHOME
Configured to read from the home terminal	Set to anything but $ZHOME

For more information about the $ZHOME process, see the *NonStop S-Series Planning and Configuration Guide*, the *NonStop S-Series Operations Guide*, or the *NonStop Operations Guide* (for J- and H-series RVUs).

INFILE { $*device* | [[$*vol*.]*subvol*.]*filename* }

specifies the input file or process name sent to this process (in the startup message) when it is started.

If this attribute is not specified, the process has the same infile ($YMIOP.#CLCI) that $ZPM has when it starts the process.

Do not specify INFILE $ZHOME.

LIBRARY [[$*vol*.]*subvol*.]*filename*

specifies a library file for this process to use when started. The file must be either a file code type 100 (TNS) or 700 (TNS/R) object file.

If you specify the SYSTEM subvolume, SCF searches for the file first on the SYSTEM subvolume and then on the current SYS*nn* subvolume.

You cannot mix TNS (type 100) and TNS/R (type 700) object code. For example, if the PROGRAM attribute points to a TNS/R object file, then the LIBRARY modifier must also point to a TNS/R library object file.

MEMPAGES *n*

specifies the number of 1024-word pages of user data memory to be allocated to this process after it is started. The range is 1 through 64 pages. This option applies only to TNS objects.

If this attribute is not specified or if the value is too low, the value of MEMPAGES becomes that assigned in the program object file for this process, when compiled.

NAME $*name*

specifies, in the Guardian environment, the process name of this process, as recognized by TACL. This attribute is required.

When you specify the ASSOCPROC attribute to create an OSS persistent process, the NAME attribute must specify the name of an OSH process.

The length limitation is six characters. If you specify the CPU attribute and more than one processor, the NAME value cannot exceed three characters (after the dollar sign). This is because the two-digit processor number is appended to the process name.

OUTFILE { $*device* | [[$*vol*.]*subvol*.]*filename* }

specifies the output file or process name sent to this process (in the startup message) when it is started.

If this attribute is not specified, the process has the same outfile ($YMIOP.#CLCI) that $ZPM has when it starts the process.

If you configure a generic process to use the system console ($OSP on D-series RVUs; $YMIOP.#CLCI or $YMIOP.#CNSL on earlier G-series RVUs), you probably should specify OUTFILE $ZHOME. See Table 6-4.

Table 6-4 Guidelines for Configuring an OUTFILE Value

If the generic process is…	The OUTFILE value should be…
Able to handle errors returned from the home terminal	Set to use the default value, $YMIOP.#CLCI
Not able to handle errors returned from the home terminal	Set to $ZHOME
Configured to be STARTMODE KERNEL or SYSTEM	Set to $ZHOME

For more information about the $ZHOME process, see the *NonStop S-Series Planning and Configuration Guide*, the *NonStop S-Series Operations Guide*, or the *NonStop Operations Guide* (for J- and H-series RVUs).

PFSSIZE *n*

specifies the size in 2048-byte pages of the process file segment (PFS) of this process. The range is 64 through 512 pages.

If this attribute is not specified, the size is calculated based on the PFSSIZE setting in the program object file for this process.

For more information about how and when to use this attribute, see the *Guardian Application Conversion Guide*.

PRIMARYCPU *n*

specifies the processor in which this process starts its primary process. For more information, see "Controlling Where a Generic Process Starts" (page 52) and "Controlling When a Generic Process Starts" (page 52).

You must specify either the CPU or PRIMARYCPU attribute, but not both in the same command.

The variable *n* can be from 0 through the maximum number of processors.

PRIORITY *n*

specifies the priority to use when starting this process. The range is 1 through 199. If this attribute is not specified, the priority is the same as the current SCF session minus 1.

If a process must be higher than 199, it must set its own priority by calling PROCESS_SETINFO_.

PROGRAM [[$*vol*.]*subvol*.]*filename*

specifies the program object file for this process to use when started. The file must be either a file code type 100 (TNS) or 700 (TNS/R) object file. This attribute is required.

If you specify the SYSTEM subvolume, SCF searches for the file first on the SYSTEM subvolume and then on the current SYS*nn* subvolume.

You cannot mix TNS (type 100) and TNS/R (type 700) object code. For example, if the PROGRAM attribute points to a TNS/R object file, then the LIBRARY modifier must also point to a TNS/R library object file.

SAVEABEND { ON | OFF }

specifies whether a saveabend file is created if this process stops abnormally. This attribute overrides the SAVEABEND setting in the PROGRAM file for this process.

ON	Specifies that a saveabend file is created automatically if the process ends abnormally.
OFF	Specifies that a saveabend file is not created automatically if the process ends abnormally.

If this attribute is not specified, the SAVEABEND setting in the program object file for this process is used to determine the setting of this attribute.

STARTMODE { KERNEL | SYSTEM | APPLICATION | MANUAL | DISABLED}

specifies when the process is started. (OSS persistent processes must specify the STARTMODE attribute as MANUAL.) The default value is DISABLED.

KERNEL	The process is started early during a system load or processor reload.
SYSTEM	The process is started as the final stage of system load or processor reload.
APPLICATION	The process is started after the system load or processor reload is finished.
MANUAL	The process can be started by the user any time after system load or processor reload is finished.
DISABLED	The process is not started unless you change the STARTMODE attribute to one of the values listed above.

For more information about using STARTMODE when configuring your own generic process, see "Start Mode Considerations" (page 53).

STARTUPMSG "*text*"

specifies a text message to be sent to the $ZPM persistence manager when the generic process is started. The length of this message can be 1 through 128 characters.

You can use the startup message to handle specification of the backup processor when configuring a process pair. To do that, specify this text, including the less than (<) and greater than (>) symbols:

```
<BCKP-CPU>
```

The value you specify for the BACKUPCPU attribute appears on the StartupMessage line of an INFO PROCESS command (which displays the configured backup processor number). For examples describing how to do this, see "Creating a Generic Process as a Process Pair" (page 59).

If you do not specify the BACKUPCPU attribute, the $ZPM persistence manager ignores (that is, it does not give an error condition for) any <BCKP-CPU> specification in the startup message.

You can also use this command to specify the startup parameters that OSH sends to an OSS application process when OSH launches it as a persistent process. For documentation of possible startup parameters, see the description of osh1 in the *Open System Services Shell and Utilities Reference Manual*. You should not specify any Guardian parameters in the STARTUPMSG attribute when you run an OSS persistent process.

STOPMODE { SPI | STANDARD | SYSMSG }

specifies what method the $ZPM persistence manager should use when aborting the generic process. The default value is STANDARD.

SPI	stops the generic process by sending it a SPI STOP command (as defined in the *SPI Common Extensions Manual*).
STANDARD	stops the generic process by using the PROCESS_STOP_ procedure (as defined in the *Guardian Procedure Calls Reference Manual*).
SYSMSG	stops the generic process by sending it an internal system message.

TYPE { FOXMON | SUBSYSTEM-MANAGER | OTHER }

specifies the type of generic process being added. The default value is OTHER.

FOXMON	specifies that this process is a FOX monitor process, as described in the *ServerNet/FX Adapter Configuration and Management Manual*.You cannot use the ALTER command to change the TYPE to or from FOXMON. Instead, you must delete and re-add the process.
SUBSYSTEM-MANAGER	specifies that this process is a subsystem manager process, and that it participates in the $ZPM reload check-in protocol. Note that $ZZLAN and $ZZFOX are exceptions; their TYPE values are OTHER and FOXMON, respectively.
OTHER	specifies that this process is not a subsystem manager process or FOXMON process.

USERID { *groupname.username* | *groupnum,usernum* }

specifies the creator access ID under which this process executes. USERID must specify an existing operating system user ID.

The user ID of the current SCF session determines what user IDs can be configured:

- For the super ID (255,255), the user ID can be set to any user on the system.
- For any other super-group user (255,*n*), the user ID is set to the user ID of the current SCF session. (This value is the default.)

Considerations

- For information on how to use the ADD PROCESS command, see "Adding a Generic Process" (page 58).
- If you enter a CONFIRM ON command (described in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*) before the ADD command, SCF displays this message in response to a successful ADD command:

  ```
  Add accepted by KERNEL: PROCESS \system.$ZZKRN.#newprocess
  ```

- After adding a generic process, SCF places it in the STOPPED object state. Use the 124) to start the process.
- If you receive a warning message in response to an ADD command, SCF has accepted the command attributes (unless you also receive an error message in response to the same command). To correct or change an attribute (concerning the warning message), use the ALTER command.
- If $ZZKRN cannot find a specified file on $SYSTEM.SYSTEM, it then searches for the file on the current $SYSTEM.SYS*nn*. This search algorithm allows you to change the operating system version yet keep the same attribute values for a process.
- When specifying a file name, avoid including the system name (unless the file must reside on a specific system). Omitting the system name allows the process to be system-independent.
- Wild cards are not supported for the ADD command.

Considerations for OSS Persistent Processes

- The STARTUPMSG attribute can have a maximum length of only 128 characters. If a longer command line is required to start the OSS application process, you can circumvent the 128 character limit by starting a script which includes the longer command line.
- The RUN command for the OSS process must include a '-name' option to set the process name to the same value as the one contained in the ASSOCPROC attribute.
- The ASSOCPROC attribute value, unlike the value for NAME, is not reserved as a process name in the configuration database. You must therefore make sure that no configured or running process has the same name as the one specified in the ASSOCPROC attribute. The system will validate ASSOCPROC at the time an ADD or ALTER command is issued, but you must make sure that no other process is using the name specified in ASSOCPROC attribute before you issue the START command.
- When you specify the CPU attribute using the ALL option or a processor list, make sure that the value of ASSOCPROC attribute, as well as the matching -name value conveyed through the STARTUPMSG attribute, contain no more than three characters in addition to the dollar sign ($) character. You must observe this restriction because the numbering scheme applied to the PROCESS value is also applied to the ASSOCPROC value. When such processes are run, a two-digit number is appended to the name of each instance of the process, indicating the processor in which that instance runs.

  An ENV variable called $ZCPU facilitates this numbering scheme. $ZPM supplies values for this variable when processes are created in different CPUs. In this way, it provides a unique name for every instance of a process. The $ZCPU variable should be included as part of the run command's -name specification, which you can include in either the

STARTUPMSG attribute itself or in a shell script that STARTUPMSG invokes. $ZPM resolves the $ZPU variable only for persistent process objects whose CPU specification is either the keyword ALL or a list of processors.

For details about writing a shell script, see the *Open System Services Management and Operations Guide*.

- If you want the initial environment to be the initial logon state, specify -ls in the the STARTUPMSG attribute or in a shell script that the startup message invokes.
- If you want the OSS persistent process to direct standard input, standard output, or standard error information to TTY devices, the command line in STARTUPMSG must redirect STDN, STDOUT, or STDERR to the appropriate devices.
- The BACKUPCPU attribute is not supported for OSS persistent processes.
- An OSS persistent processes must specify the STARTMODE attribute as MANUAL.
- For OSS persistent processes, the PRIORITY attribute should be allowed to retain its default value.
- An OSH process delays for ten seconds before it launches the OSS process.

Examples

- To add a process specifying only required attributes:

```
-> ADD PROCESS $ZZKRN.MY-OWN-PROCESS,      &
   NAME $GT72,                             &
   PROGRAM $SYSTEM.SYSTEM.NULL,            &
   CPU 3
```

- To configure the $ZZLAN SLSA subsystem manager process in the $SYSTEM.ZSYSCONF.CONFIG system configuration database:

```
-> ADD PROCESS $ZZKRN.#ZZLAN,              &
   AUTORESTART 10,                         &
   HOMETERM $ZHOME,                        &
   INFILE $YMIOP.#CLCI,                    &
   OUTFILE $ZHOME,                         &
   PRIMARYCPU 0,                           &
   BACKUPCPU 1,                            &
   DEFAULTVOL $SYSTEM.SYSTEM,              &
   NAME $ZZLAN,                            &
   PRIORITY 180,                           &
   PROGRAM $SYSTEM.SYSTEM.LANMAN,          &
   STARTMODE KERNEL,                       &
   STARTUPMSG "<BCKP-CPU>",                &
   TYPE OTHER,                             &
   USERID SUPER.SUPER
```

- To configure the $ZZWAN WAN subsystem manager process in the $SYSTEM.ZSYSCONF.CONFIG system configuration database:

```
-> ADD PROCESS $ZZKRN.#ZZWAN,
   AUTORESTART 10,                         &
   HOMETERM $ZHOME,                        &
   INFILE $YMIOP.#CLCI,                    &
   OUTFILE $ZHOME,                         &
   PRIMARYCPU 0,                           &
   BACKUPCPU 1,                            &
   DEFAULTVOL $SYSTEM.SYSTEM,              &
   NAME $ZZWAN,                            &
   PRIORITY 180,                           &
   PROGRAM $SYSTEM.SYSTEM.WANMGR,          &
   STARTMODE KERNEL,                       &
   STARTUPMSG "<BCKP-CPU>,                 &
```

```
   TYPE SUBSYSTEM-MANAGER,                    &
   USERID SUPER.SUPER
```

- To configure an OSS persistent process object that launches a persistent OSH process and persistent OSS application process in CPU 1:

```
->ADD PROCESS $ZZKRN.#OSSAPP,                    &
  NAME $OSH1,                                    &
  AUTORESTART 10,                                &
  PRIMARYCPU 1,                                  &
  STARTMODE MANUAL,                              &
  USERID OSS.APPS,                               &
  PROGRAM $SYSTEM.SYSTEM.OSH, ASSOCPROC $OSS1,   &
  STARTUPMSG "-ls -name /G/0ss1 -p /bin/tail     &
              -f log <- >>out 2>>err"
```

ADD Command for Using ASSIGNs

Use the ADD command to add ASSIGNs for a generic process.

```
ADD [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
          (ASSIGN logical-unit, actual-file-name
         [, create-open-spec]...)
          [, (ASSIGN logical-unit, actual-file-name
            [, create-open-spec]...),...]
```

For details of the ASSIGN attributes, see the *TACL Reference Manual*.

Examples

To add a process specifying ASSIGN attributes, type:

```
->ADD PROCESS $ZZKRN.#GP,         &
  (ASSIGN ABC, ABC)
```

ADD Command for Using PARAMs

Use the ADD command to add PARAMs for a generic process.

```
ADD [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
          (PARAM PARAM-name PARAM-value)
          [, (PARAM PARAM-name PARAM-value)...]
```

For details of the PARAM attributes, see the *TACL Reference Manual*.

Examples

To add a process specifying PARAM attributes, type:

```
-> ADD PROCESS $ZZKRN.#GP,         &
   (PARAM ABC ABC)
```

ADD Command for Using DEFINEs

Use the ADD command to add DEFINEs for a generic process.

```
ADD [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
          (DEFINE DEFINE-name, CLASS CLASS-name,
{DEFINE-attributes})
```

For details of the DEFINE attributes, see the *TACL Reference Manual*.

Examples

To add a process specifying DEFINE attributes, type:

```
-> ADD PROCESS $ZZKRN.#GP,         &
   (DEFINE =ABC,                      &
   CLASS MAP,                           &
   FILE ABC)
```

Considerations

- Before adding an ASSIGN or PARAM or DEFINE, make sure that a generic process exists.
- ASSIGN, PARAM, or DEFINE can be added separately; that is, one at a time.
- You cannot use the same ADD PROCESS command to add both an ASSIGN and a PARAM.
- You cannot specify multiple DEFINEs in the same ADD command.

ALTER Command (Sensitive Command)

Use the ALTER command to change one or more attribute values of an object.

```
ALTER [ / OUT file-spec / ] [ object-spec ]
   [ , attribute-spec ]...
```

object-spec

specifies one of these object type and object name combinations.

object-type	*object-name*
PROCESS	$ZZKRN.*gpname*
SUBSYS	$ZZKRN

attribute-spec

identifies the attribute names and values for the object specified in the command. The specific attributes are described in these subsections.

The ALTER PROCESS command is described in the next subsection. The ALTER SUBSYS command is described on page 101.

ALTER PROCESS Command

Use the ALTER PROCESS command to change one or more attributes of a process controlled by the Kernel subsystem manager process.

```
ALTER [ / OUT file-spec / ]
   PROCESS $ZZKRNgpname  [ , attribute-spec ]...
```

PROCESS $ZZKRN.*gpname*

is the name of a process controlled by the Kernel subsystem manager process. You can omit PROCESS, $ZZKRN, and #*gpname* if you have specified them with an ASSUME command.

attribute-spec

identifies the attribute names and values for the object specified in the command. The attributes are:

```
ASSOCPROC name
AUTORESTART n
BACKUPCPU n
CPU { ALL | FIRST | FIRSTOF (n,n1,...) | n | (n,n1,...) }
DEFAULTVOL $vol[.subvol ]
EXTSWAP { $vol | [[$vol.]subvol.]filename }
HIGHPIN { ON | OFF }
HOMETERM $device[.#subdevice]
INFILE { $device | [[$vol.]subvol.]filename }
LIBRARY [[$vol.]subvol.]filename
MEMPAGES n
NAME $name
OUTFILE { $device | [[$vol.]subvol.]filename }
PFSSIZE n
PRIMARYCPU n
PRIORITY n
PROGRAM [[$vol.]subvol.]filename
SAVEABEND { ON | OFF }
STARTMODE { KERNEL | SYSTEM | APPLICATION | MANUAL | DISABLED }
STARTUPMSG "text"
```

```
STOPMODE { SPI | STANDARD | SYSMSG }
TYPE { FOXMON | SUBSYSTEM-MANAGER | OTHER }
USERID { groupname.username | groupnum,usernum }
```

ASSOCPROC *name*

specifies the name of an OSS application to be run as part of a named generic process, whose persistence is monitored by the Kernel subsystem's persistence-manager process ($ZPM). The OSH that launches the process must also be made persistent. Thus, when you specify the ASSOCPROC attribute, you must also:

- Specify the name of an OSH process in the NAME attribute
- Specify the OSH object file in the PROGRAM attribute.
- Use the STARTUPMSG attribute to specify the startup message that is sent to the OSS application.

Thus, the PROCESS object consists of two distinct persistent processes: the OSH process and the OSS application process. An ABORT, DELETE, or START command issued to this object affects both processes.

AUTORESTART *n*

specifies the number of times that the $ZPM persistence manager attempts to restart this process within a 10-minute interval after an abnormal termination. The process abends or stops by a means other than the ABORT command.

If *n* is 0 (the default), this process is not automatically restarted.

If *n* is 1 through 10, this process is restarted as many as *n* times in 10 minutes.

If *n* is greater than 0 when a processor fails, the processor is reloaded and its previously running generic processes are restarted, but the value of *n* is not decremented. For more information about what conditions decrement the count, see Table 3-5: Effect of Stopping on the Persistence of a Generic Process (page 55).

For more information about using the AUTORESTART attribute, see "Persistence Considerations" (page 54).

BACKUPCPU *n*

specifies the processor in which this process should start its backup process. For more information, see "Controlling Where a Generic Process Starts" (page 52) and "Controlling When a Generic Process Starts" (page 52).

To specify this attribute, you must also specify (or have previously specified) the PRIMARYCPU attribute, but you must not specify the CPU attribute.

The variable *n* can be from 0 through the maximum number of processors (the BACKUPCPU value specified cannot be the same as the PRIMARYCPU value).

You should not specify BACKUPCPU for any OSS persistent process.

CPU { ALL | FIRST | FIRSTOF (*n*,*n1*,...) | *n* | (*n*,*n1*,...) }

specifies one or more processors in which to start this process. For more information, see "Controlling Where a Generic Process Starts" (page 52).

Specifying the CPU attribute clears any earlier configured PRIMARYCPU and BACKUPCPU attributes.

ALL

specifies that an instance of the process be started in all processors, even if a processor is currently not up. If you specify ALL, you must limit the process name specified in this ADD command to one, two, or three alphanumeric characters. A two-digit processor number is appended to the process name.

FIRST

specifies that the process be started in the first available processor.

Because processors 0 and 1 are always the first processors to be loaded, avoid configuring too many generic processes with CPU FIRST because this can lead to an uneven load balance among the processors in the system. OSS persistent processes running in multiple processors cannot have a specification of FIRST.

FIRSTOF (*n*,*n1*,...)

specifies that the process be started in the first available processor in the designated group. If the processor in which a process is configured fails (for example, processor 2), the process automatically starts in the next available processor (for example, processor 3). OSS persistent processes running in multiple processors cannot have a specification of FIRST OF.

n

specifies that the process be started in processor *n*.

(*n*,*n*1,...)

specifies that an instance of the process be started in each specified processor, even if a processor is currently not up. You must limit the process name specified in this ADD command to one, two, or three alphanumeric characters. A two-digit processor number is appended to the process name.

DEFAULTVOL $vol[.subvol]

specifies the default volume and subvolume information sent to this process (in the startup message) when it is started.

If this attribute is not specified, the startup default volume and subvolume is $SYSTEM.NOSUBVOL.

EXTSWAP { $*vol* | [[$*vol*.]*subvol*.]*filename* }

specifies the volume for or name of the swap file for the default extended data segment of this process. This option applies only to TNS objects.

If this attribute is not specified, the operating system selects a swap file or disk location if the process needs it.

HIGHPIN { ON | OFF }

specifies the desired PIN range for the process.

ON (the default) specifies that the process run at a high PIN if both these statements are true:

- The high-PIN bit is enabled in the program file (and in the library file or files, if any).
- A high PIN is available.

The $ZPM persistence manager tries to create this process as a high-PIN process. If no high PIN is available in the specified processor, it tries to create this process as a low-PIN process. If no low PIN is available, the process creation fails.

OFF specifies that the process run at a low PIN, regardless of any other considerations. The $ZPM persistence manager tries to create this process as a low-PIN process. If no low PIN is available, the process creation fails.

HOMETERM $*device*[.#*subdevice*]

specifies the home terminal to use when starting this process.

If this attribute is not specified, the process has the same home terminal ($YMIOP.#CLCI) that $ZPM has when it starts the process.

If you configure a generic process to use the system console ($OSP on D-series RVUs; $YMIOP.#CLCI or $YMIOP.#CNSL on earlier G-series RVUs), you probably should specify HOMETERM $ZHOME. See Table 6-3: Guidelines for Configuring a HOMETERM Value (page 88).

For more information about the $ZHOME process, see the *NonStop S-Series Planning and Configuration Guide*, the *NonStop S-Series Operations Guide*, or the *NonStop Operations Guide* (for J- and H-series RVUs).

INFILE { $*device* | [[$*vol*.]*subvol*.]*filename* }

specifies the input file or process name sent to this process (in the startup message) when it is started.

If this attribute is not specified, the process has the same infile ($YMIOP.#CLCI) that $ZPM has when it starts the process.

Do not specify INFILE $ZHOME.

LIBRARY [[$*vol*.]*subvol*.]*filename*

specifies a library file for this process to use when started. The file must be either a file code type 100 (TNS) or 700 (TNS/R) object file.

If you specify the SYSTEM subvolume, SCF searches for the file first on the SYSTEM subvolume and then on the current SYS*nn* subvolume.

You cannot mix TNS (type 100) and TNS/R (type 700) object code. For example, if the PROGRAM attribute points to a TNS/R object file, then the LIBRARY modifier must also point to a TNS/R library object file.

MEMPAGES *n*

specifies the number of 1024-word pages of user data memory to be allocated to this process after it is started. The range is 1 through 64 pages. This option applies only to TNS objects.

If this attribute is not specified or if the value is too low, the value of MEMPAGES becomes that assigned in the program object file for this process, when compiled.

NAME $*name*

specifies, in the Guardian environment, the process name of this process, as recognized by TACL. This attribute is required.

When you specify the ASSOCPROC attribute to create an OSS persistent process, the NAME attribute must specify the name of an OSH process.

The length limitation is six characters. If you specify the CPU attribute and more than one processor, the NAME value cannot exceed three characters (after the dollar sign). This is because the two-digit processor number is appended to the process name.

OUTFILE { $*device* | [[$*vol*.]*subvol*.]*filename* }

specifies the output file or process name sent to this process (in the startup message) when it is started.

If this attribute is not specified, the process has the same outfile ($YMIOP.#CLCI) that $ZPM has when it starts the process.

If you configure a generic process to use the system console ($YMIOP.#CLCI or $YMIOP.#CNSL on earlier G-series RVUs), you probably should specify OUTFILE $ZHOME. See Table 6-4: Guidelines for Configuring an OUTFILE Value (page 89).

For more information about the $ZHOME process, see the *NonStop S-Series Planning and Configuration Guide*, the *NonStop S-Series Operations Guide*, or the *NonStop Operations Guide* (for J- and H-series RVUs).

PFSSIZE *n*

specifies the size in 2048-byte pages of the process file segment (PFS) of this process. The range is 64 through 512 pages.

If this attribute is not specified, the size is calculated based on the PFSSIZE setting in the program object file for this process.

For more information about how and when to use this attribute, see the *Guardian Application Conversion Guide*.

PRIMARYCPU *n*

specifies the processor in which this process starts its primary process. For more information, see "Controlling Where a Generic Process Starts" (page 52) and "Controlling When a Generic Process Starts" (page 52).

Specifying this attribute clears an earlier configured CPU attribute. You must specify either the CPU or PRIMARYCPU attribute, but not both in the same command.

The variable *n* can be from 0 through the maximum number of processors.

PRIORITY *n*

specifies the priority to use when starting this process. The range is 1 through 199. If this attribute is not specified, the priority is the same as the current SCF session minus 1.

If a process must be higher than 199, it must set its own priority by calling PROCESS_SETINFO_.

PROGRAM [[$*vol*.]*subvol*.]*filename*

specifies the program object file for this process to use when started. The file must be either a file code type 100 (TNS) or 700 (TNS/R) object file. This attribute is required.

If you specify the SYSTEM subvolume, SCF searches for the file first on the SYSTEM subvolume and then on the current SYS*nn* subvolume.

You cannot mix TNS (type 100) and TNS/R (type 700) object code. For example, if the PROGRAM attribute points to a TNS/R object file, then the LIBRARY modifier must also point to a TNS/R library object file.

SAVEABEND { ON | OFF }

specifies whether a saveabend file is created if this process stops abnormally. This attribute overrides the SAVEABEND setting in the PROGRAM file for this process.

ON	Specifies that a saveabend file is created automatically if the process ends abnormally.
OFF	Specifies that a saveabend file is not created automatically if the process ends abnormally.

If this attribute is not specified, the SAVEABEND setting in the program object file for this process is used to determine the setting of this attribute.

STARTMODE { KERNEL | SYSTEM | APPLICATION | MANUAL | DISABLED}

specifies when the process is started. OSS persistent processes must specify the STARTMODE attribute as MANUAL. The default value is DISABLED.

KERNEL	The process is started early during a system load or processor reload.
SYSTEM	The process is started as the final stage of system load or processor reload.
APPLICATION	The process is started after the system load or processor reload is finished.
MANUAL	The process can be started by the user any time after system load or processor reload is finished.
DISABLED	The process is not started unless you change the STARTMODE attribute to one of the values listed above.

For a more complete discussion of using STARTMODE when configuring your own generic process, see "Start Mode Considerations" (page 53).

STARTUPMSG "text"

specifies a text message to be sent to the $ZPM persistence manager when the generic process is started. The length of this message can be 1 through 128 characters.

You can use the startup message to handle specification of the backup processor when configuring a process pair. To do that, specify this text, including the less than (<) and greater than (>) symbols:

```
<BCKP-CPU>
```

The value you specify for the BACKUPCPU attribute is displayed on the StartupMessage line of an INFO PROCESS command (which displays the configured backup processor number). This is more completely described under "Creating a Generic Process as a Process Pair" (page 59).

If you do not specify the BACKUPCPU attribute, the $ZPM persistence manager ignores (that is, it does not give an error condition for) any <BCKP-CPU> specification in the startup message.

You can also use this command to specify the startup parameters that OSH sends to an OSS application process when OSH launches it as a persistent process. For documentation of possible startup parameters, see the description of osh1 in the *Open System Services Shell and Utilities Reference Manual*. Note that you should not specify any Guardian parameters in the STARTUPMSG attribute when you run an OSS persistent process.

STOPMODE { SPI | STANDARD | SYSMSG }

specifies what method the $ZPM persistence manager should use when aborting the generic process. The default value is STANDARD.

SPI	stops the generic process by sending it a SPI STOP command (as defined in the *SPI Common Extensions Manual*).
STANDARD	stops the generic process by using the PROCESS_STOP_ procedure (as defined in the *Guardian Procedure Calls Reference Manual*).
SYSMSG	stops the generic process by sending it an internal system message.

TYPE { FOXMON | SUBSYSTEM-MANAGER | OTHER }

specifies the type of generic process being altered. The default is OTHER.

FOXMON	specifies that this process is a FOX monitor process, as described in the *ServerNet/FX Adapter Configuration and Management Manual*.Note that you cannot alter to or from TYPE FOXMON without first deleting and readding the process.
SUBSYSTEM-MANAGER	specifies that this process is a subsystem manager process, and that it participates in the $ZPM reload check-in protocol. Note that $ZZLAN and $ZZFOX are exceptions; their TYPE values are OTHER and FOXMON, respectively.
OTHER	specifies that this process is not a subsystem manager process or FOXMON process.

USERID { groupname.username | groupnum,usernum }

specifies the creator access ID under which this process executes. USERID must specify an existing operating system user ID.

The user ID of the current SCF session determines what user IDs can be configured:

- For the super ID (255,255), the user ID can be set to any user on the system.
- For any other super-group user (255,*n*), the user ID is set to the user ID of the current SCF session. This is the default.

ALTER PROCESS Considerations

- For a description of how to use the ALTER PROCESS command, see "Altering a Generic Process" (page 64).
- If you enter a CONFIRM ON command (described in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*) before the ADD command, SCF displays this message in response to a successful ADD command:

  ```
  ALTER accepted by KERNEL: PROCESS \system.$ZZKRN.#process
  ```

- Even if you receive a warning message in response to an ALTER command, SCF has accepted the command (unless you also receive another error message in response to the same command). If you need to correct or change an attribute in response to the warning message, use the ALTER command again.
- If $ZZKRN cannot find a specified file on $SYSTEM.SYSTEM, it then searches for the file on the current $SYSTEM.SYS*nn*. This search algorithm allows you to change the operating system version, yet keep the same attribute values for a process.
- When specifying a file name, avoid including the system name (unless the file must reside on a specific system). Omitting the system name allows the process to be system-independent.
- Wild cards are not supported for the ALTER command.

For considerations regarding the configuration of OSS persistent processes, see "Considerations for OSS Persistent Processes" (page 92) and the *Open System Services Management and Operations Guide*.

ALTER PROCESS Example

This example alters a generic process to run in processor 4 instead of processor 3:

```
-> ALTER PROCESS $ZZKRN.MY-OWN-PROCESS, CPU 4
```

ALTER SUBSYS Command

Use the ALTER SUBSYS command to change one or more system attributes of the Kernel subsystem.

```
ALTER [ / OUT file-spec / ]
   SUBSYS $ZZKRN[ , attribute-spec ]...
```

SUBSYS $ZZKRN

is the name of the Kernel subsystem manager process. You can omit SUBSYS and $ZZKRN if you have specified them with an ASSUME command.

attribute-spec

identifies the attribute names and values for the object specified in the command. The attributes are:

```
DAYLIGHT_SAVING_TIME { TABLE | USA66 | NONE }
DESTINATION_CONTROL_TABLE { SMALL | MEDIUM }
POWERFAIL_DELAY_TIME { n | RIDETHRUONLY }
RESIDENT_TEMPLATES [[$SYSTEM.]subvol.]filename
SYSTEM_NAME \sysname
SYSTEM_NUMBER n
TIME_ZONE_OFFSET [ + | - ] [h]h[:mm]
TNSMISALIGN { FAIL | NOROUND }
NONRESIDENT_TEMPLATES [[$SYSTEM.]subvol.]filename
TLE_LIMIT n
AUTO_RETRY_ON_ERROR_654 { OFF | ON }
```

DAYLIGHT_SAVING_TIME { TABLE | USA66 | NONE }

specifies the daylight-saving time algorithm to be used when the system clock is set.

See also the procedure "Changing the System Name or System Number (H-Series RVUs)" (page 37). Changes to this attribute take effect at the next Manual Reload or Hard Reset of the system.

TABLE

specifies that a table of DST (daylight-saving time) transitions is to be loaded at system-load time. To initialize the DST transitions, you use either the ADDDSTTRANSITION command (documented in the *TACL Reference Manual*) or the ADDDSTTRANSITION procedure (documented in the *Guardian Procedure Calls Reference Manual*).

You also must initialize the DST table with at least one DST transition that is less than the current date and time, and at least two DST transitions that are greater than the current date and time before the SETTIME command is entered. For considerations related to use of a DST table, see the *Guardian Procedure Calls Reference Manual*.

USA66

specifies that your location follows the rules set by the United States by the Uniform Time Act of 1966 for daylight-saving time. In the United States, starting from the year 2007, DST begins at 2:00 AM on the second Sunday in March, when clocks are advanced by one hour; DST ends at 2:00 AM on the first Sunday in November, when clocks are set back by one hour.

NONE

specifies that neither TABLE nor USA66 are to be used when system time is set. This is the default.

DESTINATION_CONTROL_TABLE { SMALL | MEDIUM }

specifies the size of the DCT limit on your system. Note that you can use DCT as the keyword in place of DESTINATION_CONTROL_TABLE.

SMALL	sets the limit to 32,767 entries.
MEDIUM	sets the limit to 65,376 entries

If you request a size decrease and the number of logical devices (ldevs) in use exceed the size that you requested, the now out-of-range ldevs remain in use. However, future ldev allocations are restricted to the new limit, and SCF displays this text:

```
DCT entries beyond the requested size (32767) are in use.
Future allocations are now limited to 32767.
```

NONRESIDENT_TEMPLATES [[$SYSTEM.]*subvol*.]*filename*

specifies the location of the Event Management Service (EMS) nonresident template file (file code 839 or 844). The procedure for using this option is described under "Changing EMS Template Files" (page 34). Changes to this attribute take effect immediately. See also the note about changing the location of the EMS template files.

POWERFAIL_DELAY_TIME { *n* | RIDETHRUONLY }

specifies the maximum time, in seconds, that the operating system is allowed to wait prior to initiating a shutdown of operations when system power failure is imminent. Changes to this attribute take effect immediately. See also the procedure "Changing the Power-Failure-to-Shutdown Time Interval" (page 35).

For NonStop S-Series systems, the range of *n* is 0 through 300 seconds. For Integrity NonStop NS-Series systems and for Integrity NonStop BladeSystems, the range of *n* is 0 through 604800 seconds. 604800 seconds equals one week. For NonStop S-Series systems, NonStop NS-Series systems and NonStop BladeSystems, the default is 30 seconds.

NOTE: For UPS-equipped NonStop NS-Series systems and NonStop BladeSystems, HP recommends a longer time than the default of 30 seconds.

For Integrity NonStop NS-Series systems and Integrity NonStop BladeSystems, you can also specify the key word RIDETHRUONLY as a value for this attribute. RIDETHRUONLY allows the operating system to wait indefinitely. The maximum ride-through time for each system will vary, depending on system load, configuration, and the Uninterruptible Power Source (UPS) capability.

NOTE: The power-monitoring capabilities of Integrity NonStop NS-series servers and Integrity NonStop BladeSystems apply only to UPS units supplied by HP and not to site UPS units.

NonStop S-series servers, Integrity NonStop NS-Series servers, and Integrity NonStop BladeSystems have the ability to continue to operate from a power source for some time after the main system power has been removed. In NonStop S-series servers, this power source consists of batteries internal to the enclosure, while in the more modular Integrity NonStop NS-series systems and the Integrity NonStop BladeSystems, the power source consists of a UPS to which various components are connected. You must ensure that the battery capacity for a fully-powered system allows for at least two minutes after OSM initiates the orderly shutdown to allow the disk cache to be flushed to nonvolatile media.

For NonStop S-Series systems, the calculated delay time should be based on the number of internal hardware devices and I/O enclosures that must be kept operational, as well as on the capabilities of the batteries. For Integrity NonStop NS-series systems and Integrity NonStop BladeSystems, the delay time depends on the capacity of the UPS, the number of components connected to it (for example, the CPU module, p-switches and disk drive enclosures), and the thermal attributes of the room.

After a power failure, if the system power has not been restored for some time, the operating system shuts down all system operations in an orderly manner. This is essential to ensure a successful recovery from the power failure.

The actual time the operating system can wait before shutdown is calculated at the time of the power failure. In NonStop S-Series systems, the operating system uses the smaller of the calculated time value and the POWERFAIL_DELAY_TIME value to determine how long it waits before starting to shut down system operations. On Integrity NonStop NS-series servers and Integrity NonStop BladeSystems, the operating system uses the value that the operator specifies as the POWERFAIL_DELAY_TIME value.

You must take care when configuring the POWERFAIL_DELAY_TIME attribute. The operating system continues to process data until *n* seconds expires. Unexpected errors might occur if peripherals used by the system (for example, modems, Ethernet hubs and routers, and tape drives not contained within a system enclosure) are not powered by uninterruptible power supplies (UPSs) during this time.

Refer to the planning guides for the various systems for more information: *NonStop S-Series Planning and Configuration Guide*, the appropriate NonStop NS-series planning guide, or the *NonStop BladeSystem Planning Guide*.

RESIDENT_TEMPLATES [[$SYSTEM.]subvol.]filename

specifies the location of the Event Management Service (EMS) resident template file (file code 839 or 844). The procedure for using this option is described under "Changing EMS Template Files" (page 34). Changes to this attribute take effect immediately.

NOTE: If you change the location of the EMS template files using the ALTER command, the INSTALL^TEMPLATES program permanently changes the location of the EMS template files. As a result, when you next run DSM/SCM, even though the Build and Apply creates new EMS templates, the subsequent system load invokes the EMS templates previously specified to the INSTALL^TEMPLATES program. To use the RTMPLATE and TEMPLATE EMS template files installed in the new SYS*nn* by DSM/SCM, you must use this ALTER command:

```
-> ALTER, RESIDENT_TEMPLATES $SYSTEM.SYSTEM.RTEMPLATE, &
   NONRESIDENT_TEMPLATES $SYSTEM.SYSTEM.TMPLATE
```

SYSTEM_NAME \sysname

specifies the name of the system in an Expand network. Each Expand system name must be unique within the network. The first character is alphabetic; the following six characters are alphanumeric. The default is \NONAME.

For more information, see the *NonStop S-Series Hardware Installation and FastPath Guide*, the appropriate NonStop NS-series hardware installation guide, or the *NonStop BladeSystem Hardware Installation Manual*. Changes to this attribute take effect at the next Manual Reload or Hard Reset of the system.

SYSTEM_NUMBER *n*

specifies the node number of the system in an Expand network. Each Expand node number must be unique within the network. The range is 0 through 254. The default is 254.

For more information, see the *NonStop S-Series Hardware Installation and FastPath Guide*, the appropriate NonStop NS-series hardware installation guide, or the *NonStop BladeSystem Hardware Installation Manual*. Changes to this attribute take effect at the next Manual Reload or Hard Reset of the system.

TIME_ZONE_OFFSET [+ | -] [h]h[:mm]

specifies an offset of standard civil time (SCT) from Greenwich mean time (GMT) in hours and minutes, where *hh* is in the range 00 through 23 hours and *mm* is in the range 0 through 59 minutes. The default is 0. SCT does not include daylight-saving time (DST).

See also the procedure "Changing the System Name or System Number (H-Series RVUs)" (page 37). Changes to this attribute take effect at the next Manual Reload or Hard Reset of the system.

Some examples of time-zone offsets are:

```
TIME_ZONE_OFFSET   0:00                    !London
TIME_ZONE_OFFSET + 01:00                   !Paris
TIME_ZONE_OFFSET + 05:30                   !Bombay
TIME_ZONE_OFFSET + 09:00                   !Tokyo
TIME_ZONE_OFFSET - 05:00                   !New York
TIME_ZONE_OFFSET - 8:00                    !California
```

When you load the system the first time after changing the TIME_ZONE_OFFSET value, you must enter a TACL SETTIME command to correct the system time, which is incorrect by the amount of the change to the TIME_ZONE_OFFSET value.

TNSMISALIGN { FAIL | NOROUND }

controls the behavior of TNS and accelerated TNS programs when compatibility traps from odd-byte misalignment occur.

FAIL

causes the system to immediately generate an INSTRUCTIONFAILURE interrupt in the process when an erroneous odd-byte extended address is used in a context that caused

round-downs on RVUs before G06.17. The interrupt abends the program or sends it into a debugger if the program has no ARMTRAP error handler.

Use the FAIL option for testing and debugging. Do not use it on a production system if the EMS log shows any misalignment events in any programs.

NOROUND

causes the system to complete all misaligned data operations at the given starting address, without round downs or INSTRUCTIONFAILURE traps. This NOROUND behavior matches the behavior of native mode programs. Former TNS cases that caused rounddowns on RVUs before G06.17 are counted and traced in the EMS log.

Some uncorrected old TNS programs depend on round-down, and may misbehave if you specify NOROUND. No INSTRUCTIONFAILURE trap is generated.

The FAIL option affects only some uses of erroneous odd-byte pointers. Other uses of erroneous odd-byte pointers always complete without trap or round down and are never counted or traced in the EMS log.

To use this attribute, see "Changing Data Misalignment Attribute" (page 42).

TLE_LIMIT *n*

specifies the limit on the number of TLEs that a process is allowed to use.

TIME LIST ELEMENT (TLE) is a kernel structure that keeps track of a timer started by a process or the time limit specified by a process when it initiates a nowaited I/O. When the time-limit expires, the NonStop Kernel performs a task. The task performed by the operating system in response to a TLE expiration depends on the type of TLE and its parameter. TLEs are system resources configured during system generation.

As of the J06.09, H06.20, and G06.32.01 RVUs, you can configure a limit on the number of TLEs (Time List Elements) that can be allocated by a process. For previous H and J-series RVUs, there was no such limit and an errant application could consume all, or nearly all, the TLEs.

Currently the maximum number of available TLEs is:

- 3600 per CPU For G-series systems
- 20000 per CPU For H and J-series systems

The value of *n* specified in the ALTER command should be between 1 and the maximum available TLEs. Otherwise, the ALTER command will fail and the following error message will be displayed:

```
KERNEL E00110 The value for TLE_LIMIT for $ZZKRN is out of
the accepted range of 1 to max-available TLE.
```

You cannot alter the TLE limit above maximum avalable TLEs.

To use this attribute, see "Changing the System TLE Limit Attribute" (page 44).

AUTO_RETRY_ON_ERROR_654 *n*

specifies how the system handles situations when a request buffer is modified. As of the J06.09 and H06.20 RVUs, depending on the value of the AUTO_RETRY_ON_ERROR_654 attribute, the system either immediately reports file-system error 654 or retries sending the request up to three times:

- A value of ON configures the system to retry sending the request up to three times whenever the system detects that a request buffer has been modified. The message or I/O operation completes successfully if a subsequent retry delivers the request to the server CPU without indication that the request was modified again during the life span of that retry.

  However, in abnormal situations in which a client repeatedly modifies the request buffer and all retries fail due to request buffer modifications, the system will complete the message with the file-system error 654. Therefore, enabling automatic retries when

request buffers are modified does not mean that the system never reports file-system error 654. This alternative system behavior simply restricts the reporting of file-system error 654 to clients that may modify a request buffer multiple times after starting a new request.

- A value of OFF configures the system to immediately report file-system error 654 whenever the system detects that a request buffer has been modified. OFF is the default setting.

To use this attribute, see "Changing Software Data Integrity Checking" (page 45).

ALTER SUBSYS Considerations

- Wild cards are not supported for the ALTER SUBSYS command.
- If the ALTER command is not successful, SCF returns an error message saying the requested operation was not completely successful. SCF also sends a warning message identifying each failed operation. To correct or change an attribute in response to a warning message, enter another ALTER command.
- If $ZZKRN cannot find a specified file on $SYSTEM.SYSTEM, it searches for the file on the current $SYSTEM.SYS*nn*. This search algorithm allows you to change the operating system version yet keep the same attribute values for a process.
- When specifying a file name, avoid including the system name (unless the file must reside on a specific system). Omitting the system name allows the process to be system-independent.

ALTER SUBSYS Examples

- This example shows how to specify new EMS template files:

```
-> ALTER SUBSYS $ZZKRN,                                &
   RESIDENT_TEMPLATE     $SYSTEM.TEMPLATE.TNEW,      &
   NONRESIDENT_TEMPLATE  $SYSTEM.TEMPLATE.NRTNEW
```

- This example shows how to set the time-zone offset for a location in California:

```
-> ALTER SUBSYS $ZZKRN, TIME_ZONE_OFFSET -8:00
```

- This example shows how to specify that if data misalignment occurs, TNS processes should complete the operation using the unrounded address:

```
-> ALTER SUBSYS $ZZKRN, TNSMISALIGN NOROUND
```

ALTER Command for Using ASSIGNs, PARAMs, and DEFINEs

Use the ALTER command to change one or more attribute values of ASSIGNs, PARAMs, and DEFINEs.

The ALTER command for ASSIGNs, PARAMs, and DEFINEs is similar to the ALTER command for a generic process.

ALTER Command for Using ASSIGNs

Use the ALTER command to alter ASSIGNs for a generic process.

```
ALTER [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
            (ASSIGN logical-unit, {ASSIGN-attributes})
            [, (ASSIGN logical-unit, {ASSIGN-attributes})...]
```

Example

To alter a process specifying ASSIGN attributes, type:

```
-> ALTER PROCESS $ZZKRN.#GP,     &
   (ASSIGN ABC, XYZ)
```

ALTER Command for Using PARAMs

Use the ALTER command to alter PARAMs for a generic process

```
ALTER [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
            (PARAM PARAM-name PARAM-value)
            [, (PARAM PARAM-name PARAM-value)…]
```

Example

To alter a process specifying PARAM attributes, type:

```
-> ALTER PROCESS $ZZKRN.#GP,     &
   (PARAM ABC XYZ)
```

ALTER Command for Using DEFINEs

Use the ALTER command to alter DEFINEs for a generic process

```
ALTER [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
            (DEFINE DEFINE-name, {DEFINE-attributes})
```

Example

To alter a process specifying DEFINE attributes, type:

```
-> ALTER PROCESS $ZZKRN.#GP,     &
   (DEFINE =ABC, FILE XYZ)
```

Considerations

- For an ALTER command on ASSIGN, if the associated physical filename has to be altered, then it should be the first attribute after the assign name itself.
- The ALTER command on ASSIGN, PARAM, or DEFINE parameters related to a generic process alters only the specified ASSIGN, PARAM, or DEFINE attribute, unlike TACL ASSIGN, PARAM, and DEFINE command that replaces the previous parameters with the new definitions.
- Only one of ASSIGN or PARAM or DEFINE can be altered at a time.
- You cannot specify multiple DEFINEs in the same ALTER command.

CONTROL Command (Sensitive Command)

Use the CONTROL command to power down a NonStop S-series server.

CAUTION: For the system power-off procedure and when to use this command, see the *NonStop S-Series Operations Guide* or the *NonStop Operations Guide* (for J- and H-series RVUs).

You must be in interactive mode to use this sensitive command.

```
CONTROL [ /OUT file-spec/ ] SUBSYS $ZZKRN , SHUTDOWN
```

SUBSYS $ZZKRN

is the name of the Kernel subsystem. You can omit SUBSYS and $ZZKRN if you specified them with an ASSUME command.

SHUTDOWN

specifies that the system perform an orderly shutdown of its hardware, prior to unplugging it from the external power supply.

Consideration

Entering this command generates an operator message that reports the command, the time the command was entered, and the group name and user name of the person issuing the command.

Example

This command powers off a NonStop server:

-> CONTROL SUBSYS $ZZKRN, SHUTDOWN

DELETE Command (Sensitive Command)

Use the DELETE command to remove a process object from the Kernel subsystem and from the system configuration database (CONFIG). Only objects that were added with the ADD command can be deleted.

```
DELETE [ / OUT file-spec / ] PROCESS $ZZKRN.gpname
```

PROCESS $ZZKRN*gpname*

is the name of a process controlled by the Kernel subsystem manager process. You can omit PROCESS, $ZZKRN, and #*gpname* if you have specified them with an ASSUME command.

Considerations

- Deleting a Generic Process on page 3-28 describes how to use the DELETE command. Specifically, you must put a generic process in the STOPPED object state before deleting it.
- If you enter a CONFIRM ON command (described in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*) before the ADD command, SCF displays This message in response to a successful ADD command:

  ```
  Delete accepted by KERNEL: PROCESS \system.$ZZKRN.#process
  ```

- It is recommended that you enter a TIMEOUT command (described in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*) to specify a timeout value that is larger than the default 90 seconds when deleting a generic process configured in multiple processors. Several processes can likely be deleted within the 90 second default. But if you delete a generic process that has been configured as a group (by, for example, the CPU ALL attribute), or if you start multiple generic processes (by using a wild card in the ABORT command), more time may be needed.
- You cannot use the DELETE command on the $ZZKRN Kernel subsystem manager process itself.
- Wild-card support is limited to the trailing asterisk (*) for #*gpname*. However, you cannot use an asterisk when deleting a subsystem manager. To delete a process configured with TYPE SUBSYSTEM-MANAGER, you must specify the complete #*gpname*.

Examples

- This example shows how to delete a generic process named #MY-OWN:

  ```
  -> DELETE PROCESS $ZZKRN.
  ```

- This example shows how to delete all generic processes whose names begin with $ZZKRN.#MY*:

  ```
  -> CONFIRM ON
  -> DELETE PROCESS $ZZKRN.
  Delete accepted by KERNEL: PROCESS \EAST.$ZZKRN.#MYPROC
  ```

DELETE Command for Using ASSIGNs, PARAMs, and DEFINEs

Usage of the DELETE command is similar to the DELETE command for a generic process.

- DELETE command for ASSIGN

  ```
  DELETE [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
                ASSIGN (logical-unit [, logical-unit]...)
  ```

- DELETE command for PARAM

```
DELETE [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
            PARAM (PARAM-name [, PARAM-name]...)
```

- DELETE command for DEFINE

```
DELETE [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
            DEFINE (DEFINE-name [, DEFINE-name]...)
```

Examples

- To delete a process specifying ASSIGN, PARAM, and DEFINE attributes, type:

```
-> DELETE PROCESS $ZZKRN.#GP
```

- To delete the ASSIGN, PARAM, or DEFINE attribute of a generic process, type respectively:

```
-> DELETE PROCESS $ZZKRN.#GP, ASSIGN ABC
-> DELETE PROCESS $ZZKRN.#GP, PARAM ABC
-> DELETE PROCESS $ZZKRN.#GP, DEFINE =ABC
```

- To delete multiple ASSIGN or PARAM attributes from a generic process, type respectively:

```
-> DELETE PROCESS $zzkrn.#gp, ASSIGN (ABC1, ABC2, ABC3)
-> DELETE PROCESS $zzkrn.#gp, PARAM (ABC1, ABC2, ABC3)
```

- To delete all the ASSIGNs, PARAMs, or DEFINEs of a generic process, type respectively:

```
-> DELETE PROCESS $ZZKRN.#GP, ASSIGN (*)
-> DELETE PROCESS $ZZKRN.#GP, PARAM (*)
-> DELETE PROCESS $ZZKRN.#GP, DEFINE (*)
```

Consideration

You cannot delete ASSIGNs, PARAMs, and DEFINEs in a single command line.

INFO Command

Use the INFO command to display configuration information, including attribute values, for the specified object. For a generic process, this configuration information displays the attribute values set by its ADD command.

This is a nonsensitive command.

```
INFO [ / OUT file-spec / ] [ object-spec ]
[ , DETAIL | OBEYFORM ]
```

The value of *object-spec* is one of these object type and object name combinations:

object-type	*object-name*
PROCESS	$ZZKRN[.#*gpname*]
SUBSYS	$ZZKRN

The INFO PROCESS command is described in the next subsection. The INFO SUBSYS command is described on page 116.

INFO PROCESS Command

Use the INFO PROCESS command to display configuration information about the specified process.

```
INFO [ / OUT file-spec / ]
   PROCESS $ZZKRN[.#gpname ] [ , DETAIL | OBEYFORM ]
```

PROCESS $ZZKRN

is the name of the Kernel subsystem manager process. You can omit PROCESS and $ZZKRN if you specified them with an ASSUME command.

This form of the command causes the display to show the current values for the Kernel subsystem manager process.

PROCESS $ZZKRN.*gpname*

is the name of a generic process controlled by the $ZZKRN Kernel subsystem manager. You can omit PROCESS, $ZZKRN, and *gpname* if you specified them with an ASSUME command.

This form of the command causes the display to show the configured values for the specified generic process.

DETAIL

causes SCF to display detailed information about the specified process.

OBEYFORM

displays information about the PROCESS object in the format used in an ADD PROCESS command. If you specify OBEYFORM, you cannot specify DETAIL.

INFO PROCESS Consideration

Wild-card support is limited to the trailing asterisk (*) for #*gpname*.

INFO PROCESS Summary Display Format

The format of the summary display for the INFO PROCESS command (without the DETAIL option) is described here. See also the 112.

```
-> INFO PROCESS $ZZKRN.#gpname

 NONSTOP KERNEL - Info PROCESS \system.$ZZKRN.#gpname

Symbolic Name                 *Name  *Autorestart *Program
gpname                         $proc  n            $vol.subv.file
```

An asterisk (*) before an attribute name indicates that its attribute value can be changed with the SCF ALTER command.

These terms appear in the preceding display:

Symbolic Name	The symbolic name of a generic process, as specified in the ADD command. For the Kernel subsystem manager, this name is ZZKRN.
Name	The name of the process, as specified by the NAME attribute of the SCF ADD or ALTER command, and as recognized by TACL. If this generic process is configured in more than one processor, its name ends in *nn*, representing a two-digit processor number.
Autorestart	The persistence count of the generic process; that is, the number of times the $ZPM persistence manager attempts to restart the process in ten minutes if it goes down abnormally. The persistence count is configured by the AUTORESTART attribute of the SCF ADD or ALTER command. If a generic process with a configured AUTORESTART value greater than 0 is aborted by a processor failure, the process is restarted when the processor reloads, and its Autorestart value is not decremented.
Program	The location of the program file for the generic process, as specified in the ADD or ALTER command.

INFO PROCESS Detailed Display Format

The format of the display for the INFO PROCESS command (with the DETAIL option) is described here. See also the "INFO PROCESS Examples" (page 112).

-> INFO PROCESS $ZZKRN.*gpname*, DETAIL

```
NONSTOP KERNEL - Detailed Info Process \EAST.$ZZKRN.#gpname

 *AutoRestart..............n
 *BackupCPU................n
 *CPUList..................n
```

```
*DefaultVolume.............$vol.subv
*ExtSwap...................$vol.subv.file
*HighPIN...................{ ON | OFF }
*HomeTerminal..............$term[.#subdev ]
*InFile....................$vol.subv.file
*Library...................$vol.subv.file
*MemPages..................n
*Name......................$process
*Associate Process Name    $process
*OutFile...................$vol.subv.file
*PFSSize...................n
*PrimaryCPU................n
*Priority..................n
*Program...................$vol.subv.file
*SaveAbend.................{ ON | OFF }
*StartMode.................{KERNEL | SYSTEM | APPLICATION|MANUAL|DISABLED}
*StartupMessage............"text"
*StopMode..................{ SPI | STANDARD | SYSMSG }
*Type......................{ FOXMON | SUBSYSTEM-MANAGER | OTHER }
*UserId....................SUPER.n      (255,n)
```

An asterisk (*) before an attribute name indicates that its attribute value can be changed with the SCF ALTER command.

These terms appear in the preceding display. For more information about each attribute and value, see the "ADD Command (Sensitive Command)" (page 86).

AutoRestart	The persistence count of the generic process; that is, the number of times the $ZPM persistence manager attempts to restart the process in ten minutes if it goes down abnormally. The persistence count is configured by the AUTORESTART attribute of the SCF ADD or ALTER command. If a generic process with a configured AUTORESTART value greater than 0 is aborted by a processor failure, the process is restarted when the processor reloads, and its Autorestart value is not decremented.
BackupCPU	The number of the backup processor in which this process starts its backup process.
CPUList	The processor or processors (from 0 through ALL) in which this process starts. This value is specified by the CPU attribute in the most recent ADD or ALTER command for this process.
DefaultVolume	The default volume and subvolume information sent to this process (in the startup message) when it starts.
ExtSwap	The extended swap file name or disk location for this process to use when it starts.
HighPIN	The desired PIN range for the process.
HomeTerminal	The name of the home terminal used when this process starts.
InFile	The input file information sent to this process (in the startup message) when it starts.
Library	The library file name this process uses when it starts.
MemPages	The number of 1024-word pages of user data memory allocated to this process after it starts.
Name	The name of the process, as specified by the NAME attribute of the SCF ADD or ALTER command, and as recognized by TACL. If this generic process is configured in more than one processor, its name ends in *nn*, representing a two-digit processor number. For persistent process objects, NAME is an OSH process.
Associate Process Name	is the name of an OSS persistent process. This attribute is omitted from the display if no associate process has been defined.
OutFile	The output file information sent to this process (in the startup message) when it starts.
PFSSize	The size, in kilobytes, of the process file segment (PFS) of the process.
PrimaryCPU	The number of the primary processor in which this process starts its primary process.

Priority	The priority of the process when it was configured, or, if it is not currently running, the priority level it will have when it is next started.
Program	The location of the program file for the generic process, as specified in the ADD or ALTER command.
SaveAbend	Whether or not a saveabend file is created if this process stops abnormally.
StartMode	Whether or not the $ZPM persistence manager starts this process automatically at system load or initial processor reload. A process with a MANUAL or DISABLED start mode is not automatically started at system load or initial processor reload.
StartupMessage	The startup text message sent to this process when it starts.
StopMode	What method the $ZPM persistence manager uses when aborting this generic process.
Type	Whether the process is a subsystem manager process, a user-created generic process, or a FOX monitor process.
UserId	The user ID under which this process executes. The default value is the user ID of the current SCF session.

INFO PROCESS Obeyform Display Format

An example below shows the display format for the INFO PROCESS command with the OBEYFORM option.The OBEYFORM option allows an INFO PROCESS command to produce output in the form of an ADD PROCESS command. OBEYFORM makes it easy to create SCF obey files for configuration backup.

```
NONSTOP KERNEL - Detailed Info Process in obeyform \EAST.$ZZKRN.#gpname

ADD PROCESS $ZZKRN.#ZZKRN , &

    AUTORESTART n , &
    BACKUPCPU n , &
    DEFAULTVOL $vol.subv  , & &
    HIGHPIN  { ON | OFF } , &
    HOMETERM $term[.#subdev ] , &
    MEMPAGES n , &
    NAME $process  , &
    PFSSIZE n , &
    PRIMARYCPU n , &
    PRIORITY n , &
    PROGRAM $vol.subv.file , &
    SAVEABEND { ON| OFF }, &
    STARTMODE {KERNEL | SYSTEM | APPLICATION|MANUAL|DISABLED} , &
    STARTUPMESSAGE "text" , &
    STOPMODE { SPI | STANDARD | SYSMSG } , &
    TYPE { FOXMON | SUBSYSTEM-MANAGER | OTHER } , &
    USERID SUPER.n      (255,n)
```

INFO PROCESS Examples

- This example gives current values for the $ZZKRN Kernel subsystem manager process. As shown in the Name column in the display, $ZZKRN is also the name recognized by TACL for the $ZZKRN Kernel subsystem manager process.

```
-> INFO PROCESS $ZZKRN.#ZZKRN

NONSTOP KERNEL - Info PROCESS \EAST.$ZZKRN.#ZZKRN

Symbolic Name             *Name    *Autorestart *Program
ZZKRN                      $ZZKRN  10           $SYSTEM.SYSTEM.OZKRN
```

- This example gives current values for the #TEMP generic process:

```
-> INFO PROCESS $ZZKRN.#TEMP
```

```
NONSTOP KERNEL - Info PROCESS \EAST.$ZZKRN.#TEMP

Symbolic Name             *Name    *Autorestart *Program
TEMP                       $TEMP    0            $SYSTEM.SYSTEM.TEMP
```

- This example displays all the generic processes currently configured on system \WEST:

```
-> INFO PROCESS \WEST.$ZZKRN.#*

NONSTOP KERNEL - Info PROCESS \WEST.$ZZKRN.#*

Symbolic Name            *Name  *Autorestart *Program
CEV-SERVER-MANAGER-P0     $ZCVP0 5            $SYSTEM.SYSTEM.CEVSMX
CEV-SERVER-MANAGER-P1     $ZCVP1 5            $SYSTEM.SYSTEM.CEVSMX
OSM-APPSRVR               $ZOSM  10           $SYSTEM.SYSTEM.APPSRVR
OSM-CIMOM                 $ZCMOM 5            $SYSTEM.SYSTEM.CIMOM
OSM-CONFLH-RD             $ZOLHI 0            $SYSTEM.SYSTEM.TACL
OSM-OEV                   $ZOEV  10           $SYSTEM.SYSTEM.EVTMGR
QIOMON                    $ZMnn  10           $SYSTEM.SYSTEM.QIOMON
ROUTING-DIST              $TSMRD 0            $SYSTEM.SYSTEM.TACL
SP-EVENT                  $ZSPE  10           $SYSTEM.SYSTEM.ZSPE
TSM-SNMP                  $TSMS  5            $SYSTEM.SYSTEM.SNMPAGT
TSM-SRM                   $ZTSM  5            $SYSTEM.SYSTEM.SRM
ZLOG                      $ZLOG  5            $SYSTEM.SYSTEM.EMSACOLL
ZTCP0                     $OSMM0 0            $SYSTEM.SYSTEM.TACL
ZTCP1                     $OSMM1 0            $SYSTEM.SYSTEM.TACL
ZZKRN                     $ZZKRN 10           $SYSTEM.SYSTEM.OZKRN
ZZLAN                     $ZZLAN 10           $SYSTEM.SYSTEM.LANMAN
ZZSTO                     $ZZSTO 10           $SYSTEM.SYSTEM.TZSTO
ZZWAN                     $ZZWAN 10           $SYSTEM.SYSTEM.WANMGR
```

- This example gives detailed configuration information about the $ZZKRN subsystem manager process:

```
-> INFO PROCESS $ZZKRN.#ZZKRN, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \SUN.$ZZKRN.#ZZKRN

 *AutoRestart..............10
 *BackupCPU................1
 *CPU......................Not Specified
 *DefaultVolume............$SYSTEM.SYSTEM
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$ZHOME
 *InFile...................Not Specified
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$ZZKRN
 *OutFile..................$ZHOME
 *PFSSize..................Not Specified
 *PrimaryCPU...............0
 *Priority.................180
 *Program..................$SYSTEM.SYSTEM.OZKRN
 *SaveAbend................ON
 *StartMode................KERNEL
 *StartupMessage...........<BCKP-CPU>
 *StopMode.................STANDARD
 *Type.....................OTHER
 *UserId...................SUPER.SUPER ( 255,255 )
```

- This example gives detailed configuration information about the #TEMP generic process. Because #TEMP was configured to start in all processors (the CPU value is ALL), the name is the configured name ($GP) plus two digits representing the processor number part of the name.

```
-> INFO PROCESS $ZZKRN.#TEMP, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \EAST.$ZZKRN.#TEMP

 *AutoRestart..............10
 *BackupCPU................Not Specified
 *CPU......................ALL
 *DefaultVolume............$SYSTEM.NOSUBVOL
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............Not Specified
 *InFile...................Not Specified
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$GP08
 *OutFile..................Not Specified
 *PFSSize..................Not Specified
 *PrimaryCPU...............Not Specified
 *Priority.................148
 *Program..................$SYSTEM.SYSTEM.O999GP
 *SaveAbend................OFF
 *StartMode................APPLICATION
 *StartupMessage...........Not Specified
 *StopMode.................STANDARD
 *Type.....................OTHER
 *UserId...................SUPER.SUPER ( 255,255 )
```

- This example gives detailed configuration information about the $ZHOME process.

```
-> INFO PROCESS $ZZKRN.#ZHOME, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \SUN.$ZZKRN.#ZHOME

 *AutoRestart..............10
 *BackupCPU................Not Specified
 *CPU......................Not Specified
 *DefaultVolume............$SYSTEM.NOSUBVOL
 *ExtSwap..................Not Specified
 *Highpin..................ON
 *HomeTerminal.............$YMIOP.#CLCI
 *InFile...................$YMIOP.#CLCI
 *Library..................Not Specified
 *MemPages.................Not Specified
 *Name.....................$ZHOME
 *OutFile..................$YMIOP.#CLCI
 *PFSSize..................Not Specified
 *PrimaryCPU...............0
 *Priority.................199
 *Program..................$SYSTEM.SYSTEM.ZHOME
 *SaveAbend................OFF
 *StartMode................KERNEL
 *StartupMessage...........Not Specified
 *StopMode.................STANDARD
 *Type.....................OTHER
 *UserId...................SUPER.SUPER ( 255,255 )
```

- This example gives detailed information about an OSS persistent process object named $OSSAPP that runs the process specified by the ASSOCPROC attribute in a single processor.

```
-> INFO PROCESS $ZZKRN.#OSSAPP, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \VOLCANO.$ZZKRN.#OSSAPP

 *AutoRestart..............5
 *BackupCPU................Not Specified
 *CPU......................Not Specified
```

```
 *DefaultVolume.............$SYSTEM.NOSUBVOL
 *ExtSwap...................Not Specified
 *Highpin...................ON
 *HomeTerminal..............$ZHOME
 *InFile....................Not Specified
 *Library...................Not Specified
 *MemPages..................Not Specified
 *Name......................$OSH1
 *Associated Process Name...$OSS1
 *OutFile...................$YMIOP.#CLCI
 *PFSSize...................Not Specified
 *PrimaryCPU................2
 *Priority..................167
 *Program...................$SYSTEM.SYSTEM.OSH
 *SaveAbend.................OFF
 *StartMode.................MANUAL
 *StartupMessage............-ls -p /bin/sh startmyapp reload
 *StopMode..................STANDARD
 *Type......................OTHER
 *UserId....................SUPER.SUPER ( 255,255 )
```

- This example gives detailed information about an OSS persistent process object named $OSSAPP. $OSSAPP.creates persistent processes in multiple processors.

```
-> INFO PROCESS $ZZKRN.#OSSAPP, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \VOLCANO.$ZZKRN.#OSSAPP

 *AutoRestart..............5
 *BackupCPU.................Not Specified
 *CPU.......................ALL
 *DefaultVolume.............$SYSTEM.NOSUBVOL
 *ExtSwap...................Not Specified
 *Highpin...................ON
 *HomeTerminal..............$ZHOME
 *InFile....................Not Specified
 *Library...................Not Specified
 *MemPages..................Not Specified
 *Name......................$ALLnn
 *Associated Process Name...$APPnn
 *OutFile...................Not Specifified
 *PFSSize...................Not Specified
 *PrimaryCPU................Not Specified
 *Priority..................167
 *Program...................$SYSTEM.SYSTEM.OSH
 *SaveAbend.................OFF
 *StartMode.................MANUAL
 *StartupMessage............-ls -p /bin/sh launchmyapp reload
 *StopMode..................STANDARD
 *Type......................OTHER
 *UserId....................SUPER.SUPER ( 255,255 )
```

- This example shows the INFO PROCESS command with the OBEYFORM option.

```
->INFO PROCESS $ZZKRN, OBEYFORM

 NONSTOP KERNEL - Detailed Info Process in obeyform \EAST.$ZZKRN.#gpname

ADD PROCESS $ZZKRN.#ZZKRN , &
    ASSOCPROC , &
    AUTORESTART 10 , &
    BACKUPCPU 1 , &
    DEFAULTVOL $SYSTEM.SYSTEM , & &
    HIGHPIN ON , &
    HOMETERM $YMIOP.#CLCI , &
    MEMPAGES 64 , &
    NAME $ZZKRN , &
```

```
PFSSIZE 128 , &
PRIMARYCPU 0 , &
PRIORITY 180 , &
PROGRAM $SYSTEM.SYSTEM.OZKRN , &
SAVEABEND OFF , &
STARTMODE KERNEL , &
STARTUPMESSAGE "< BCKP-CPU>" , &
STOPMODE STANDARD , &
TYPE SUBSYSTEM-MANAGER , &
USERID SUPER.SUPER
```

INFO SUBSYS Command

Use the INFO SUBSYS command to display currently configured values for the system parameters managed by the $ZZKRN Kernel subsystem manager.

```
INFO  [ / OUT file-spec / ] SUBSYS $ZZKRN
[ , DETAIL | OBEYFORM ]
```

SUBSYS $ZZKRN

is the name of the Kernel manager process. You can omit SUBSYS and $ZZKRN if you have specified them in an ASSUME command.

DETAIL

is non-functional. You can specify it without receiving an error message. However, if you specify the DETAIL option, the resulting display is the exactly same as what you would receive had you not specified the DETAIL option.

OBEYFORM

displays information about the SUBYS object in the format used in an ALTER SUBSYS command. If you specify OBEYFORM, you cannot specify DETAIL.

INFO SUBSYS Considerations

Wild cards are not supported for the INFO SUBSYS command.

INFO SUBSYS Display Format

The format of the display for the INFO SUBSYS command is described here. See also the example on page 118.

Note that if you use the ALTER command (described on page 95) to change the system name, system number, or a time attribute, this change does not take effect until the next system load.

The INFO SUBSYS command lists any pending changed parameters at the bottom of the display.

```
NONSTOP KERNEL - Info SUBSYS $ZZKRN

*DAYLIGHT_SAVING_TIME.................. { USA66 | TABLE | NONE }
*DESTINATION_CONTROL_TABLE............. { SMALL | MEDIUM }
*NONRESIDENT_TEMPLATES.................
*POWERFAIL_DELAY_TIME   ............... n
*RESIDENT_TEMPLATES.................... $vol.subvol.file
 SUPER_SUPER_IS_UNDENIABLE............. { ON | OFF }
*SYSTEM_NAME........................... \system
*SYSTEM_NUMBER......................... n
 SYSTEM_PROCESSOR_TYPE................. NSR-x
*TIME_ZONE_OFFSET......................
*TNSMISALIGN........................... { FAIL | NOROUND }
*TLE_LIMIT............................. 2000
*AUTO_RETRY_ON_ERROR_654............... OFF

Pending Changes will take effect at next Manual Reload or Hard Reset of the system

*DAYLIGHT_SAVING_TIME.................. {USA66 | TABLE | NONE}
*SYSTEM_NAME........................... \system
*SYSTEM_NUMBER......................... n
*TIME_ZONE_OFFSET...................... [-]hh:mm
```

An asterisk (*) before an attribute name indicates that its attribute value can be changed with the SCF ALTER command.

No asterisk before a name indicates that the parameter corresponds to an ALLPROCESSORS paragraph entry in the CONFTEXT file. For detailed information regarding these parameters, see the *System Generation Manual for G-Series RVUs* and the *DSM/SCM User's Guide*.

These terms apply to the preceding display:

DAYLIGHT_SAVING_TIME	The daylight-saving time algorithm used when the system clock is set: TABLE, USA66, or NONE.
DESTINATION_CONTROL_TABLE	The size of the Destination Control Table: SMALL (32,767 entries) and MEDIUM (65,376). If you used the ALTER command to reduce the size and the reduction is incomplete because a number of logical devices (ldevs) exceeds the requested size, INFO SUBSYS displays the requested size.
NONRESIDENT_TEMPLATES	The location of the EMS nonresident template file. If $ZZKRN cannot find an EMS template file on $SYSTEM.SYSTEM, it then searches for the file on the current $SYSTEM.SYS*nn*.
POWERFAIL_DELAY_TIME	The maximum number of seconds after a power failure that the operating system waits before initiating an orderly shutdown of the system.
RESIDENT_TEMPLATES	The location of the EMS resident template file. If $ZZKRN cannot find an EMS template file on $SYSTEM.SYSTEM, it then searches for the file on the current $SYSTEM.SYS*nn*.
SUPER_SUPER_IS_UNDENIABLE	Whether or not Safeguard security ignores explicit denials of access authority to the super ID (255,255).
SYSTEM_NAME	The name of the system.
SYSTEM_NUMBER	The Expand node number of the system.
SYSTEM_PROCESSOR_TYPE	The type of processor used in the system.
TIME_ZONE_OFFSET	The time offset for the system from Greenwich mean time (GMT).
TNSMISALIGN	How TNS and accelerated TNS programs behave when odd-byte data misalignment occurs.
TLE_LIMIT	Number of TLEs a process is allowed to use.
AUTO_RETRY_ON_ERROR_654	How the system handles situations when a request buffer has been modified.

INFO SUBSYS Obeyform Display Format

The format of the display for the INFO SUBSYS command with the OBEYFORM OPTION is described here.The OBEYFORM option allows an INFO SUBSYS command to produce output in the form of an ALTER SUBSYS command.

```
NONSTOP KERNEL - Info SUBSYS in obeyform $ZZKRN

ALTER SUBSYS $ZZKRN &,
      DAYLIGHT_SAVING_TIME   { USA66 | TABLE | NONE } &
      NONRESIDENT_TEMPLATES $vol.subvol.file , &
      POWERFAIL_DELAY_TIME n , &
      RESIDENT_TEMPLATES $vol.subvol.file , &
      SYSTEM_NAME \system , &
      SYSTEM_NUMBER n , &
      TIME_ZONE_OFFSET [-]hh:mm , &
      TNSMISALIGN { FAIL | NOROUND }, &
      DESTINATION_CONTROL_TABLE { SMALL | MEDIUM } &
      TLE_LIMIT n  &
      AUTO_RETRY_ON_ERROR_654 { ON | OFF }
```

INFO SUBSYS Example

This example displays information about current changes to the operating system:

```
-> INFO SUBSYS $ZZKRN

NONSTOP KERNEL - Info SUBSYS  \EAST.$ZZKRN

Current Settings

*DAYLIGHT_SAVING_TIME ............... USA66
*NONRESIDENT_TEMPLATES............... $SYSTEM.SYSTEM.TEMPLATE
*POWERFAIL_DELAY_TIME................ 30
*RESIDENT_TEMPLATES.................. $SYSTEM.SYSTEM.RTMPLATE
 SUPER_SUPER_IS_UNDENIABLE........... OFF
*SYSTEM_NAME......................... \EAST
*SYSTEM_NUMBER....................... 254
 SYSTEM_PROCESSOR_TYPE ............... NSR-W
*TIME_ZONE_OFFSET.................... -8:00
*TNSMISALIGN......................... NOROUND
*DESTINATION_CONTROL_TABLE........... SMALL
*TLE_LIMIT........................... 2000
*AUTO_RETRY_ON_ERROR_654............. OFF
```

INFO SUBSYS Obeyform Example

This example shows a display for the INFO SUBSYS command with the OBEYFORM option).

```
-> INFO SUBSYS $ZZKRN, OBEYFORM

NONSTOP KERNEL - Info SUBSYS in obeyform $ZZKRN

ALTER SUBSYS $ZZKRN , &
      DAYLIGHT_SAVING_TIME   USA66 , &
      NONRESIDENT_TEMPLATES $SYSTEM.TEMPLATE.NRES , &
      POWERFAIL_DELAY_TIME 30 , &
      RESIDENT_TEMPLATES $SYSTEM.TEMPLATE.RES , &
      SYSTEM_NAME \NCC1701 , &
      SYSTEM_NUMBER 142 , &
      TIME_ZONE_OFFSET -8:00 , &
      TNSMISALIGN NOROUND , &
      DESTINATION_CONTROL_TABLE SMALL &
      TLE_LIMIT 2000 &
      AUTO_RETRY_ON_ERROR_654 OFF
```

INFO Command for Using ASSIGNs, PARAMs, and DEFINEs

Usage of the INFO command is similar to the INFO command for a generic process.

```
INFO  [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname ,
            DETAIL | OBEYFORM]
```

Example

To display configuration information about a process having ASSIGN, PARAM, or DEFINE attribute, type:

```
-> INFO PROCESS $ZZKRN.#GP, DETAIL

NONSTOP KERNEL - Detailed Info PROCESS \MANGO.$ZZKRN.#GP

*AutoRestart..............0
*BackupCPU................Not Specified
*CPU......................Not Specified
*DefaultVolume............$SYSTEM.NOSUBVOL
*ExtSwap..................Not Specified
*Highpin..................ON
```

```
*HomeTerminal.............Not Specified
*InFile...................Not Specified
*Library..................Not Specified
*MemPages.................Not Specified
*Name.....................$GP
*OutFile..................Not Specified
*PFSSize..................Not Specified
*PrimaryCPU...............1
*Priority.................167
*Program..................$SYSTEM.SYSTEM.TACL
*SaveAbend................OFF
*StartMode................MANUAL
*StartupMessage...........Not Specified
*StopMode.................STANDARD
*Type.....................OTHER
*UserId...................SUPER.SUPER (  255,255  )

*Assigns:
   ABC..............................$SYSTEM.SYSTEM.ABC

*Params:
   ABC..............................ABC

*Defines:
   =ABC..............................CLASS MAP, FILE
                                     \MANGO.$SYSTEM.SYSTEM.ABC

-> INFO PROCESS $ZZKRN.#GP, OBEYFORM

     == KERNEL - Obeyform Information PROCESS \MANGO.$ZZKRN.#GP

ADD PROCESS \MANGO.$ZZKRN.#GP , &
    AUTORESTART 0 , &
    DEFAULTVOL $SYSTEM.NOSUBVOL , &
    HIGHPIN ON , &
    NAME $GP , &
    PRIMARYCPU 1 , &
    PRIORITY 167 , &
    PROGRAM $SYSTEM.SYSTEM.TACL , &
    SAVEABEND OFF , &
    STARTMODE MANUAL , &
    STOPMODE STANDARD , &
    TYPE OTHER , &
    USERID SUPER.SUPER

ADD PROCESS \MANGO.$ZZKRN.#GP , &
    ASSIGN ( ABC, $SYSTEM.SYSTEM.ABC )

ADD PROCESS \MANGO.$ZZKRN.#GP , &
    PARAM ( ABC ABC )

ADD PROCESS \MANGO.$ZZKRN.#GP , &
    DEFINE =ABC, CLASS MAP, FILE \MANGO.$SYSTEM.SYSTEM.ABC
```

NAMES Command

Use the NAMES command to display a list of the subordinate object types and names associated with the specified object.

This is a nonsensitive command.

```
NAMES [ / OUT file-spec / ]  [ object-spec ]
```

The value of *object-spec* is one of these object type and object name combinations.

```
object-type       object-name
null              { $ZZKRN | $ZSNET }
PROCESS           $ZZKRN[.#gpname ]
SERVERNET         $ZSNET
SUBSYS            $ZZKRN
```

These versions of the NAMES command are discussed on these pages:

Command	Page
"NAMES null Command "	120
"NAMES PROCESS Command "	121
"NAMES SERVERNET Command "	123
"NAMES SUBSYS Command "	123

NAMES null Command

Use the NAMES *null* command to display a list of subordinate object types and names associated with $ZZKRN or $ZSNET. The NAMES *null* command is described next.

```
NAMES [ / OUT file-spec / ] { $ZZKRN | $ZSNET }
```

{ $ZZKRN | $ZSNET }
: is the $ZZKRN Kernel subsystem manager or the $ZSNET ServerNet manager process. You can omit $ZZKRN or $ZSNET if you have specified either of them with an ASSUME command.

NAMES null Display Formats

The format of the display for the NAMES $ZZKRN command is described here.

```
NONSTOP KERNEL - Names  \system.$ZZKRN
Subsys
$ZZKRN

Process
$ZZKRN.#gpname1                         $ZZKRN.#gpname2
$ZZKRN.#gpname3                         $ZZKRN.#gpname4
...
```

These terms appear in the preceding display:

Subsys	The name of the $ZZKRN Kernel subsystem manager process.
Process	An alphabetized two-column list of the PROCESS objects associated with the $ZZKRN Kernel subsystem manager process.

The format of the display for the NAMES $ZSNET command is described here.

```
NONSTOP KERNEL Names \system.$ZSNET
SERVERNET
$ZSNET
$ZSNET.fabric.cpu0                      $ZSNET.fabric.cpu0
$ZSNET.fabric.cpu1                      $ZSNET.fabric.cpu1
$ZSNET.fabric.cpu2                      $ZSNET.fabric.cpu2
$ZSNET.fabric.cpu3                      $ZSNET.fabric.cpu3
$ZSNET.fabric.cpu4                      $ZSNET.fabric.cpu4
$ZSNET.fabric.cpu5                      $ZSNET.fabric.cpu5
$ZSNET.fabric.cpu6                      $ZSNET.fabric.cpu6
$ZSNET.fabric.cpu7                      $ZSNET.fabric.cpu7
$ZSNET.fabric.cpu8                      $ZSNET.fabric.cpu8
$ZSNET.fabric.cpu9                      $ZSNET.fabric.cpu9
$ZSNET.fabric.cpu10                     $ZSNET.fabric.cpu10
$ZSNET.fabric.cpu11                     $ZSNET.fabric.cpu11
```

```
$ZSNET.fabric.cpu12                    $ZSNET.fabric.cpu12
$ZSNET.fabric.cpu13                    $ZSNET.fabric.cpu13
$ZSNET.fabric.cpu14                    $ZSNET.fabric.cpu14
$ZSNET.fabric.cpu15                    $ZSNET.fabric.cpu15
```

These terms appear in the preceding display:

SERVERNET	A number-ordered two-column list of the SERVERNET objects associated with the $ZZKRN Kernel subsystem manager process, in the form of the $ZSNET process name, an X fabric or Y fabric designation, and a processor number.

NAMES null Examples

1. To list the objects associated with $ZZKRN, type:

```
-> NAMES $ZZKRN

NONSTOP KERNEL - Names PROCESS \SUN.$ZZKRN
Subsys
$ZZKRN

Process
$ZZKRN.#CEV-SERVER-MANAGER-P0      $ZZKRN.#CEV-SERVER-MANAGER-P1
$ZZKRN.#CLCI-TACL                  $ZZKRN.#CHK
$ZZKRN.#OSM-APPSRVR                $ZZKRN.#OSM-CIMOM
$ZZKRN.#OSM-CONFLH-RD              $ZZKRN.#OSM-OEV
$ZZKRN.#QIOMON                     $ZZKRN.#ROUTING-DIST
$ZZKRN.#TCPIP-ZTC02                $ZZKRN.#TSM-SNMP
$ZZKRN.#SP-EVENT                   $ZZKRN.#TSM-SRM
$ZZKRN.#ZLOG                       $ZZKRN.#ZTCP0
$ZZKRN.#ZTCP1                      $ZZKRN.#ZHOME
$ZZKRN.#ZZKRN                      $ZZKRN.#ZZLAN
$ZZKRN.#ZZSTO                      $ZZKRN.#ZZWAN
```

2. To list the objects associated with $ZSNET, type:

```
-> NAMES $ZSNET

NONSTOP KERNEL Names \EAST.$ZSNET
SERVERNET
$ZSNET
$ZSNET.X.0            $ZSNET.Y.0
$ZSNET.X.1            $ZSNET.Y.1
$ZSNET.X.2            $ZSNET.Y.2
$ZSNET.X.3            $ZSNET.Y.3
$ZSNET.X.4            $ZSNET.Y.4
$ZSNET.X.5            $ZSNET.Y.5
$ZSNET.X.6            $ZSNET.Y.6
$ZSNET.X.7            $ZSNET.Y.7
$ZSNET.X.8            $ZSNET.Y.8
$ZSNET.X.9            $ZSNET.Y.9
$ZSNET.X.10           $ZSNET.Y.10
$ZSNET.X.11           $ZSNET.Y.11
$ZSNET.X.12           $ZSNET.Y.12
$ZSNET.X.13           $ZSNET.Y.13
$ZSNET.X.14           $ZSNET.Y.14
$ZSNET.X.15           $ZSNET.Y.15
```

NAMES PROCESS Command

Use the NAMES PROCESS command to display a list of generic processes associated $ZZKRN.

```
NAMES [ / OUT file-spec / ] PROCESS $ZZKRN[.#gpname  ]
```

PROCESS $ZZKRN[.#*gpname*]

is the name of a generic process controlled by the $ZZKRN Kernel subsystem manager. You can omit PROCESS, $ZZKRN, and #*gpname* if you have specified them with an ASSUME command.

NAMES PROCESS Consideration

The trailing asterisk (*) wild-card character is supported for #*gpname*.

NAMES PROCESS Display Format

The format of the display for the NAMES PROCESS command is described here.

```
NONSTOP KERNEL - Names PROCESS \system.$ZZKRN[.#gpname ]
Process
$ZZKRN.#gpname1                          $ZZKRN.#gpname2
$ZZKRN.#gpname3                          $ZZKRN.#gpname4
```

This term appears in the preceding display:

Process	An alphabetized two-column list of the PROCESS objects associated with the $ZZKRN Kernel subsystem manager process.

NAMES PROCESS Examples

1. This example shows how to list the PROCESS objects associated with the Kernel subsystem manager process:

```
-> NAMES PROCESS $ZZKRN

NONSTOP KERNEL - Names PROCESS \EAST.$ZZKRN
Process
$ZZKRN.#CEV-SERVER-MANAGER-P0      $ZZKRN.#CEV-SERVER-MANAGER-P1
$ZZKRN.#CLCI-TACL                  $ZZKRN.#CHK
$ZZKRN.#OSM-APPSRVR                $ZZKRN.#OSM-CIMOM
$ZZKRN.#OSM-CONFLH-RD              $ZZKRN.#OSM-OEV
$ZZKRN.#QIOMON                     $ZZKRN.#ROUTING-DIST
$ZZKRN.#TCPIP-ZTC02                $ZZKRN.#TSM-SNMP
$ZZKRN.#SP-EVENT                   $ZZKRN.#TSM-SRM
$ZZKRN.#ZLOG                       $ZZKRN.#ZTCP0
$ZZKRN.#ZTCP1                      $ZZKRN.#ZHOME
$ZZKRN.#ZZKRN                      $ZZKRN.#ZZLAN
$ZZKRN.#ZZSTO                      $ZZKRN.#ZZWAN
```

2. This example shows how to display the name for the WAN subsystem manager process:

```
-> NAMES PROCESS $ZZKRN.#ZZWAN

NONSTOP KERNEL - Names PROCESS \EAST.$ZZKRN.#ZZWAN
Process
$ZZKRN.#ZZWAN
```

3. This example shows how to list the PROCESS object associated with a user-created generic process:

```
-> NAMES PROCESS $ZZKRN.#GP

NONSTOP KERNEL - Names PROCESS \EAST.$ZZKRN.#GP
Process
$ZZKRN.#GP
```

4. This command produces the same display as in NAMES PROCESS example Step 3:

```
-> NAMES PROCESS $ZZKRN.#G*
```

NAMES SERVERNET Command

Use the NAMES SERVERNET command to display a list of SERVERNET objects associated with the $ZSNET process.

```
NAMES [ / OUT file-spec / ] SERVERNET $ZSNET
```

SERVERNET $ZSNET

is the name of the ServerNet manager process. NAMES SERVERNET Display Format and Example.

The format of the display for the NAMES SERVERNET command is described here and produced by this command:

```
 -> NAMES SERVERNET $ZSNET

NONSTOP KERNEL - Names SERVERNET \system.$ZSNET
SERVERNET
$ZSNET
$ZSNET.X.0            $ZSNET.Y.0
$ZSNET.X.1            $ZSNET.Y.1
$ZSNET.X.2            $ZSNET.Y.2
$ZSNET.X.3            $ZSNET.Y.3
$ZSNET.X.4            $ZSNET.Y.4
$ZSNET.X.5            $ZSNET.Y.5
$ZSNET.X.6            $ZSNET.Y.6
$ZSNET.X.7            $ZSNET.Y.7
$ZSNET.X.8            $ZSNET.Y.8
$ZSNET.X.9            $ZSNET.Y.9
$ZSNET.X.10           $ZSNET.Y.10
$ZSNET.X.11           $ZSNET.Y.11
$ZSNET.X.12           $ZSNET.Y.12
$ZSNET.X.13           $ZSNET.Y.13
$ZSNET.X.14           $ZSNET.Y.14
$ZSNET.X.15           $ZSNET.Y.15
```

This term appears in the preceding display:

SERVERNET	A list of the SERVERNET objects associated with the $ZZKRN Kernel subsystem manager, in the form of the $ZSNET process name, an X fabric or Y fabric designation, and a processor number. All possible processors are listed, regardless of how many are up.

NAMES SUBSYS Command

Use the NAMES SUBSYS command to display a list of PROCESS objects associated with the $ZZKRN process.

```
NAMES [ / OUT file-spec / ] SUBSYS $ZZKRN
```

SUBSYS $ZZKRN

is the name of the ServerNet manager process. You can omit SUBSYS and $ZZKRN if you have specified them in an ASSUME command.

NAMES SUBSYS Consideration

Wild cards are not supported for the NAMES SUBSYS command.

NAMES SUBSYS Display Format

The format of the display for the NAMES SUBSYS command is described here.

```
NONSTOP KERNEL - Names SUBSYS \system.$ZZKRN
Subsys
$ZZKRN
```

```
Process
$ZZKRN.#gpname1              $ZZKRN.#gpname2
$ZZKRN.#gpname3              $ZZKRN.#gpname4
```

These terms appear in the preceding display:

Subsys	The name of the $ZZKRN Kernel subsystem manager process.
Process	An alphabetized two-column list of the PROCESS objects associated with the $ZZKRN Kernel subsystem manager process.

NAMES SUBSYS Example

This example shows how to list the SUBSYS objects associated with the Kernel subsystem manager process:

```
-> NAMES SUBSYS $ZZKRN

NONSTOP KERNEL - Names SUBSYS \BLUE.$ZZKRN
Subsys
$ZZKRN

Process
$ZZKRN.#SSM                  $ZZKRN.#TEMP
$ZZKRN.#XYZZZ                $ZZKRN.#ZZKRN
```

START Command (Sensitive Command)

Use the START command to initiate the operation of an object.

```
START [ / OUT file-spec / ] [ object-spec ]
```

The value of *object-spec* is one of these object type and object name combinations.

```
object-type     object-name
PROCESS         $ZZKRN.#gpname
SERVERNET       $ZSNET.{X|Y}.{cpu|*}
```

The START PROCESS command is described in the next subsection. The START SERVERNET command is described on page 125.

START PROCESS Command

Use the START PROCESS command to start running a generic process.

```
START [ / OUT file-spec / ] PROCESS $ZZKRN.#gpname
```

PROCESS $ZZKRN.*#gpname*

is the name of a generic process controlled by the $ZZKRN Kernel subsystem manager. You can omit PROCESS, $ZZKRN, and *#gpname* if you have specified them with an ASSUME command.

START PROCESS Considerations

- "Starting a Generic Process" (page 62) describes how to use the START PROCESS command.
- It is recommended that you enter a TIMEOUT command (described in the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*) to specify a timeout value that is larger than the default 90 seconds when starting a generic process configured in multiple processors. Several processes can likely be started within the 90 second default. But if you start a generic process that has been configured as a group (by, for example, the CPU ALL attribute), or if you start multiple generic processes (by using a wild card in the ABORT command), more time may be needed.

- The START PROCESS command initiates execution of generic processes. Executing this command directs the $ZPM persistence manager to create one or many processes, depending on the values configured for the CPU, PRIMARYCPU, and BACKUPCPU attributes.
- If the start mode is not DISABLED but the processor in which the generic process is configured is down, the START command puts the process into the STOPPED object state, substate STOPPED. When the processor comes up, the generic process starts.
- If the start mode is not DISABLED and the processor in which the generic process is configured is up, a successful START command puts the process into the STARTED object state.
- A successful completion of the START command indicates that this process (or processes, if started in more than one processor) has been started (if its processor is up) and the startup message has been sent to it.
- The START command sets the persistence count to the configured value.
- Wild-card support is limited to the trailing asterisk (*) for *#gpname*.
- The START PROCESS command is not supported for the $ZZKRN Kernel subsystem manager process. (The $ZPM persistence manager ensures that $ZZKRN remains up at all times.)

Consideration for OSS Persistent Processes

When you START an OSS persistent process object, you start the processes specified in its NAME and ASSOCPROC attributes.

START PROCESS Example

This example shows how to start the process $ZZKRN.#TEMP:

```
-> START PROCESS $ZZKRN.TEMP
```

START SERVERNET Command

Use the START SERVERNET command to start a ServerNet fabric.

```
START [ / OUT file-spec / ]
   SERVERNET $ZSNET.{X|Y}.{cpu |*}
```

$ZSNET.{X|Y}.*{cpu|*}*

is the name of a ServerNet fabric (X or Y) and the processor number (*cpu*). You can omit SERVERNET and $ZSNET if you specified them with an ASSUME command.

START SERVERNET Considerations

- The asterisk (*) wild-card character specifies all available processors.
- The START SERVERNET command tells a specific processor to begin using a specific ServerNet fabric.
- For complete information about using this command when adding an enclosure or upgrading memory, see the *NonStop S-Series Hardware Installation and FastPath Guide*, the appropriate NonStop NS-series hardware installation guide, or the *NonStop BladeSystem Hardware Installation Manual*.

START SERVERNET Display Format

The format of the display for the START SERVERNET command is described here.

```
NONSTOP KERNEL - Start SERVERNET $ZSNET.XorY.cpu
XorY-FABRIC
   TO    0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
FROM
    0    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    1    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    2    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
```

```
 3    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
 4    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
 5    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
 6    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
 7    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
 8    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
 9    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
10    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
11    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
12    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
13    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
14    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
15    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
```

XorY

indicates whether the X or Y fabric is being started.

cpu

is a processor number or an asterisk (*) for starting all processors.

st

indicates the status of the path between the two processors and has one of these values:

DIS	(disabled) A ServerNet fabric is down at the "TO" location. As a result, the path from the "FROM" processor to the "TO" processor is down for receiving, which means that the "TO" processor cannot receive from any other processor or from I/O devices on that fabric. DIS overrides UP and DN.
DN	(down) The path from the FROM processor to the TO processor is down because the path is failing. The FROM processor cannot communicate with the TO processor on that fabric.
<-DOWN	(across a row) The FROM processor is down or nonexistent.
E*nnn*	The ServerNet fabric unexpectedly returned file-system error nnn regarding the path from the FROM processor to the TO processor.
ERROR*nnn*	(across a row) The FROM processor unexpectedly returned file-system error *nnn* to the ServerNet fabric. For information about the file-system error, see the *Guardian Procedure Errors and Messages Manual*.
UNA	(unavailable) The link from the FROM processor to the "TO" processor is down because the TO processor is down or nonexistent. UNA overrides all other values.
UP	The path from the FROM processor to the TO processor is up.

START SERVERNET Examples

1. This example shows the results of starting the ServerNet X fabric in processor 0. Only the first four processors are up in the system.

```
-> START SERVERNET $ZSNET.X.0

NONSTOP KERNEL - Start SERVERNET $ZSNET.X.0
X-FABRIC
   T0    0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
FROM
    0    UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    1    UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    2    UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    3    UP  UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    4  <-DOWN
    5  <-DOWN
    6  <-DOWN
    7  <-DOWN
    8  <-DOWN
    9  <-DOWN
   10  <-DOWN
   11  <-DOWN
   12  <-DOWN
   13  <-DOWN
   15  <-DOWN
```

2. This example shows how to start the ServerNet X fabric in all configured processors in the system:

```
-> START SERVERNET $ZSNET.X.*
```

STATUS Command

Use the STATUS command to display current status information about an object.

This command is nonsensitive.

```
STATUS  [ / OUT file-spec / ] [ object-spec ] [ ,  DETAIL  ]
```

The value of *object-spec* is one of these object type and object name combinations.

```
object-type     object-name
PROCESS         $ZZKRN[.gpname ]
SERVERNET       $ZSNET
SUBSYS          $ZZKRN
```

These versions of the STATUS command are discussed on these pages:

Command	Page
“STATUS PROCESS Command”	127
“STATUS SERVERNET Command ”	132
“STATUS SUBSYS Command ”	134

STATUS PROCESS Command

The STATUS PROCESS command displays current status about a process. The command takes this form.

```
STATUS [ / OUT file-spec / ]
   PROCESS $ZZKRN[.#gpname ] [ , DETAIL ]
```

PROCESS $ZZKRN

is the name of the Kernel subsystem manager process. You can omit PROCESS and $ZZKRN if you specified them with an ASSUME command.

This form of the command causes the display to show current status information for the Kernel subsystem manager.

PROCESS $ZZKRN.#*gpname*

is the name of a generic process controlled by the $ZZKRN Kernel subsystem manager. You can omit PROCESS and $ZZKRN.#*gpname* if you have specified them with an ASSUME command.

This form of the command causes the display to show current status information for the specified process.

DETAIL

causes SCF to display detailed information about the specified process.

STATUS PROCESS Consideration

Wild-card support is limited to the trailing asterisk (*) for #*gpname*.

STATUS PROCESS Summary Display Format

The format of the summary display for the STATUS PROCESS command (without the DETAIL option) is described here.

```
NONSTOP KERNEL - Status Process       \system.$ZZKRN.[#gpname]
Symbolic Name     Name      State    Sub Primary  Backup   Owner
                                        PID      PID      ID
gpname            $process  state    sub n,m      n,m      nnn,nnn
```

These terms appear in the preceding display:

Symbolic Name	The symbolic name of a generic process, as specified in the ADD command. For the Kernel subsystem manager, this name is ZZKRN.
Name	The name of the process, as specified by the NAME attribute of the SCF ADD or ALTER command, and as recognized by TACL. If this generic process is configured in more than one processor with some form of the CPU attribute, its name ends in *nn*, representing a two-digit processor number.
State	The current object state of the PROCESS object. Possible states are ABORTING, STARTED, and STOPPED.
Sub	The ABT (ABORTED) substate, if the PROCESS object has been stopped by the ABORT command.
Primary PID	The number of the primary processor and process identification number (PIN) for this process.
Backup PID	The number of the backup processor and process identification number (PIN) for this process, if it exists.
Owner ID	The owner group number and user number of this process.

STATUS PROCESS Detailed Display Format

The format of the detailed display for the STATUS PROCESS command (with the DETAIL option) is described here.

```
NONSTOP KERNEL - Detailed Status Process \system.gpname Backup PID.........n,m
Creation Time......dd mmm yyyy, hh:mm:ss:ff
Name..............$name
Owner ID...........n,m
Primary PID........n,m
Priority...........n
State..............state
Substate...........substate
```

These terms appear in the preceding display:

Backup PID	The number of the backup processor and process identification number (PIN) for this process, if it exists.
Creation Time	The date and time when the process was created; this means the time it became a process in the operating system; for example, as a result of an SCF START command.
Name	The name of the process, as specified by the NAME attribute of the SCF ADD or ALTER command, and as recognized by TACL. If this generic process is configured in more than one processor with some form of the CPU attribute, its name ends in *nn*, representing a two-digit processor number.
Owner ID	The owner group number and user number of this process.
Primary PID	The number of the primary processor and process identification number (PIN) for this process.
Priority	The current priority of the process or, if it is not currently running, the priority level it will have when it is next started. The process may have been configured with a different priority level than this display indicates. This is possible because the priority could have been changed by an operating system or TACL command, and such a change is reflected by SCF STATUS, but not the SCF INFO command. To find out the configured priority of the process, use the INFO PROCESS, DETAIL command (see page 109).
State	The current object state of the PROCESS object. Possible states are ABORTING, STARTED, and STOPPED.
Substate	ABORTED, if the PROCESS object has been aborted.

STATUS PROCESS Examples

1. This example displays summary current status information for the $ZZKRN Kernel subsystem manager:

```
-> STATUS PROCESS $ZZKRN

NONSTOP KERNEL - Status PROCESS \EAST.$ZZKRN.#ZZKRN

Symbolic Name         Name    State    Sub Primary  Backup   Owner
                                            PID      PID      ID
ZZKRN                 $ZZKRN STARTED      0 ,11    1,11     255,255
```

2. This example displays detailed current status information for $ZZKRN:

```
-> STATUS PROCESS $ZZKRN, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \SUN.$ZZKRN.#ZZKRN
   Backup PID........ None
   Creation Time..... JAN 21,2000 11:41:59
   Name.............. $ZZKRN
   OwnerID........... 255, 255
   Primary PID....... 0 , 15
   Priority.......... 180
   State............. STARTED
   Substate..........
```

3. This example displays summary current status information about a user-configured generic process that is configured in two processors, when one processor is down:

```
-> STATUS PROCESS $ZZKRN.GP

NONSTOP KERNEL - Status PROCESS \EAST.$ZZKRN.#GP

Symbolic Name        Name    State    Sub Primary  Backup   Owner
                                           PID      PID      ID
GP                   $GP00 STARTED       0 ,16    None     255,255
GP                   $GP01 STOPPED       None     None
```

4. This example displays detailed current status information about the same user-configured generic process. Because $GP01 is stopped, the STATUS command cannot display information about the creation time, owner, or priority for the process.

```
-> STATUS PROCESS $ZZKRN.GP, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \EAST.$ZZKRN.#GP

Backup PID........ None
Creation Time..... JAN 21,2000 11:51:50
Name.............. $GP00
OwnerID........... 255, 255
Primary PID....... 0 , 16
Priority.......... 148
State............. STARTED
Substate..........

NONSTOP KERNEL - Detailed Status PROCESS \EAST.$ZZKRN.#GP

Backup PID........ None
Creation Time.....
Name.............. $GP01
OwnerID...........
Primary PID....... None
Priority..........
State............. STOPPED
Substate..........
```

5. This example displays summary current status information about an OSS persistent process object named PROC, whose CPU attribute specifies a list of three processors.

```
-> STATUS PROCESS $ZZKRN.PROC

NONSTOP KERNEL - Status PROCESS \EAST.$ZZKRN.#PROC

Symbolic Name        Name   State    Sub Primary  Backup   Owner
                                         PID      PID      ID
PROC                 $LST00 STARTED  ABT  None     None
PROC                 $LST01 STOPPED  ABT  None     None
PROC                 $LST01 STOPPED  ABT  None     None
PROC                 $ASC00 STARTED  ABT  None     None
PROC                 $ASC01 STOPPED  ABT  None     None
PROC                 $ASC01 STOPPED  ABT  None     None
```

6. This example displays detailed current status information about an OSS persistent process object named OSSAPP. The PRIMARYCPU attribute of $OSSAP has been specified as one (0I). Thus, the processes specified in its NAME ($TDN1) and ASSOCPROC ($TNT1) attributes run in processor 01 exclusively. The first section of the display gives information about the OSH process, and the second section gives information about the OSS application process:

```
-> STATUS PROCESS $ZZKRN.OSSAPP, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \EAST.$ZZKRN.#OSSAPP
Backup PID........ None

Creation Time..... MAY 13,2004 12:14:15
Name.............. $TNT1

OwnerID.......... 255, 255
Primary PID....... 0 , 401
Priority.......... 167
State............. STARTED

Substate..........

Creation Time..... MAY 13,2004 12:14:15
Name.............. $TDN

OwnerID.......... 255, 255
Primary PID....... 0 , 401
Priority.......... 167
State............. STARTED

Substate..........
```

- This example displays detailed status information about a persistent process object named OSSAPP whose CPU attribute specifies ALL. Thus, an instance of both the OSH process and the OSS application process run on each of the system's sixteen processors (0 through 15).

  As in the previous example, status information is listed in sections. First, the information about the OSH process ($ALLnn) is presented in sixteen sequential sections, one section for each instance of the process. Then, the information about the OSS process ($APPnn) is presented in a similar sequence of sections.

  This screen shows the first section in the sequence that gives status for the OSH process ($ALLnn):

```
-> STATUS PROCESS $ZZKRN.OSSAPP, DETAIL

NONSTOP KERNEL - Detailed Status PROCESS \EAST.$ZZKRN.#OSSAPP
Backup PID........ None

Creation Time..... MAY 13,2004 12:14:15
Name.............. $ALL01

OwnerID.......... 255, 255
Primary PID....... 0 , 401
Priority.......... 167
State............. STARTED

Substate..........
```

In the sequence of sixteen sections that follow, status is given for the OSS application ($APP00 through $APP15). The screen below shows the first section of the sequence for the OSS application.

```
NONSTOP KERNEL - Detailed Status PROCESS \EAST.$ZZKRN.#OSSAPP
Backup PID........ None

Creation Time..... MAY 13,2004 12:14:15
Name.............. $APP01
```

```
OwnerID.......... 255, 255
Primary PID....... None
Priority.......... 167
State............. STARTED

Substate..........
```

STATUS SERVERNET Command

The STATUS SERVERNET command displays the current object status of all ServerNet fabrics on the system. The command takes this form.

```
STATUS [ / OUT file-spec / ] SERVERNET $ZSNET
```

SERVERNET $ZSNET
: is the name of the ServerNet manager process. You can omit SERVERNET and $ZSNET if you specified them in an ASSUME command.

STATUS SERVERNET Considerations

- The DIS status overrides the display of UP or DN status.
- The UNA status overrides all other displayed values.

STATUS SERVERNET Display Format

The format of the display for the STATUS SERVERNET command is described here.

```
NONSTOP KERNEL - Status SERVERNET
X-FABRIC
   TO     0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
FROM
    0    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    1    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    2    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    3    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    4    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    5    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    6    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    7    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    8    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    9    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   10    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   11    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   12    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   13    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   14    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   15    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st

Y-FABRIC
   TO     0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
FROM
    0    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    1    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    2    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    3    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    4    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    5    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    6    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    7    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    8    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    9    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   10    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   11    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   12    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
```

```
   13    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   14    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   15    st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
```

The STATUS SERVERNET command displays a matrix for the ServerNet X fabric and Y fabric. Each matrix shows the status of the path between all pairs of processors. The status (*st*) can be:

DIS	(disabled) A ServerNet fabric is down at the "TO" location. As a result, the path from the "FROM" processor to the "TO" processor is down for receiving, which means that the "TO" processor cannot receive from any other processor or from I/O devices on that fabric. DIS overrides UP and DN.
DN	(down) The path from the FROM processor to the TO processor is down because the path is failing. The FROM processor cannot communicate with the TO processor on that fabric.
<-DOWN	(across a row) The FROM processor is down or nonexistent.
ERROR*nnn*	(across a row) The FROM processor unexpectedly returned file-system error *nnn* to the ServerNet fabric. For information about the file-system error, see the *Guardian Procedure Errors and Messages Manual*.
UNA	(unavailable) The link from the FROM processor to the "TO" processor is down because the TO processor is down or nonexistent. UNA overrides all other values.
UP	The path from the FROM processor to the TO processor is up.

STATUS SERVERNET Example

This example displays the status of the ServerNet network in a four-processor system:

```
-> STATUS SERVERNET $ZSNET

NONSTOP KERNEL - Status SERVERNET
X-FABRIC
   TO    0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
FROM
    0    UP  DN  UNA UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    1    UP  UP  UNA UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    2  <-DOWN
    3    UP  UP  UNA UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    4  <-DOWN
    5  <-DOWN
    6  <-DOWN
    7  <-DOWN
    8  <-DOWN
    9  <-DOWN
   10  <-DOWN
   11  <-DOWN
   12  <-DOWN
   13  <-DOWN
   15  <-DOWN
Y-FABRIC
   TO    0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
FROM
    0    UP  UP  UNA DIS UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    1    UP  UP  UNA DIS UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    2  <-DOWN
    3    DN  DN  UNA DIS UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    4  <-DOWN
    5  <-DOWN
    6  <-DOWN
    7  <-DOWN
    8  <-DOWN
    9  <-DOWN
   10  <-DOWN
   11  <-DOWN
   12  <-DOWN
   13  <-DOWN
```

```
14   <-DOWN
15   <-DOWN
```

The display shows that:

- Processor 2 is down, as shown by the UNA status in column 2 and the DOWN in row 2 for both fabrics (shown in bold).
- A single point-to-point link (the X fabric from processor 0 to processor 1) is down, as shown by the DN status in the X fabric matrix (shown in bold).
- The Y fabric is totally down in processor 3, as shown by the DIS status in column 3 of the Y fabric and the DN status in row 3 of the Y fabric.

STATUS SUBSYS Command

The STATUS SUBSYS command displays current status information about the Kernel subsystem manager. The command takes this form.

```
STATUS [ / OUT file-spec / ] SUBSYS $ZZKRN [ , DETAIL ]
```

SUBSYS $ZZKRN
: is the name of the Kernel subsystem manager process. You can omit SUBSYS and $ZZKRN if you have specified them with an ASSUME command.

DETAIL
: causes SCF to display detailed information about the specified process.

STATUS SUBSYS Summary Display Format

The format of the summary display for the STATUS SUBSYS command (without the DETAIL option) is described here.

```
NONSTOP KERNEL -  Status SUBSYS \system.$ZZKRN

Name             State    Processes
                          (conf/strd)
$ZZKRN           state    (   n/m   )
```

These terms appear in the preceding display:

Name	$ZZKRN, the name of the Kernel subsystem manager process.
State	The current object state of $ZZKRN is always STARTED.
Processes (conf/strd)	The number of generic process records or groups configured and the number with at least one instance of the process started. To display the names of these processes, use the INFO PROCESS (on page 109) or NAMES PROCESS command (on page 121).

STATUS SUBSYS Detailed Display Format

The format of the detailed display for the STATUS SUBSYS command (with the DETAIL option) is described here.

```
NONSTOP KERNEL - Detailed Status SUBSYS \system.$ZZKRN

 Primary PID.......... n,m       Backup PID ........ n,m
 Subsystem Owner...... n,m       Subsystem State ... state
 Processes Configured. n         Processes Started.. n
```

These terms apply to the preceding display:

Primary PID	The number of the primary processor and process identification number (PIN) for the $ZZKRN process.
Backup PID	The number of the backup processor and process identification number (PIN) for the $ZZKRN process.

Subsystem Owner	The owner group number and user number of the $ZZKRN process is always the super ID (255,255).
Subsystem State	The current object state of $ZZKRN is always STARTED.
Processes Configured	The number of generic process currently configured.
Processes Started	The number of generic processes that currently have at least one instance of the generic process started.

STATUS SUBSYS Examples

1. This example displays the status of the $ZZKRN subsystem manager process. Of 12 processes that have been configured, 10 are started.

```
-> STATUS SUBSYS $ZZKRN

NONSTOP KERNEL -  Status SUBSYS \EAST.$ZZKRN

Name             State    Processes
                          (conf/strd)
$ZZKRN           STARTED  (  12/10  )
```

2. This example displays the detailed status of the same $ZZKRN subsystem manager process as in the preceding example:

```
-> STATUS SUBSYS $ZZKRN, DETAIL

NONSTOP KERNEL - Detailed Status SUBSYS \EAST.$ZZKRN

 Primary PID.......... 1 , 51    Backup PID ........ 0 , 51
 Subsystem Owner...... 255, 255  Subsystem State ... STARTED
 Processes Configured. 12        Processes Started.. 10
```

STOP Command (Sensitive Command)

The STOP command terminates the activity of a ServerNet fabric in a normal manner. Upon the successful completion of the STOP command, the ServerNet fabric is left in the STOPPED object state.

You must be in interactive mode to use this sensitive command.

```
STOP [ / OUT file-spec / ] SERVERNET $ZSNET.{X|Y}.{cpu|*}
```

SERVERNET $ZSNET.{X|Y}.*{cpu|*}*

stops a ServerNet fabric (X or Y) in one or all processors. The variable *cpu* is the processor number.

Considerations

- The asterisk (*) wild-card character in a STOP SERVERNET command specifies all available processors.
- You cannot stop both ServerNet fabrics at the same time. If you attempt to do this, the operating system keeps up the last path from processor 0 to processor 1 in the second fabric brought down (shown in Example Step 2).
- For complete information about using this command when adding an enclosure or upgrading memory, see the *NonStop S-Series Hardware Installation and FastPath Guide*, the appropriate NonStop NS-series hardware installation guide, or the *NonStop BladeSystem Hardware Installation Manual*.
- Because of the serious effects of this command, the operating system responds to a STOP SERVERNET command with a confirmation request.

Display Format

The format of the display for the STOP SERVERNET command is described here.

```
NONSTOP KERNEL - Stop SERVERNET $ZSNET.XorY.cpu
XorY-FABRIC
   TO     0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
FROM
    0     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    1     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    2     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    3     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    4     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    5     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    6     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    7     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    8     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
    9     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   10     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   11     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   12     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   13     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   14     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
   15     st  st  st  st  st  st  st  st  st  st  st  st  st  st  st  st
```

XorY

indicates whether the X or Y fabric is being stopped.

cpu

is a processor number or an asterisk (*) for stopping all processors.

st

indicates the status of the path between the two processors and has one of these values:

DIS	(disabled) A ServerNet fabric is down at the “TO” location. As a result, the path from the “FROM” processor to the “TO” processor is down for receiving, which means that the “TO” processor cannot receive from any other processor or from I/O devices on that fabric. DIS overrides UP and DN.
DN	(down) The path from the FROM processor to the TO processor is down because the path is failing. The FROM processor cannot communicate with the TO processor on that fabric.
<-DOWN	(across a row) The FROM processor is down or nonexistent.
E*nnn*	The ServerNet fabric unexpectedly returned file-system error nnn regarding the path from the FROM processor to the TO processor.
ERROR*nnn*	(across a row) The “FROM” processor unexpectedly returned file-system error *nnn* to the ServerNet fabric. For information about the file-system error, see the *Guardian Procedure Errors and Messages Manual*.
UNA	(unavailable) The link from the FROM processor to the “TO” processor is down because the TO processor is down or nonexistent. UNA overrides all other values.
UP	The path from the FROM processor to the TO processor is up.
UP*	The path from the “FROM” processor to the “TO” processor was left up in order not to bring down the last path between these two processors (STOP command only).

Examples

1. This example shows how to stop the Y fabric in processor 0:

```
-> STOP SERVERNET $ZSNET.Y.0
```

This command brings down the connection from processor 0 to the ServerNet Y fabric. As a result, the display shows the column for processor 0 to be UNA, while the row for processor 0 is DN. These changes are shown in bold.

```
NONSTOP KERNEL - Stop SERVERNET $ZSNET.Y.0
Y-FABRIC
   TO    0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
FROM
    0    UNA DN  DN  DN  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    1    UNA UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    2    UNA UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    3    UNA UP  UP  UP  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    4  <-DOWN
    5  <-DOWN
    6  <-DOWN
    7  <-DOWN
    8  <-DOWN
    9  <-DOWN
   10  <-DOWN
   11  <-DOWN
   12  <-DOWN
   13  <-DOWN
   15  <-DOWN
```

2. This example shows the results of taking down the Y fabric on a system when the X fabric is already down. Note that because the X fabric is already down, the “UP*” status (shown here in bold) is maintained for the point-to-point link from processor 0 to processor 1 so as to not bring down the last path between the two processors.

```
-> STOP SERVERNET $ZSNET.Y.*

NONSTOP KERNEL - Stop SERVERNET $ZSNET.Y.*
Y-FABRIC
   TO    0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
FROM
    0    DN  UP* DN  DN  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    1    DN  DN  DN  DN  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    2    DN  DN  DN  DN  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    3    DN  DN  DN  DN  UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA UNA
    4  <-DOWN
    5  <-DOWN
    6  <-DOWN
    7  <-DOWN
    8  <-DOWN
    9  <-DOWN
   10  <-DOWN
   11  <-DOWN
   12  <-DOWN
   13  <-DOWN
   15  <-DOWN

UP* indicates that the path was not brought down because
    it is the last path up between these two processors
```

VERSION Command

Use the VERSION command to display the operating system version level of the specified object.

This is a nonsensitive command.

```
VERSION [ / OUT file-spec / ]  [ object-spec ] [ , DETAIL ]
```

The value of *object-spec* is one of these object type and object name combinations.

```
object-type       object-name   null              $ZZKRN
PROCESS           $ZZKRN
```

```
SERVERNET          $ZSNET
SUBSYS             $ZZKRN
```

These versions of the VERSION command are discussed on these pages:

Command	Page
"VERSION null and VERSION PROCESS Commands "	138
"VERSION SERVERNET Command "	139
"VERSION SUBSYS Command "	140

VERSION null and VERSION PROCESS Commands

The VERSION null command and the VERSION PROCESS command both display the version level of the $ZZKRN subsystem manager process.

```
VERSION [ / OUT file-spec / ] [ PROCESS ] $ZZKRN [ , DETAIL ]
```

VERSION null and VERSION PROCESS Summary Display Format

The format of the summary display for the VERSION null and VERSION PROCESS commands (without the DETAIL option) is described here. See also Example 1 on page 138.

```
VERSION [ PROCESS ] \system.$ZZKRN: KERNEL (MGR)  -  T1085vff - (ddmmmyy) - (ddmmmyy)
```

VERSION null and VERSION PROCESS Detailed Display Format

The format of the detailed display for the VERSION null and VERSION PROCESS commands (with the DETAIL option) is described here. See also Example 2 on page Step 2.

```
Detailed VERSION PROCESS \system.$ZZKRN
  SYSTEM \system
    KERNEL (MGR)  -   T1085vff - (ddmmmyy) - (ddmmmyy)
    GUARDIAN - T9050 - (tos)
    SCF KERNEL - T9082vff - (ddmmmyy) (ddmmmyy)
    KERNEL PM - T1084vff - (ddmmmyy) - (ddmmmyy)
```

These terms appear in the preceding two displays:

\system
: is the system $ZZKRN is running on.

vff
: is the product version; for example, F40.

tos
: is the operating system version of the Guardian kernel for this RV; for example, N40.

ddmmmyy
: are the RVU and compile dates, respectively, for the product.

VERSION null and VERSION PROCESS Examples

1. This example shows how to display version information about the $ZZKRN subsystem manager process:

   ```
   -> VERSION PROCESS $ZZKRN

   VERSION PROCESS \EAST.$ZZKRN: KERNEL (MGR) - T1085F40 - (31AUG99) - (31AUG99)
   ```

2. This example shows how to display detailed version information about the $ZZKRN subsystem manager process:

```
-> VERSION $ZZKRN, DETAIL

Detailed VERSION \EAST.$ZZKRN
  SYSTEM \EAST
    KERNEL (MGR)  -   T1085F40 - (31AUG99) - (28MAY99)
    GUARDIAN - T9050 - (N40)
    SCF KERNEL - T9082F40 - (01AUG97) (23JUL97)
    KERNEL PM - T1084F40 - (31AUG99) - (28MAY99)
```

VERSION SERVERNET Command

The VERSION SERVERNET command displays the version level of the $ZSNET ServerNet manager process.

```
VERSION [ / OUT file-spec / ] SERVERNET $ZSNET [ , DETAIL ]
```

VERSION SERVERNET Summary Display Format

The format of the summary display for the VERSION SERVERNET command (without the DETAIL option) is described here.

```
VERSION \system.$ZSNET: SERVERNET (MGR) - T1085vff - (ddmmmyy) - (ddmmmyy)
```

VERSION SERVERNET Detailed Display Format

The format of the detailed display for the VERSION SERVERNET command (with the DETAIL option) is described here.

```
Detailed VERSION \system.$ZSNET
  SYSTEM \system
    SERVERNET (MGR) - T1085vff - (ddmmmyy) - (ddmmmyy)
    GUARDIAN - T9050 - (tos)
    SCF KERNEL - T9082vff - (ddmmmyy) (ddmmmyy)
    KERNEL PM - T1084vff - (ddmmmyy) - (ddmmmyy)
```

These terms appear in the preceding two displays.

\system
: is the system $ZSNET is running on.

vff
: is the product version; for example, F40.

tos
: is the operating system version of the Guardian kernel for this RV; for example, N40.

ddmmmyy
: are the RVU and compile dates, respectively, for the product.

VERSION SERVERNET Examples

1. This example shows how to display the version of the $ZSNET ServerNet manager process:

```
-> VERSION SERVERNET $ZSNET

VERSION \EAST.$ZSNET: SERVERNET (MGR) - T1085F40 - (31AUG99) - (28MAY99)
```

2. This example shows how to display detailed version information about the $ZZKRN Kernel subsystem manager:

```
-> VERSION SERVERNET $ZSNET, DETAIL

Detailed VERSION \EAST.$ZSNET
  SYSTEM \EAST
    SERVERNET (MGR) - T1085F40 - (31AUG99) - (28MAY99)
    GUARDIAN - T9050 - (N40)
    SCF KERNEL - T9082F40 - (01AUG97) (23JUL97)
    KERNEL PM - T1084F40 - (31AUG99) - (28MAY99)
```

VERSION SUBSYS Command

The VERSION SUBSYS command displays the version level of the $ZZKRN Kernel subsystem manager process.

```
VERSION [ / OUT file-spec / ] SUBSYS $ZZKRN [ , DETAIL ]
```

VERSION SUBSYS Summary Display Format

The format of the summary display for the VERSION SUBSYS command (without the DETAIL option) is described here.

```
VERSION SUBSYS \system.$ZZKRN: KERNEL (MGR) - T10859vff - (ddmmmyy) - (ddmmmyy)
```

VERSION SUBSYS Detailed Display Format

The format of the detailed display for the VERSION SUBSYS command (with the DETAIL option) is described here.

```
Detailed VERSION SUBSYS \system.$ZZKRN
  SYSTEM \system
    KERNEL (MGR)  -   T1085vff - (ddmmmyy) - (ddmmmyy)
    GUARDIAN - T9050 - (tos)
    SCF KERNEL - T9082vff - (ddmmmyy) (ddmmmyy)
    KERNEL PM - T1084vff - (ddmmmyy) - (ddmmmyy)
```

These terms appear in the preceding two displays:

\system
: is the system $ZZKRN is running on.

vff
: is the product version; for example, F40.

tos
: is the operating system version of the Guardian kernel for this RV; for example, N40.

ddmmmyy
: are the RVU and compile dates, respectively, for the product.

VERSION SUBSYS Examples

1. This example shows how to display the version of the $ZZKRN subsystem manager process:

```
-> VERSION SUBSYS $ZZKRN

VERSION SUBSYS \EAST.$ZZKRN: KERNEL (MGR)  -  T1085F40 - (31AUG99) - (28MAY99)
```

2. This example shows how to display detailed version information about the $ZZKRN subsystem manager process:

```
-> VERSION SUBSYS $ZZKRN, DETAIL

Detailed VERSION SUBSYS \EAST.$ZZKRN
  SYSTEM \EAST
    KERNEL (MGR) - T1085F40 - (31AUG99) - (28MAY99)
    GUARDIAN - T9050 - (N40)
    SCF KERNEL - T9082F40 -   (01AUG97) (23JUL97)
    KERNEL PM - T1084F40 - (31AUG99) - (28MAY99)
```

A SCF Command Summary for the Kernel Subsystem

```
ABORT [ /OUT file-spec / ] PROCESS $ZZKRN.gpname

ADD [ / OUT file-spec / ]
  PROCESS $ZZKRN.#gpname  [ , attribute-spec ]...

ADD [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
         (ASSIGN logical-unit, actual-file-name
        [, create-open-spec]...)
         [, (ASSIGN logical-unit, actual-file-name
           [, create-open-spec]...),...]

ADD [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
         (PARAM PARAM-name PARAM-value)
         [, (PARAM PARAM-name PARAM-value)...]

ADD [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
         (DEFINE DEFINE-name, CLASS CLASS-name,
{DEFINE-attributes})

ALTER [ / OUT file-spec / ]
   PROCESS $ZZKRN.#gpname [ , attribute-spec ]...

ALTER [ / OUT file-spec / ]
   SUBSYS $ZZKRN [ , attribute-spec ]...

ALTER [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
            (ASSIGN logical-unit, {ASSIGN-attributes})
            [, (ASSIGN logical-unit, {ASSIGN-attributes})...]

ALTER [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
            (PARAM PARAM-name PARAM-value)
            [, (PARAM PARAM-name PARAM-value)…]

ALTER [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
            (DEFINE DEFINE-name, {DEFINE-attributes})

CONTROL [ /OUT file-spec / ] SUBSYS $ZZKRN , SHUTDOWN

DELETE [ / OUT file-spec / ] PROCESS $ZZKRN.gpname

DELETE [ /OUT file-spec /] PROCESS $ZZKRN.#gpname,
             ASSIGN (logical-unit [, logical-unit]...)

DELETE [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
             PARAM (PARAM-name [, PARAM-name]...)

DELETE [ /OUT file-spec / ] PROCESS $ZZKRN.#gpname,
             DEFINE (DEFINE-name [, DEFINE-name]...)

INFO [ / OUT file-spec / ]
   PROCESS $ZZKRN[.#gpname ] [ OBEYFORM , DETAIL ]

INFO [ / OUT file-spec / ]
    SUBSYS $ZZKRN [ OBEYFORM , DETAIL ]
```

```
LISTOPENS [/ OUT file-spec / ] SUBSYS $ZZNSK

NAMES [ / OUT file-spec / ] { $ZZKRN | $ZSNET }

NAMES [ / OUT file-spec / ] PROCESS $ZZKRN[.#gpname ]

NAMES [ / OUT file-spec / ] SERVERNET $ZSNET

NAMES [ / OUT file-spec / ] SUBSYS $ZZKRN

START [ / OUT file-spec / ] PROCESS $ZZKRN.#gpname

START [ / OUT file-spec / ]
   SERVERNET $ZSNET.{X|Y}.{cpu|*}

STATUS  [ / OUT file-spec / ]
   PROCESS $ZZKRN[.#gpname ]  [ , DETAIL ]

STATUS  [ / OUT file-spec / ] SERVERNET $ZSNET

STATUS  [ / OUT file-spec / ]  SUBSYS $ZZKRN   [ , DETAIL ]

STOP [ / OUT file-spec / ] SERVERNET $ZSNET.{X|Y}.{cpu|*}

VERSION [ / OUT file-spec / ] [ PROCESS ] $ZZKRN [ , DETAIL ]

VERSION [ / OUT file-spec / ] SERVERNET $ZSNET  [ , DETAIL ]

VERSION [ / OUT file-spec / ] SUBSYS $ZZKRN  [ , DETAIL ]
```

B SCF Kernel Subsystem Error Messages

This section describes two types of SCF error messages that apply to the Kernel subsystem:

- Messages with positive error numbers are generated by the Kernel subsystem. These messages begin on page 145.
- Messages with negative error numbers, also known as common error messages, are general SCF messages, but they are documented here because they have additional cause and recovery information specific to the Kernel subsystem. These messages begin on page 175.

For the list of SCF messages for all subsystems, see the *SCF Reference Manual for G-Series RVUs* or the *SCF Reference Manual for J-Series and H-Series RVUs*.

If You Have to Call Your Service Provider

If the recovery for an error message indicates that you should contact your service provider, you should be prepared to supply a log file with this information. (If the error caused SCF to terminate, reenter SCF.)

1. Enter a LOG command to collect the following displays into a single file; for example

   ```
   -> LOG PROBLEM !
   ```

2. Enter a LISTPM command to collect information about the product versions of the SCF components, a list of the product modules on your system, and information about any product modules running when the error occurred; for example

   ```
   -> LISTPM
   ```

3. Enter an ENV command to collect information about the SCF environment that was present when the error occurred; for example

   ```
   -> ENV
   ```

   If the error caused SCF to terminate, respecify any environmental characteristics that were present when the error occurred.

4. Enter DETAIL CMDBUFFER and DETAIL RSPBUFFER commands to capture the contents of the SPI buffer; for example

   ```
   -> DETAIL CMDBUFFER ON
   -> DETAIL RSPBUFFER ON
   ```

5. Reproduce the sequence of commands that produced the SCF error.
6. Close the log file:

   ```
   -> LOG
   ```

SCF Error Messages

00001

`KRN 00001 Too many object names. Object name` *`objname`*`.`

`objname`

is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause Too many object names have been specified in the request buffer.

Effect SCF waits for the next command.

Recovery Retry the command with fewer object names specified. Up to three object names are allowed for a given object type.

00002

`KRN 000`*`02`*` Negative Subsys response. OBJNAME` *`objname`* `File error` *`err-num`*

`objname`

is the new name of a generic process, in the form $ZZKRN.#*gpname*.

err-num
is a file-system error number.

Cause The command has been rejected by the Kernel subsystem handling facility.

Effect SCF waits for the next command.

Recovery For an explanation of the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command.

00003

```
KRN 00003 Empty response.
```

Cause You entered a command that has no error or warning message response.

Effect SCF waits for the next command.

Recovery Informational message only; no corrective action is needed.

00004

KRN 00004 Token conflict. OBJNAME *objname* TOKEN *num*

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

num
is an SPI token number.

Cause The token specified conflicts with tokens in the requester buffer.

Effect SCF waits for the next command.

Recovery Correct the command and retry.

00005

KRN 00005 Object type and name mismatched. OBJNAME *objname* OBJTYPE *objtype*

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

objtype
is an SCF object type.

Cause The object name does not match the given or assumed object type in this command.

Effect SCF waits for the next command.

Recovery Retry the command with the correct object type and object name.

00006

KRN 00006 Required attribute is missing: *attribute* for *objname*.

attribute
is an object attribute.

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause The attribute listed is required but was not specified.

Effect SCF waits for the next command.

Recovery Retry the command, specifying the required attribute.

00007

```
KRN 00007 INTERNAL ERROR: Case value out of range.
```

Cause An invalid case value was generated with no associated case label.

Effect SCF waits for the next command.

Recovery As described under "If You Have to Call Your Service Provider" (page 145), save pertinent information and contact your service provider.

00008

```
KRN 00008 Duplicate Attribute attribute.
```

attribute
is an object attribute.

Cause You entered the same attribute twice in one command.

Effect SCF waits for the next command.

Recovery Remove the duplicate attribute and retry the command.

00009

```
KRN 00009 Negative Process response. File error err-num.
```

err-num
is a file-system error number.

Cause Because of file-system error *err-num*, the command is rejected.

Effect SCF waits for the next command.

Recovery For recovery from the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command.

00010

```
KRN 00010 Both PrimaryCPU and CPU specified for objname.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause You specified both PRIMARYCPU and CPU in the same command. These attributes are mutually exclusive.

Effect SCF waits for the next command.

Recovery Retry the command, specifying either PRIMARYCPU or CPU.

00011

```
KRN 00011 Invalid Attribute or Attribute value:  attribute for objname.
```

attribute
is an object attribute.

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause Either the value entered is not valid for the attribute, or the attribute is not valid for the object name.

Effect SCF waits for the next command.

Recovery Retry the command with the correct attribute and value.

00012

```
KRN 00012 Wrong filecode for attribute for objname.
```

attribute
is an object attribute.

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause The file specified has the wrong file code. Program and library files should be file code 100 or 700, but both must be the same file code. Resident and nonresident template files should be file code 844 or file code 839.

Effect SCF waits for the next command.

Recovery Retry the command specifying a file name with the proper file code.

00013

`KRN 00013 Cannot START a DISABLED Generic Process: objname.`

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause You attempted to start a generic process that has its STARTMODE value set to DISABLED.

Effect SCF waits for the next command.

Recovery If you want to start this generic process, enter an ALTER command to change the STARTMODE value to something other than DISABLED.

00014

`KRN 00014 Non-existent CPU cpunum specified: for attribute for objname.`

cpunum
is a processor number.

attribute
is an object attribute.

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause Processor *cpunum* does not exist on this system.

Effect SCF waits for the next command.

Recovery Retry the command, specifying an existing processor number.

00015

`KRN 00015 No PRIMARYCPU specified for objname.`

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause You entered an ADD or ALTER command with a BACKUPCPU specified but no PRIMARYCPU specified.

Effect SCF waits for the next command.

Recovery Retry the command, specifying both a PRIMARYCPU and a BACKUPCPU.

00016

`KRN 00016 No CPU or PRIMARYCPU specified for objname.`

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause You entered an ADD or ALTER command without specifying either a PRIMARYCPU or CPU attribute.

Effect SCF waits for the next command.

Recovery Retry the command, including either the PRIMARYCPU or CPU attribute.

00017

`KRN 00017 CPU specified more than once: cpunum for attribute for objname.`

cpunum
is a processor number.

attribute
is the CPU attribute.

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause You entered an ADD or ALTER command that specified processor *cpunum* more than once in the CPU entry.

Effect SCF waits for the next command.

Recovery Retry the command, without repeating any processor numbers.

00018

KRN 00018 Same CPU specified for PRIMARYCPU and for BACKUPCPU for *objname*.

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause You entered an ADD or ALTER command that specified the same processor number for PRIMARYCPU and BACKUPCPU.

Effect SCF waits for the next command.

Recovery Retry the command, using different processor numbers for the primary and backup processors.

00019

KRN 00019 Process name-prefix too long for group of processes for *objname*.

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause You entered an ADD or ALTER command that specified more than one processor for the CPU attribute, but the process name was longer than $followed by three characters.

Effect SCF waits for the next command.

Recovery Retry the command, with a name no longer than $ plus three characters if more than one process is to be started.

00020

KRN 00020 Template update failed: File system error *err-num* on Attribute *attribute*.

err-num
is a file-system error number.

attribute
is an object attribute.

Cause Because of file-system error *err-num* for *attribute*, the EMS templates were not updated.

Effect SCF waits for the next command.

Recovery For recovery from the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command.

00021

KRN 00021 Template update failed: Bad EMS template file for *attribute* for *objname*.

attribute
is an object attribute.

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause The EMS template file specified is not a valid EMS template file. Resident and nonresident template files have the characteristics listed in this table.

Characteristic	Resident Template File	Nonresident Template File
File code	839 or 844	839 or 844
File type	Unstructured	Key-sequenced
Minimum size	4096 bytes	1 byte
Maximum size	65512 bytes	Not defined

Effect SCF waits for the next command.

Recovery Retry the command with a proper EMS template file.

00022

```
KRN 00022 Template update failed: Error err-num in allocating EMS template segment.
```

err-num
is a file-system error number.

Cause The system was unable to allocate the EMS template segment needed, due to the file-system error returned by the call to GETSEGMENT.

Effect SCF waits for the next command.

Recovery Retry the command later. If the needed segment has become available, the command will succeed.

00023

```
KRN 00023 Template update failed: Resident template file > 64k for attribute.
```

attribute
is an object attribute.

Cause The file specified for RESIDENT_TEMPLATES was larger than 64K words.

Effect SCF waits for the next command.

Recovery Retry the command with a template file smaller than 64K.

00024

```
KRN 00024 Template update failed: error err-num in setting up EMS templates.
```

err-num
is a file-system error number.

Cause Because of file-system error *err-num*, the Monitor was unable to set up the EMS templates.

Effect SCF waits for the next command.

Recovery Retry the command, if the source of the problem is understood and can be resolved. If the problem persists, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00025

```
KRN 00025 Template update failed: EMS replacement error err-num occurred.
```

err-num
is a file-system error number.

Cause A file-system error occurred while attempting to replace the EMS templates.

Effect SCF waits for the next command.

Recovery Retry the command, if the problem is understood and can be resolved. If the problem persists, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00026

```
KRN 00026 Process $name failed to start; A process create error err-num, detail det-err occurred.
```

$*name*

is the name of a process.

err-num

is a process creation error number.

det-err

is an error detail number.

Cause A process creation error occurred while attempting to start the process.

Effect SCF waits for the next command.

Recovery Retry the command. For an explanation of the process creation error and the related error detail number, see the *Guardian Procedure Errors and Messages Manual*. If the problem persists, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00027

```
KRN 00027 Error occurred in deleting old reserved name after
successful ALTER: Generic Process objname.
Error 1, Detail err-det.
```

objname

is the new name of a generic process, in the form $ZZKRN.#*gpname*.

err-det

is a file-system error number.

Cause The ALTER command succeeded, resulting in a change in the process name, but an unexpected error occurred while trying to delete a record for the old name.

Effect The name or names remain reserved. SCF waits for the next command.

Recovery Check for an EMS event that was put out regarding this. This EMS event should indicate the name or names that could not be unreserved in the destination control table. Save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00028

```
KRN 00028 Process $name aborted successfully.
```

$*name*

is the name of a process.

Cause The process stopped successfully in response to the ABORT command.

Effect SCF waits for the next command.

Recovery Informational message only; no corrective action is needed.

00029

```
KRN 00029 Generic Process objname not altered.
Only STARTMODE can be altered because the Generic Process
is in ABORTING state.
```

objname

is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause An ALTER command failed because one of more of the processes controlled by this generic process is in the ABORTING state. A generic process must be in the STOPPED state in order to alter other process attributes.

Effect SCF waits for the next command.

Recovery Either (1) wait until the generic process is in the STOPPED state before using the ALTER command to change attributes or (2) use the ALTER command to change the start mode.

00030

```
KRN 00030 Process $name started successfully.
```

$*name*

is the name of a process.

Cause The process started successfully in response to the START command.

Effect SCF waits for the next command.

Recovery Informational message only; no corrective action is needed.

00031

```
KRN 00031 Process $name has refused the abort request,
but will be considered ABORTING.
```

$*name*

is the name of a process.

Cause The process receiving the abort request cannot stop because it is in an invalid state.

Effect Even though the process may still be running, the $ZPM persistence manager reports the process state as ABORTING, so the process does not start after a future processor reload. SCF waits for the next command.

Recovery Retry the ABORT command. If the problem persists and the process was configured with a PROGRAM file supplied by HP, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00032

```
KRN 00032 Process $name did not stop because of an
error err-num, but will be considered ABORTING.
```

$*name*

is the name of a process.

err-num

is a file-system error number.

Cause Because of the specified file-system error, the process has not stopped.

Effect Even though the process may still be running, the $ZPM persistence manager reports the process state as ABORTING, so the process does not start after a future processor reload. SCF waits for the next command.

Recovery Retry the ABORT command. If the problem persists and the process was configured with a program file supplied by HP, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00033

```
KRN 00033 The abort request to process $name has timed out,
but will be considered ABORTING.
```

$*name*

is the name of a process.

Cause The abort request was not acknowledged by $*name*, so the process did not stop.

Effect Even though the process may still be running, the $ZPM persistence manager reports the process state as ABORTING, so the process does not start after a future processor reload. SCF waits for the next command.

Recovery Retry the ABORT command. If the problem persists, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00034

```
KRN 00034 The abort request for process $name has been queued.
Until the process stops, it will be considered ABORTING.
```

$*name*

is the name of a process.

Cause The abort request was sent to $*name*, but its state is currently unstoppable. The $ZPM persistence manager considers the process to be in the ABORTING state; if it eventually stops, it is then considered ABORTED.

Effect Even though the process may still be running, the $ZPM persistence manager reports the process state as ABORTING. As a result, the process does not start after a future processor reload. SCF waits for the next command.

Recovery Informational message only; no corrective action is needed.

00035

```
KRN 00035 Process $name did not finish processing the abort request,
but the process will be considered ABORTING.
```

$*name*

is the name of a process.

Cause The process has not entered the ABORTED state in response to an abort request. The $ZPM persistence manager considers the process to be in the ABORTING state; if it eventually stops, it is then considered ABORTED.

Effect SCF waits for the next command.

Recovery Retry the ABORT command. Even though the process may still be running, the $ZPM persistence manager reports the process state as ABORTING. As a result, the process does not start after a future processor reload.

00036

```
KRN 00036 Generic Process objname is a subsystem manager
and cannot be aborted by a wildcard ABORT.
```

objname

is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause This generic process is a subsystem manager. Subsystem managers cannot be stopped by an ABORT command using a wild card (*).

Effect SCF waits for the next command.

Recovery To stop a subsystem manager, you must specify its complete name in an ABORT command.

00037

```
KRN 00037 Other processes of objname considered ABORTED;
CPU currently down.
```

objname

is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause One or more of the processes represented by this generic process were configured in processors that are down or nonexistent. These processes are now considered ABORTED and will not be started when their processor comes up.

Effect SCF waits for the next command.

Recovery Informational message only; no corrective action is needed.

00038

```
KRN 00038 The USERID specified for attribute USERID for objname
is not known to this system.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause The value given for USERID is not a known user ID on this system.

Effect SCF waits for the next command.

Recovery Be sure that this is the correct entry that you want, because this will not be checked other than through an attempted START. If this is not the USERID value you intend to use, use the ALTER command to correct the value.

00039

```
KRN 00039 Generic Process objname is a Subsystem Manager,
and is not deleted.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause This generic process is a subsystem manager. Subsystem managers cannot be removed by a DELETE command using a wild card (*).

Effect SCF waits for the next command.

Recovery To remove a subsystem manager, you must specify its complete name in a DELETE command.

00040

```
KRN 00040 Generic Process objname not deleted:
Not all its processes are stopped.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause Not all the processes represented by the specified generic process are in the STOPPED or ABORTED state, so it cannot be deleted.

Effect SCF waits for the next command.

Recovery If you wish to delete this generic process, you must first enter an ABORT command on it.

00041

```
KRN 00041 Generic Process objname not deleted:
Error 1, Detail err-det deleting reserved names.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

err-det
is a file-system error number.

Cause $ZZKRN could not delete the definition of the specified generic process from the system configuration database because one or more instances of the process (which was configured in multiple processors) has not finished stopping.

Effect Some but not all of the old names may have been deleted. Do not try to START this process because not all the instances of it remain. SCF waits for the next command.

Recovery Wait and retry the DELETE command.

00042

```
KRN 00042 Generic Process objname not found by $ZPM;
it is considered ABORTED.
```

objname
is the name of a generic process, in the form $ZZKRN.#*gpname*.

Cause In response to an ABORT command, the $ZPM persistence manager cannot find the specified generic process.

Effect $ZPM reports the process state as ABORTED. As a result, the process does not start after a future processor reload. SCF waits for the next command.

Recovery Informational message only; no corrective action is needed.

00043

```
KRN 00043 File-system error err-num returned from $ZPM
when trying to { ABORT | START | STATUS } objname.
```

err-num
is a file-system error number.

objname
is either the new name of a generic process, in the form $ZZKRN.#*gpname* (if the object type is PROCESS) or the $ZZKRN subsystem manager (if the object type is SUBSYSTEM).

Cause The $ZPM persistence manager returned the specified file-system error when trying to abort, start, or display the status of the specified object.

Effect SCF waits for the next command.

Recovery For an explanation of the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command.

00044

```
KRN 00044 Process $name failed to start;
FS error err-num on program file.
```

$*name*
is the name of a process.

err-num
is a file-system error number.

Cause The specified file-system error occurred while accessing the program file.

Effect SCF waits for the next command.

Recovery For an explanation of the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command.

00045

```
KRN 00045 Process $name failed to start;
FS error err-num on library file.
```

$*name*
is the name of a process.

err-num
is a file-system error number.

Cause The specified file-system error occurred while accessing the library file.

Effect SCF waits for the next command.

Recovery For an explanation of the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command.

00046

```
KRN 00046 Process $name failed to start;
FS error err-num on swap file.
```

$*name*
is the name of a process.

err-num
is a file-system error number.

Cause The specified file-system error occurred while creating or opening the swap file.

Effect SCF waits for the next command.

Recovery For an explanation of the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command.

00047

```
KRN 00047 Process $name failed to start;
FS error err-num on extended swap file.
```

$*name*
is the name of a process.

err-num
is a file-system error number.

Cause The specified file-system error occurred while creating or opening the extended swap file.

Effect SCF waits for the next command.

Recovery For an explanation of the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command.

00048

```
KRN 00048 Process $name failed to start;
FS error err-num on extended data segment initializiation.
```

$*name*
is the name of a process.

err-num
is a file-system error number.

Cause Because of file-system error *err-num*, the operating system was unable to set up an extended data segment.

Effect SCF waits for the next command.

Recovery For an explanation of the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command.

00049

```
KRN 00049 Process $name failed to start;
FS error err-num on home term.
```

$*name*
is the name of a process.

err-num
is a file-system error number.

Cause Because of file-system error *err-num*, the specified home terminal does not exist or was not a legal process or terminal name.

Effect SCF waits for the next command.

Recovery For an explanation of the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command.

00050

```
KRN 00050 Process $name failed to start;
FS error err-num occurred while attempting to
communicate with the monitor process.
```

$*name*
is the name of a process.

err-num
is a file-system error number.

Cause Because of file-system error *err-num*, the $ZPM persistence manager is unable to communicate with the system monitor, possibly because the processor module where the process was to be run does not exist or is inoperable.

Effect SCF waits for the next command.

Recovery For an explanation of the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command.

00051

```
KRN 00051 Process $name failed to start;
FS error err-num on process name.
```

$*name*
is the name of a process.

err-num
is a file-system error number.

Cause Because of file-system error *err-num*, the process name was invalid.

Effect SCF waits for the next command.

Recovery For an explanation of the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command.

00052

```
KRN 00052 Process $name has undefined externals.
```

$*name*
is the name of a process.

Cause The process has external references that could not be resolved.

Effect SCF waits for the next command.

Recovery Determine the reason for the undefined externals and remedy it. If the problem persists, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

If the process started in spite of the undefined externals, it may be corrupted. Abort the process, fix the problem, and then start the process again.

00053

```
KRN 00053 Process $name failed to start;
No available Process Control Block.
```

$*name*
is the name of a process.

Cause All entries in the process control block table for the processor are in use. This is usually the result if all of the low PIN values (less than 256) are being used, and a low PIN is requested.

Effect SCF waits for the next command.

Recovery Either (1) start the process in a different processor, (2) specify HIGHPIN ON for the process (if HIGHPIN OFF was specified), or (3) wait and retry the operation.

00054

```
KRN 00054 Process $name failed to start;
Unable to allocate map.
```

$*name*

is the name of a process.

Cause Not enough space was available in the processor's MAPPOOL to permit the system monitor to generate the code and data-map copies required by the new process.

Effect SCF waits for the next command.

Recovery Start the process in a different processor, or wait and retry the operation.

00055

```
KRN 00055 Process $name failed to start;
Unlicensed privileged program.
```

$*name*

is the name of a process.

Cause The program file contains procedures having CALLABLE or PRIV attributes, but the program is not licensed to execute in privileged mode and is not being run by the super ID (255, 255).

Effect SCF waits for the next command.

Recovery Have the super ID license the program, as described in the *File Utility Program (FUP) Reference Manual*.

00056

```
KRN 00056 Process $name failed to start; Library conflict.
```

$*name*

is the name of a process.

Cause The operation specifies a library file, but either (1) the program is already running with another library or no library or (2) the library and program file are of different file codes.

Effect SCF waits for the next command.

Recovery Perform the operation either (1) without specifying a library file or (2) with library and program files of the same file code.

00057

```
KRN 00057 Process $name failed to start;
Program and Library files cannot be the same file.
```

$*name*

is the name of a process.

Cause You specified the same file for the program file and library file.

Effect SCF waits for the next command.

Recovery Retry the command, specifying different files for the library and program files.

00058

```
KRN 00058 Process $name failed to start;
Process SUBTYPE is bad.
```

$*name*

is the name of a process.

Cause You attempted to create a process with an improper SUBTYPE.

Effect SCF waits for the next command.

Recovery Correct the cause of the error and retry the command.

00059

```
KRN 00059 The value given for { SYSTEM_NAME | SYSTEM_NUMBER } for $ZZKRN
is known to already exist in the network.
```

Cause You entered an ALTER SUBSYS command for system name or system number, but the specified value for the attribute is known to already exist in the network.

Effect SCF issues this warning, makes the change, and waits for the next command.

Recovery If you did not mean to specify this attribute value, reissue the ALTER command with the value you want to use.

00060

```
KRN 00060 Process $name failed to start; Not SUPER group.
```

$*name*

is the name of a process.

Cause The operation requires the user to be a super-group user (255, *n*).

Effect SCF waits for the next command.

Recovery Log on as a super-group user and retry the operation.

00061

```
KRN 00061 Process $name failed to start;
Process name in use.
```

$*name*

is the name of a process.

Cause The process name is currently in use by another process.

Effect SCF waits for the next command.

Recovery If this is not a generic process that has some processes in its group started, use the ALTER command to change the name, and then start it again. If this is a generic process that has some processes in its group started, use the ABORT command to stop the processes before using the ALTER command.

00062

```
KRN 00062 Process $name failed to start;
unexpected BADIOP, detail err-num occurred.
```

$*name*

is the name of a process.

err-num

is an error number.

Cause The process failed to start due to a process creation BADIOP error.

Effect SCF waits for the next command.

Recovery The wrong program file may be in use. Retry the command with the correct program file. If the problem continues, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00063

```
KRN 00063 Process $name failed to start;
Invalid PFS size in object file.
```

$*name*

is the name of a process.

Cause The process object file has a process file segment (PFS) size that is either less than 128K or greater than 1024K.

Effect SCF waits for the next command.

Recovery If you changed the PFS size, set it back to its original value and retry the command. If you did not change it and the error occurs, either set the size to a legal value and retry the command or, if this is a file supplied by HP, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145)

00064

```
KRN 00064 Process $name failed to start;
Unable to allocate a PRIV stack.
```

$*name*
is the name of a process.

Cause A PRIV stack could not be allocated for the process started.

Effect SCF waits for the next command.

Recovery Retry the command. If the failure continues, start the process in a different processor.

00065

```
KRN 00065 Process $name failed to start;
Unable to lock a PRIV stack.
```

$*name*
is the name of a process.

Cause A PRIV stack could not be locked for the process started.

Effect SCF waits for the next command.

Recovery Retry the command. If the failure continues, start the process in a different processor.

00066

```
KRN 00066 Process $name failed to start;
Unable to allocate a MAIN stack.
```

$*name*
is the name of a process.

Cause A MAIN stack could not be allocated for the process started.

Effect SCF waits for the next command.

Recovery Retry the command. If the failure continues, start the process in a different processor.

00067

```
KRN 00067 Process $name failed to start;
Unable to lock a MAIN stack.
```

$*name*
is the name of a process.

Cause A MAIN stack could not be locked for the process started.

Effect SCF waits for the next command.

Recovery Retry the command. If the failure continues, start the process in a different processor.

00068

```
KRN 00068 Process $name failed to start;
FS error err-num occurred during an attempt
to obtain or propagate security indentity information.
```

$*name*
is the name of a process.

err-num
is a file-system error number.

Cause A file-system error occurred while setting up the security context for the process.

Effect SCF waits for the next command.

Recovery For information on this error, see the *Guardian Procedure Errors and Messages Manual*.

00069

```
KRN 00069 Process $name failed to start;
Illegal OSS process creation request.
```

$*name*

is the name of a process.

Cause The process you attempted to start is an Open Systems Services (OSS) process, not a generic process.

Effect SCF waits for the next command.

Recovery SCF does not support starting an OSS process.

00070

```
KRN 00070 Process $name failed to start;
{ Program | Library } file is not a disk file.
```

$*name*

is the name of a process.

Cause The program or library file for the specified process is not a disk file.

Effect SCF waits for the next command.

Recovery Retry the command, specifying a disk file.

00071

```
KRN 00071 Process $name failed to start;
{ Program | Library } file is invalid object file.
```

$*name*

is the name of a process.

Cause The program or library file for the specified process is an invalid object file.

Effect SCF waits for the next command.

Recovery Retry the command, specifying a valid object file.

00072

```
KRN 00072 Process $name failed to start;
{ Program | Library } file has bad file structure.
```

$*name*

is the name of a process.

Cause The program or library file for the specified process has a bad file structure.

Effect SCF waits for the next command.

Recovery Retry the command, specifying a correct file.

00073

```
KRN 00073 Process $name failed to start;
{ Program | Library } file requires later operating system version.
```

$*name*

is the name of a process.

Cause The program or library file for the specified process requires a later version of the operating system.

Effect SCF waits for the next command.

Recovery Install the required version of the operating system, or use a different version of this process.

00074

```
KRN 00074 Process $name failed to start;
Program has no main procedure.
```

$*name*
is the name of a process.

Cause The program file does not contain a main procedure.

Effect SCF waits for the next command.

Recovery Retry the command, specifying a program file that contains a main procedure.

00075

```
KRN 00075 Process $name failed to start;
Library contains a main procedure.
```

$*name*
is the name of a process.

Cause The library file contains a main procedure.

Effect SCF waits for the next command.

Recovery Retry the command, specifying a library file that does not contain a main procedure.

00076

```
KRN 00076 Process $name failed to start;
Program has no data pages.
```

$*name*
is the name of a process.

Cause The program file does not contain any data pages.

Effect SCF waits for the next command.

Recovery Retry the command, specifying a program file that contains data pages.

00077

```
KRN 00077 Process $name failed to start;
{ Program | Library } file has invalid PEP.
```

$*name*
is the name of a process.

Cause The program or library file for the specified process contains an invalid procedure entry point (PEP) table.

Effect SCF waits for the next command.

Recovery Retry the command, specifying a file that contains a valid PEP table.

00078

```
KRN 00078 Process $name failed to start;
{ Program | Library } file header fields not consistent with size.
```

$*name*
is the name of a process.

Cause The program or library file header INITSEGS for the specified process is not consistent with its size.

Effect SCF waits for the next command.

Recovery Retry the command, specifying a correct file.

00079

```
KRN 00079 Process $name failed to start;
{ Program | Library } file resident size greater than code area length.
```

$*name*

is the name of a process.

Cause The program or library file resident size for the specified process is greater than the code area length.

Effect SCF waits for the next command.

Recovery Retry the command, specifying a correct file.

00080

```
KRN 00080 Process $name failed to start;
{ Program | Library } file not fixed up by binder.
```

$*name*

is the name of a process.

Cause The program or library file for the specified process has not been processed by the Binder program.

Effect SCF waits for the next command.

Recovery Use Binder to fix the file. Then retry the command.

00081

```
KRN 00081 Process $name failed to start;
{ Program | Library } file has undefined data blocks.
```

$*name*

is the name of a process.

Cause The program or library file for the specified process has undefined data blocks.

Effect SCF waits for the next command.

Recovery Correct the error that caused this condition, and retry the command. If you cannot do this, and this is a file supplied by HP, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00082

```
KRN 00082 Process $name failed to start;
{ Program | Library } file has unresolved code block references in data blocks.
```

$*name*

is the name of a process.

Cause The program or library file for the specified process has unresolved code block references in data blocks.

Effect SCF waits for the next command.

Recovery Correct the error that caused this condition, and retry the command. If this is a file supplied by HP, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00083

```
KRN 00083 Process $name failed to start;
{ Program | Library } file contains too many code segments.
```

$*name*

is the name of a process.

Cause The program or library file for the specified process contains too many code segments for the process to be able to start.

Effect SCF waits for the next command.

Recovery Reconfigure the process with an object file that has fewer code segments.

00084

```
KRN 00084 Process $name failed to start;
{ Program | Library } file has unrecognized error err-num.
```

$name
is the name of a process.

err-num
is an error number.

Cause An unrecognized error occurred on the specified program or library file.

Effect SCF waits for the next command.

Recovery Retry the command, using a different version of the file that resulted in the error.

00085

```
KRN 00085 Process $name failed to start;
Process create error UNKNOWN, detail err-num occurred.
```

$name
is the name of a process.

err-num
is a process creation error detail number.

Cause A process create UNKNOWN error, with the specified error detail, occurred while trying to start the process.

Effect SCF waits for the next command.

Recovery Retry the command. For an explanation of the related error detail number, see the *Guardian Procedure Errors and Messages Manual*. If the failure continues, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00086

```
KRN 00086 System Configuration Database is locked;
Cannot access it for objname.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause The $SYSTEM.ZSYSCONF.CONFIG system configuration database is locked, so it cannot be accessed.

Effect SCF waits for the next command.

Recovery Retry the command. If the failure continues, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00087

```
KRN 00087 System Configuration Database access for objname failed:
Error err-num1, Detail err-num2.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

err-num1
is an error number.

err-num2
is another error number.

Cause The $SYSTEM.ZSYSCONF.CONFIG system configuration database access routines returned the stated error and error detail, and the access failed.

Effect SCF waits for the next command.

Recovery Retry the command. If the failure continues, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00088

```
KRN 00088 A record with the given name already exists in the System Configuration Database:
ADD failed for objname.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause An attempt was made to add a record to the $SYSTEM.ZSYSCONF.CONFIG system configuration database, but a generic process of the same *objname* already exists for this subsystem.

Effect SCF waits for the next command.

Recovery Retry the ADD command, using a different object name.

00089

```
KRN 00089 A process name required for objname is already reserved.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause The ADD or ALTER command failed because a process name was previously reserved.

Effect SCF waits for the next command.

Recovery Retry the ADD or ALTER command, using a different NAME value.

00090

```
KRN 00090 Illegal attempt to set the USERID for objname.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause Only the local super ID (255,255) can give the USERID attribute a value other than the user's own user ID.

Effect SCF waits for the next command.

Recovery Retry the command, but do not attempt to set a USERID value.

00091

```
KRN 00091 Name reservation error for objname:
error 1, error detail err-det.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

err-det
is a file-system error number:

13	FEBADNAME	Process had a bad name
6026	FETYPEMISMATCH	Process type is not 0 (for a generic process)

Cause An ADD or ALTER command failed because an unexpected error occurred while attempting to reserve a name for the generic process.

Effect SCF waits for the next command.

Recovery Attempt to determine the cause of the errors, or retry the operation using a different process name.

00092

```
KRN 00092 Error occurred in deleting a reserved name entry after ADD or ALTER failed:
Generic Process objname. Error 1, Detail err-det.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

err-det
is a file-system error number.

Cause An ADD or ALTER command failed, resulting in the need to unreserve names that had been reserved, but an unexpected error occurred while unreserving.

Effect The names meant to be unreserved remain reserved. SCF waits for the next command.

Recovery Check for an EMS event that was put out regarding this. This EMS event should indicate which names were reserved and could not be unreserved. Save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00093

```
KRN 00093 err-type error err-num occurred
trying to update SP with new System NAME, NUMBER, DST or Time Zone Offset.
```

err-type
is one of these values:

Error Type	Effect
SP 1 Cruhandle Get	No update to service processor 1.
SP 0 FirFileWrite	No update to service processor 0.
SP 1 FirFileWrite	No update to service processor 1.

err-num
is a service processor (SP) error number.

Cause An error has occurred in attempting to update the values for SYSTEM_NAME, SYSTEM_NUMBER, DAYLIGHT_SAVING_TIME, or TIME_ZONE_OFFSET.

Effect See the preceding table. If you get an error on both SP 0 and SP 1, the system configuration database is not updated. SCF waits for the next command.

Recovery Save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00094

```
KRN 00094 ALTER SUBSYS failed: No changes were made.
```

Cause The ALTER SUBSYS command has failed due to other errors listed with this message.

Effect No changes are made to the system or to the system configuration database. SCF waits for the next command.

Recovery Retry the operation after resolving the errors returned. If the problem persists, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00095

```
KRN 00095 Powerfail delay time update failed.
```

Cause An error occurred while changing the power failure time interval.

Effect The POWERFAIL_DELAY_TIME value is not changed. SCF waits for the next command.

Recovery Retry the command. If the problem persists, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00096

```
KRN 00096 Attribute attribute for objname
is not on this system.
```

attribute
is an object attribute.

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause The specified attribute is not on this system. This message can be either an ERROR or a WARNING. If this is an ERROR, it is because it is a requirement that the attribute be on this system.

Effect SCF waits for the next command.

Recovery If this is an ERROR, specify an attribute that is on this system, and retry the operation. If this is a WARNING, be sure that this is the correct entry that you want for this attribute (considering that it is not on this system), because this will not be checked again other than through an attempted START. If this is not the entry you intend to use, use the ALTER command to change to the appropriate entry.

00097

```
KRN 00097 The filename specified for attribute for
objname was not located.
```

attribute
is an object attribute.

objname
is the name of a new generic process, in the form $ZZKRN.#*gpname*.

Cause The file does not exist. This message may be either an ERROR or a WARNING. If this is an ERROR, it is because it is a requirement that the file exist at this time.

Effect SCF configures the process and waits for the next command.

Recovery Be sure that this is the correct file location you want, and be sure to put the correct file there, because the file name will not be checked other than through an attempted START. If this is not the file you intend for use, use the ALTER command to change to the appropriate file.

00098

```
KRN 00098 The volume specified for attribute for
objname was not located.
```

attribute
is an object attribute.

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause The volume does not currently exist.

Effect SCF waits for the next command.

Recovery Be sure that this is the correct volume location you want, because this will not be checked other than through an attempted START. If this is not the volume you intend to use, use the ALTER command to change to the appropriate volume.

00099

```
KRN 00099 The system named for the file for attribute for
objname is not known.
```

attribute
is an object attribute.

objname
is the name of a newly configured generic process, in the form $ZZKRN.#*gpname*.

Cause The system name for the file or device is not known to this system.

Effect SCF configures the process and waits for the next command.

Recovery Be sure that this is the correct entry that you want, because this will not be checked other than through an attempted START. If this is not the system you intend to use, use the ALTER command to change to the appropriate system.

00100

```
KRN 00100 The path to the system for attribute for
objname is currently down.
```

attribute
is an object attribute.

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause The path to the system that the specified file or device is on is currently down.

Effect SCF waits for the next command.

Recovery Be sure that this is the correct entry that you want, because this will not be checked other than through an attempted START. If this is not the entry you intend to use, use the ALTER command to change to the appropriate entry.

00101

```
KRN 00101 The name specified for attribute HOMETERM for
objname was not recognized.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause The name is for a device that is not currently on the system, or was not correctly specified.

Effect SCF waits for the next command.

Recovery Be sure that this is the correct entry that you want, because this will not be checked other than through an attempted START. If this is not the HOMETERM value you intend to use, use the ALTER command to change to the appropriate home terminal.

00102

```
KRN 00102 Generic Process objname not altered:
Not all its processes are stopped.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause The ALTER command failed because not all the processes controlled by this generic process are in the STOPPED state.

Effect SCF waits for the next command.

Recovery Use the ABORT command on a generic process before using the ALTER command. A generic process must be in the STOPPED state in order for the ALTER command to work.

00103

```
KRN 00103 The device specified for attribute for
objname was not located.
```

attribute
is an object attribute.

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause The device does not currently exist.

Effect SCF waits for the next command.

Recovery Be sure that this is the correct entry that you want, because this will not be checked other than through an attempted START. If this is not the entry you intend to use, use the ALTER command after starting the process to change to the appropriate entry.

00104

```
KRN 00104 The KRN subsystem manager, Generic Process $ZZKRN, cannot be aborted.
```

Cause You attempted to abort the $ZZKRN subsystem manager, but the $ZZKRN subsystem manager cannot be aborted.

Effect SCF waits for the next command.

Recovery Informational message only; no corrective action is needed.

00105

```
KRN 00105 Generic Process objname cannot be altered or deleted:
inconsistent status response from $ZPM.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause The $ZPM persistence manager returned an inconsistent status response to an internal check, which was done to verify that all instances of this generic process are stopped. The status indicated the STOPPED state, but not all PIDs were null (displayed as "None" in a STATUS PROCESS command).

Effect SCF waits for the next command.

Recovery One or more instances of the generic process may still be running. Issue an ABORT command on *objname* and then check the status. If an inconsistent status (STOPPED state, but PIDs not "None") is displayed, contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00106

```
KRN 00106 Other processes of objname did not start due to cpu down.
They will start when their CPU is reloaded.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause One or more of the processes controlled by this generic process were configured in processors that are down or nonexistent. These processes will be started by the $ZPM persistence manager when their processor is reloaded.

Effect SCF waits for the next command.

Recovery Informational message only; no corrective action is needed.

00107

```
KRN 00107 Invalid values given for the SERVERNET object.
```

Cause The form of the SERVERNET object name is not valid. For the START or STOP command, the SERVERNET object name must be specified as one of these:

$name.X.*n*
$name.Y.*n*
$name.X.*
$name.Y.*

where X or Y represents the ServerNet fabric, *n* represents a processor number, and the asterisk (*) represents all processors. For the STATUS or NAMES command, only $*name* is required.

Effect SCF waits for the next command.

Recovery Retry the command with the correct form of the SERVERNET object name.

00108

```
KRN 00108 SP Session Destroy failed with error err-num.
```

err-num
is a service processor error number.

Cause When requested to close the completed session, the service processor (SP) returns an error.

Effect SCF waits for the next command.

Recovery Save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00109

```
KRN 00109 The system SHUTDOWN is unsuccessful.
The SP returned { Session Create | Power Down } error err-num.
```

err-num
is a service processor (SP) error number.

Cause When requested to power down the system, the service processor (SP) returns error *err-num* and does not shut down the system.

Effect SCF waits for the next command.

Recovery Save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00111

```
KRN 00111 Only the Kernel Manager name can be given for this object.
```

Cause The name given for the object with this command must be "$ZZKRN" and nothing else. (Wild cards are not supported.)

Effect SCF waits for the next command.

Recovery Retry the operation giving only the $ZZKRN subsystem manager name.

00112

```
KRN 00112 File system error err-num occurred trying to open $ZPM.
```

err-num
is a file-system error number.

Cause The specified error occurred while $ZZKRN was trying to open the $ZPM persistence manager process.

Effect SCF waits for the next command.

Recovery For an explanation of the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command. If the problem continues, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00113

```
KRN 00113 File system error err-num occurred on WRITEREAD with $ZPM.
```

err-num
is a file-system error number.

Cause The specified file system error occurred during the WRITEREAD operation used for communication between $ZZKRN and the $ZPM persistence manager.

Effect Note that if the error is file-system error 40 (The operation timed out) and this occurred during an ABORT or a START command, the command may have completed successfully. Use the STATUS command to verify this. SCF waits for the next command.

Recovery For an explanation of the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command. If the problem continues, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00114

```
KRN 00114 Attribute attribute for
objname is not on the system-load volume.
```

attribute
is an object attribute.

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause The specified attribute is not on the system-load volume. This message may be either an ERROR or a WARNING. If this is an ERROR, it is because it is a requirement that the attribute be on the system-load volume.

Effect SCF waits for the next command.

Recovery If this is an ERROR, specify an attribute value that is on the system-load volume and retry the operation. If this is a WARNING, be sure that this is the correct entry that you want for this attribute (considering that it is not on the system-load volume), because this will not be checked again other than through an attempted START. If this is not the entry you intend to use, use the ALTER command to change to the appropriate entry.

00115

```
KRN 00115 File system error err-num occurred in
attempting SERVERNET operation.
```

err-num
is a file-system error number.

Cause The specified file-system error was returned by the system library call used to attempt the SERVERNET operation.

Effect SCF waits for the next command.

Recovery For an explanation of the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command. If the problem continues, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00116

```
KRN 00116 err-type error err-num occurred
in trying to read system values from the Service Processor.
```

err-type
is one of these error types:

Error Type	Definition
Cruhandle Get	Unable to get the CRU handle of either service processor.
FirFileRead	Unable to read from either service processor.
Session Create	Unable to create the session with the service processor.
Session Destroy	The service processor session has not been properly terminated.

err-num
is a file-system error number.

Cause An error has occurred during an attempt to read the values for system name, system number, daylight-saving time, and time zone offset from the service processor.

If you get this error in response to an ALTER command, the alter of these attributes is not carried out, unless *err-type* is Session Destroy.

If you get this error in response to an INFO command, the Pending Changes for these values could not be obtained, unless *err-type* is Session Destroy.

Effect SCF waits for the next command.

Recovery For an explanation of the file-system error, see the *Guardian Procedure Errors and Messages Manual*. Then retry the command. If the problem continues, save pertinent information and contact your service provider, as described under "If You Have to Call Your Service Provider" (page 145).

00120

```
KRN 00120 Destination Control Table Resize support not available.
```

Cause The Kernel Subsystem Manager ($ZZKRN) rejected the command because support for Destination Control Table resizing is not present.

Effect The command is not executed. SCF waits for the next command.

Recovery Check to see that you have a version of the subsystem manager that supports Destination Control Table resizing.

0121

```
KRN 00120 Destination Control Table Entries In Use.
```

Cause When the Kernel Subsystem Manager ($ZZKRN) successfully executed the command to resize the Destination Control Table to make it SMALL, some Destination Control Table entries outside the range supported by a SMALL Destination Control Table were in use when the resize command was executed.

Effect SCF waits for the next command.

Recovery This is an informational message only.

00122

```
KRN 00122 ALTER PROCESS failed: No changes were made.
```

Cause The ALTER PROCESS command failed due to the causes described in the error messages accompanying this message.

Effect No changes are made to the system or to the System Configuration Database.

Recovery Retry the operation after resolving the problems that the error messages indicate. If the problem persists, save pertinent information and contact your service provider:

00123

```
KRN 00123 Insufficient Free Destination Control Table Entries would remain.
```

Cause Some DCT entries (below the current DCT entry limit) must remain free for correct system behavior. The DCT resize request you issued would leave insufficient free entries below the requested new limit. Therefore, this DCT resize request has been rejected.

Effect No changes are made to the system. SCF waits for the next command.

Recovery Please retry the operation after there are sufficient free DCT entries below the requested new DCT limit

00124

```
KRN 00124 Invalid combination of attributes was used.
```

Cause You issued a command specifying one or more attributes inconsistent with one another.

Effect SCF does not execute the command and waits for the next command.

Recovery Issue this command to examine the command syntax:

```
HELP KERNEL command object-type
```

command
is a SCF KERNEL command.

object-type
is an SCF object type.

The HELP command gives information about the combination of attributes allowed with the command. Resolve the attribute inconsistency.

00125

```
KRN 00125 Error err-num returned in response by $ZPM.
```

err-num
is the error number.

Cause An invalid command was issued, causing a $ZPM to return the specified error.

Effect No changes are made to the system.

Recovery For a description of *err-num*, use the Conferr tool on the SYSnn. Correct the error and retry the command. If the problem persists, save pertinent information and contact your service provider.

00126

```
KRN 00126 ASSIGN, PARAM, or DEFINE for Generic Process objname not added:
Not all its processes are stopped.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause Not all the processes represented by the specified generic process are in the STOPPED or ABORTED state, so no ASSIGN, PARAM, or DEFINE can be added.

Effect The ASSIGN, PARAM, or DEFINE attributes for the generic process are not added.

Recovery To add an ASSIGN, PARAM, or DEFINE to this generic process, you must first either stop or abort the process.

00127

```
KRN 00127 proc-name returns error err-num
while operating on the DEFINE attribute attr-name
```

proc-name
is the name of the procedure.

err-num
is the error number.

attr-name
is the name of the DEFINE attribute.

Cause An invalid DEFINE attribute name or value was specified and so the *proc-name* encountered an error while operating on the specified DEFINE attribute.

Effect The DEFINE attribute does not get added.

Recovery For information on the specific DEFINE error, see the *Guardian Procedure Errors and Messages Manual*.

00128

```
KRN 00128 proc-name returned error err-num
while operating on the DEFINE.
```

proc-name
is the name of the procedure.

err-num
is the error number.

Cause An invalid DEFINE name, DEFINE attribute name or value was specified and so the *proc-name* encountered an error while operating on the DEFINE.

Effect The DEFINE attribute does not get added.

Recovery For information on the specific DEFINE error, see the *Guardian Procedure Errors and Messages Manual*.

00129

```
KRN 00129 DEFINERESTORE failed for define-name with
error err-num and error detail err-det.
```

define-name
is the name of the DEFINE.

err-num
is the error number.

err-det
is a DEFINE attribute error. For details, see the *Guardian Programmer's Guide*.

Cause An invalid saved DEFINE buffer was being restored to the process space.

Effect The saved DEFINE buffer does not get restored.

Recovery The error value determines what action needs to be taken. If the error indicates a failure on consistency checks, the error detail indicates the checknum value. This checknum value identifies which consistency check failed.

For details, see the *Guardian Procedure Errors and Messages Manual*.

00130

```
KRN 00130 ADD PROCESS failed: No changes were made.
```

Cause An attempt was made to add more than one parameter out of ASSIGN, PARAM, and DEFINE.

Effect The ASSIGN, PARAM, or DEFINE attribute is not added.

Recovery Retry the command, including no more than one of ASSIGN, PARAM, or DEFINE in the command at a time.

00131

```
KRN 00131 Cannot ADD multiple DEFINEs.
```

Cause An attempt was made to add multiple DEFINEs.

Effect The DEFINEs are not added.

Recovery Retry the command, adding only one DEFINE in the command.

00132

```
KRN 00132 TLE_LIMIT attribute support not available.
```

Cause The Kernel Subsystem Manager ($ZZKRN) rejected the command because the underlying T9050 NonStop Kernel version does not support TLE LIMIT attribute.

Effect The command is not executed. SCF waits for the next command.

Recovery Check to see that you have a version of T9050 that supports a limit on per process TLE usage. (A limit on TLE usage can be configured as of the J06.09, H06.20, and G06.32.01 RVUs.)

00133

```
KRN 00133 AUTO_RETRY_ON_ERROR_654 attribute support not available.
```

Cause

The Kernel Subsystem Manager ($ZZKRN) rejected the command because the underlying T9050 NonStop Kernel version does not support the AUTO_RETRY_ON_ERROR_654 attribute.

Effect

The command is not executed. SCF waits for the next command.

Recovery

Check to see that you have a version of T9050 that supports the AUTO_RETRY_ON_ERROR_654 feature. (This feature is available as of the J06.09 and H06.20 RVUs).

00134

```
KRN 00134 One or more attributes are not supported.
```

Cause The Kernel Subsystem Manager ($ZZKRN) rejected the command because the underlying T9050 NonStop Kernel version does not support one or more attributes.

Effect The command is not executed. SCF waits for the next command.

Recovery Check the version compatibility with T9050.

Common Error Messages

-00004

```
KRN E-00004 Command/object type combination is not valid for this subsystem.
```

Cause A command has been issued, and the object contained in the command buffer is not a valid object type for this subsystem.

Effect SCF waits for the next command.

Recovery Correct the object type and retry the command.

-00005

```
KRN E-00005 Command is not supported by this subsystem.
```

Cause A command has been issued that is unknown to this subsystem.

Effect SCF waits for the next command.

Recovery Use the HELP KERNEL command to list supported commands.

-00008

```
KRN E-00008 Internal error:  SPI buffer larger than expected for objname.
```

objname

is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause A command has been issued, and the byte count read by a file-system procedure (for example, READUPDATE) is smaller than the used-length value in the command buffer.

Effect SCF terminates.

Recovery As described under "If You Have to Call Your Service Provider" (page 145), save pertinent information and contact your service provider.

-00014

```
KRN E-00014 Unable To Obtain Memory: text for objname.
```

text

is a text string issued by SCF.

objname

is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause This subsystem is unable to obtain memory space. Typically, this error indicates a resource availability problem caused by a large amount of activity (assuming that the configuration is proper for the operating environment and that the associated hardware is functioning normally).

Effect SCF waits for the next command.

Recovery Attempt to determine and correct the problem, or wait awhile before reissuing the request.

-00016

```
KRN E-00016 Object objname is already in state state.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

state
is the state of the generic process.

Cause A request that would change the state of an object has been issued, but the object is already in that state.

Effect SCF waits for the next command.

Recovery Informational message only; no corrective action is needed.

-00017

```
KRN E-00017 Object objname not found
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause There are two possible causes:

1. The specified *objname* could not be found within this subsystem. This can mean that the object has not been added yet, or the command refers explicitly to an object and no object of that type is supported by this subsystem.
2. You have loaded a new version of the system configuration database and no record of *objname* exists.

Effect SCF waits for the next command.

Recovery Either add the missing object, correct the object name, or correct the object type. Then retry the original command.

-00018

```
KRN E-00018 Object type is not supported by this subsystem.
```

Cause This object type is not supported. An invalid, unknown, or unsupported object type has been encountered.

Effect SCF waits for the next command.

Recovery Correct the object type and retry the command.

-00019

```
KRN E-00019 Invalid object name: objname
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause There are two possible causes:

1. An invalid template for *objname* was found in the command buffer.
2. You have loaded a new version of the system configuration database and no record of *objname* exists.

Effect SCF waits for the next command.

Recovery Take one of these actions, depending on the cause:

1. Correct the object name syntax and retry the command.
2. Either add the missing object, correct the object name, or correct the object type. Then retry the original command.

-00022

```
KRN E-00022 Security violation for objname.
```

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause A sensitive command has been entered by a user unauthorized to issue such a command. Unless this subsystem has assumed control over command security, SCP generates this error.

Effect SCF waits for the next command.

Recovery If possible, acquire the proper security and retry the command.

-00023

```
KRN E-00023 Internal Error:  SPI error err-num text
on call-name (Tnm: token-number).
```

err-num
is an SPI error number.

text
is a text string issued by SCF.

call-name
is an SPI procedure call.

token-number
is an SPI token number.

Cause An error occurred having to do with a call to an SPI procedure.

Effect SCF terminates.

Recovery As described under "If You Have to Call Your Service Provider" (page 145), save pertinent information and contact your service provider.

-00024

```
KRN E-00024 Internal Error:  Invalid subsystem ID: Owner:
KRN Number: subsystem-number Version: version-number.
```

subsystem-number
is the subsystem number of this subsystem manager process.

version-number
is the version number of this subsystem manager process.

Cause The subsystem ID specified (using a call to SSINIT) in the request is invalid.

Effect SCF terminates.

Recovery As described under "If You Have to Call Your Service Provider" (page 145), save pertinent information and contact your service provider.

-00026

```
KRN E-00026 Internal Error:  Invalid Token:  Token Code: hexadecimal-token-code
(Tnm: token-number).
```

hexadecimal-token-code
is a hexadecimal SPI code.

token-number
is an SPI token number.

Cause An invalid, unknown, or unsupported token code has been encountered.

Effect SCF terminates.

Recovery As described under "If You Have to Call Your Service Provider" (page 145), save pertinent information and contact your service provider.

-00027

```
KRN E-00027 Internal Error: Duplicate Token:
Token Code: hexadecimal-token-code (Tnm: token-number).
```

hexadecimal-token-code
is a hexadecimal SPI code.

token-number
is an SPI token number.

Cause The command buffer contains more than one occurrence of a token that can be specified only once.

Effect SCF terminates.

Recovery As described under "If You Have to Call Your Service Provider" (page 145), save pertinent information and contact your service provider.

-00029

```
KRN E-00029 Internal Error: Required Token Missing:
Token Code: hexadecimal-token-code (Tnm: token-number).
```

hexadecimal-token-code
is a hexadecimal SPI code.

token-number
is an SPI token number.

Cause A command has been issued in which a required token is missing.

Effect SCF terminates.

Recovery As described under "If You Have to Call Your Service Provider" (page 145), save pertinent information and contact your service provider.

-00030

```
KRN E-00030 Internal Error:  Invalid Value: text
(Tnm: token-number, Offset: offset-value for objname.
```

token-number
is an SPI token number.

offset-value
is an offset value.

objname
is the new name of a generic process, in the form $ZZKRN.#*gpname*.

Cause An illegal value has been encountered in a supported token.

Effect SCF terminates.

Recovery Recovery action depends on the token in error and on the value specified in that token. Refer to the SCF reference manual for the specific subsystem for information about recovery from this error by that subsystem.

Index

Symbols

A

B

C

D

P

Q

R

S

T

U

V

W

X

Y