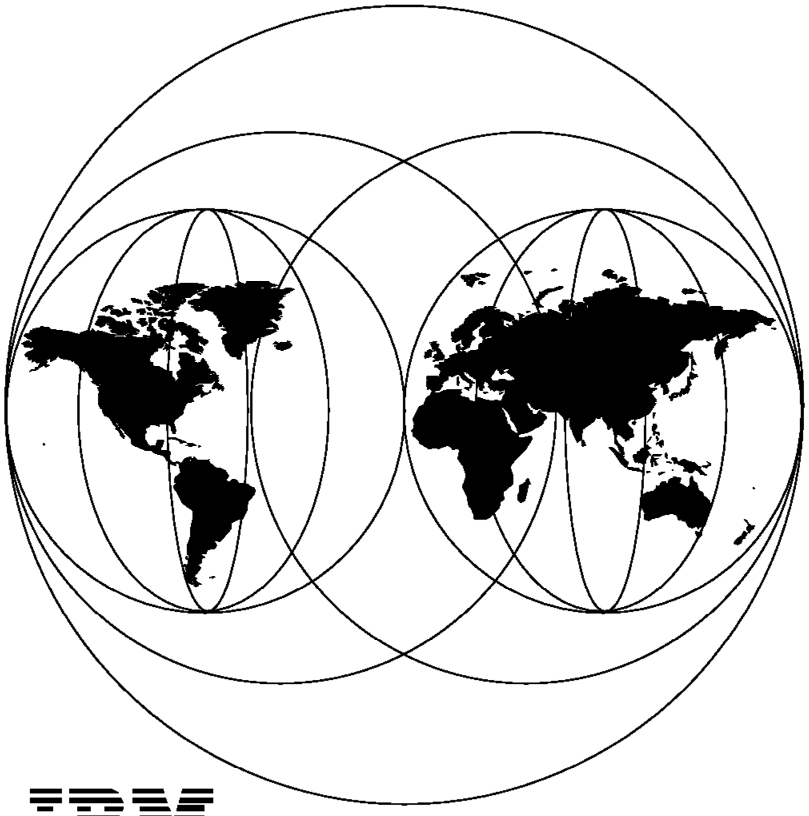
SG24-4928-00

RS/6000 SP PSSP 2.2 Survival Guide

August 1997



IBM

International Technical Support Organization

Poughkeepsie Center

International Technical Support Organization SG24-4928-00

RS/6000 SP PSSP 2.2 Survival Guide

August 1997

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix G, "Special Notices" on page 309.

First Edition (August 1997)

This edition applies to PSSP Version 2, Release 2 for use with the AIX Version 4 Operating System

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

Contents

Figures

Tables

Preface

This redbook covers problem determination for the POWERparallel System Support Programs Version 2.2. It is written for people who deal with RS/6000 SP support and management. It provides various scenarios and examples of problem determination for different subsystems, including Topology Services, Group Services, Event Management, and Problem Management.

The redbook also includes a chapter on installation and customization problems. Several examples are given, along with a complete explanation of the installation and customization process.

The last chapter provides a comprehensive procedure for problem determination for the SP Switch. It also offers a case study, and includes diagrams listing additional information about most problems that could occur in the Switch.

All examples and scenarios included in the redbook were tested at the ITSO Poughkeepsie Lab, along with the proposed solution procedures.

This redbook is essential documentation for people who support the RS/6000 SP or provide installation and customization services. System managers responsible for the operation and administration of RS/6000 SP sites will also find it very useful in their daily work.

Since the book was written for RS/6000 SP specialists, some familiarity with PSSP and AIX is assumed.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Poughkeepsie Center.

Marcelo R. Barrios is an Advisory International Technical Support Organization (ITSO) Specialist for RS/6000 SP at the Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on all areas of RS/6000 SP. Before joining the ITSO, Mr. Barrios worked as an assistant professor in the Electronics Department of Santa Maria University, Valparaiso, Chile. In 1993, he joined IBM as a Marketing Specialist in the RS/6000 Unit, IBM Chile.

Ewa Frisch is a certified AIX systems specialist in the AIX System Support Center, IBM Austria. After her study at Vienna University she joined IBM Austria and worked for two years in the education center as a UNIX and AIX instructor. For the last four years she has worked at the AIXSSC in Vienna, where her responsibilities include giving customers technical support on AIX, mainly on RS/6000 SP machines.

Yasuhiro Saitoh is an IT specialist at IBM Japan Systems Engineering Co., Ltd. in Makuhari (Japan). He joined IBM Japan in 1991, around the announcement of AIX V3.2. He has been working for a division that provides second-level support for AIX. He is mainly responsible for RS/6000 SP and AIX OS itself, especially the GUI.

Alex Wood is a certified AIX systems specialist in the RS/6000 Support Center, IBM UK Ltd. After joining IBM a mere two and a half years ago with his only grounding in the PC market, he has concentrated mainly on AIX and the RS/6000 SP software products, with his expertise coming from working with customers on site and remotely on post-sale technical issues. His main passion, however, does not lie with the IT industry, but with the sporting scene, specifically with the gentlemen′s (!) game of Rugby.

Jun Nakano is an IT specialist at RS/6000 Product Management and Marketing, IBM Japan. He joined IBM in 1990 and studied combinatorial optimization and computational geometry for five years at Tokyo Research Laboratory. He has been working with SP since 1995, supporting technical/commercial benchmarks, installations, and education. His interest includes algorithms, architectures, computer networks, and operating systems.

Karel Coufal is an IT specialist at the Availability Services department of IBM Czech Republic. After joining IBM in 1995, he has concentrated mainly on service support for RS/6000 SP, HACMP for AIX, and Internet Solutions on the AIX platform.

Marco Vallone is a specialist in the AIX Support Center of Rome, Italy. He joined IBM in 1989 as Product Engineer at S.Palomba, the manufacturing plant of RS/6000 in Italy. He is the focal point for the RS/6000 SP software in his country.

Thanks to the following people for their invaluable contributions to this project:

Bob Fartfai
IBM Poughkeepsie

Richard Ferri
IBM Poughkeepsie

Bob Simon
IBM Poughkeepsie

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 321 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

For Internet users	`http://www.redbooks.ibm.com`
For IBM Intranet users	`http://w3.itso.ibm.com`

- Send us a note at the following address:

  `redbook@vnet.ibm.com`

Chapter 1. RS/6000 SP Infrastructure

In this chapter we divide the SP infrastructure discussion into two areas: hardware and software.

Although the hardware structure did not change much in PSSP 2.2, except maybe for the inclusion of the High Node, this chapter describes some basic hardware topics, especially those related to node numbering.

The software discussion contains a brief overview of the software components of an SP, and then deals with the new subsystems, such as Topology Services, Group Services, and Event Management. It also contains several examples of how to diagnose problems with these subsystems, and proposes solutions and corrective procedures.

1.1 Hardware Structure

A standard RS/6000 SP system is composed of a frame or multiple frames, nodes that are packaged inside the frame, a switch (also inside the frame), and a Control Workstation (CW). (Refer to the technical reference for supported models.) The SP system can be managed as a single unit through the CW.

1.1.1 Hardware Components

There are many hardware components that make the RS/6000 SP system more complex and powerful than a conventional RS/6000 machine, although it is built with the standard AIX and RS/6000 hardware parts.

Control Workstation

The Control Workstation is the console of the SP System. It provides facilities for system installation, managing user accounts, hardware and software configuration, error logging, mounting external file systems, system partitioning, and many other functions. The Control Workstation is connected, with the supervisor card, to each frame via an RS-232 line, and each node has a node supervisor card that is connected to the frame supervisor card. The CW provides hardware control functions via this RS-232 connection. The CW is connected to each node by the SP Ethernet. This network is usually dedicated for SP traffic only. A second CW can be designated as a backup connected by the new supervisor card to the CW by using the High Availability Control Workstation (HACWS) software option.

Network Connectivity Adapters

Network connectivity is supplied by various adapters (some optional) that can provide connection to I/O devices, networks of workstations, and mainframe networks. Ethernet, FDDI, Token-Ring, HiPPI, SCSI, FCS, and ATM are examples of adapters that can be used as part of an SP system.

Frames

There are two types of frames: 49-inch frames (short) that contain four drawers with eight slots, and 79-inch frames (tall) that contain eight drawers with 16 slots. One drawer has two slots available. An SP frame can contain all three types of nodes mixed together, but only tall frames support 604 High Nodes. Up to three expansion frames (a frame without a switch) can be connected to a single switched frame (a frame with a switch installed). If you attach more than two frames, or if the two serial ports on the CW are needed for other uses, then you must add a multiport serial adapter to the CW to provide the necessary connection points.

Processor Nodes

The SP system is scalable from two to 512 processor nodes that can be contained in multiple frames.

Thin Nodes

Thin nodes are half the size of wide nodes, and can be packaged eight to a 49-inch frame and 16 to a 79-inch frame. Each node uses one slot. Thin nodes are built with Power2 processors (66 Mhz), or Power2 Super Chip processors (120 Mhz).

Wide Nodes

Wide nodes can be packaged four to a 49-inch frame and eight to a 79-inch frame. Each node uses two slots. Wide nodes are built with Power2 processors (77 Mhz), or Power2 Super Chip processors (135 Mhz).

604 High Nodes

604 High nodes can be packaged four to a 79-inch frame and *cannot* be packaged in 49-inch frames. Each node uses four slots in two drawers. High nodes are built with 604 SMP processors 2- to 8-way (112 Mhz). 604 High nodes cannot be connected to the High Performance Switch-LC8 or the SP Switch-8.

Switch

The Switch connects each node via the Switch Adapter to one or multiple frames. Each Switch Board has 16 ports to connect to 16 nodes. To connect more than 16 nodes, another switch and another frame are needed. Only one switch per frame is allowed. The Switch provides low-latency, high-bandwidth communication between nodes. It consists of a switch assembly and the internal cables to support connections to processor nodes in a system. The Switch speeds up TCP/IP, file transfers, remote procedure calls, and database functions.

1.1.2 Node Numbering

Frame Numbers

Frame numbers are established by the system administrator when the system is installed. Each frame is referenced by the TTY port on the CW to which the frame supervisor cable is attached, and is assigned a numeric identifier. The order in which the TTYs are identified defines the sequence in which frames will be examined during the configuration process. This order is used to assign global identifiers to the switch port and nodes. This is also the order used to determine which frames share a switch. Nodes in frames that do not contain any switch board will be attached to the switch board in the first preceding frame that has a switch board. The frames without switch boards are called *expansion frames*. A maximum of three expansion frames are supported per switch.

Expansion frame numbers must follow the number of the frame with the switch board to which their nodes are connected. Thus, if n is the frame number of a frame with a switch and there are three expansion frames connected to the same switch, their numbers will be n+1, n+2, and n+3.

Within switch-only frames, called *switch-expansion* frames, switch boards should be numbered consecutively, from bottom to top, beginning with 1001.

The High Performance Switch Expansion frame or the SP Switch Expansion frame should be the last frame in the system. Frame numbers do not have to be contiguous. You should assign a frame number to the switch expansion frame that allows for the future addition of node frames.

Slot Numbers

Slots in the same drawer have consecutive numbers. Slot numbers only have meaning within the context of a specific frame.

Node Numbers

A node number is a global ID assigned to a node. It is the primary means by which the administrator references a specific node in the system. Node numbers are assigned by the following formula:

```
node_number = (16 x (frame_number - 1)) + slot_number
```

where `slot_number` is the lowest slot number occupied by the node. Each type (size) of node occupies a consecutive sequence of slots. For each node there is an integer, n, such that a thin node occupies slot n, a wide node occupies slots n, n+1, and a 604 high node occupies n, n+1, n+2, n+3. For wide and high nodes, n must be odd.

Note: In a single frame any combination of nodes is allowed. Node 0 refers to the Control Workstation.

Switch Port Number

A switch port number is a global identifier for a port on the switch. It does not identify the physical node that it is attached to. A switch port can be thought of as a connector on the back of a switch board in an SP frame.

Therefore, the switch port number is the global ID assigned to each switch port that is unique across the RS/6000 SP. It is used internally by the PSSP software as a direct index into the switch topology file and to determine routes between switch ports. If a node is connected to the switch, then its switch node number is the same as the number of the switch port it is connected to. The SDR contains information for each node to identify the switch port number that it is attached to. Since there is no automatic method for determining how nodes are wired to switch ports, fixed rules have been defined and implemented in SDR_config. You must follow these rules to properly configure and partition the system. The rules are such that if you add an expansion frame to an existing system, the existing switch node numbers are not changed; you only add new switch node numbers to the topology files.

If node is:

- Connected to a switch within its frame, then

  `switch_port_number = (switch_number - 1) X 16 + (slot_number -1).`

- Connected to a switch outside of its frame (Expansion Frame), then

  `switch_port_number = (switch_number - 1) X 16 + port_number.`

Here, `switch_number` is the number of the switch board to which the node is connected, and `port_number` is the port position on the switch board to which the node is connected.

High Performance Switch-LC8 and SP Switch-8 Port Numbering

For High Performance Switch-LC8 and SP Switch-8, a different algorithm is used for assigning nodes their node numbers and switch port numbers. A system with this type of switch contains only switch port numbers 0 through 7.

The following algorithm is used to assign nodes their switch port numbers for systems with the High Performance Switch-LC8 and SP Switch-8:

1. Assign the node in slot 1 to switch_port_number = 0, and increment switch_port_number by 1.
2. Check the next slot. If there is a node in the slot, assign it the current switch_port_number, then increment the number by 1.

Repeat until you reach the last slot in the frame or switch port number 7, whichever comes first.

1.2 Software Structure

The subdirectories that contain the Parallel System Support Program software images (PSSP), AIX 4.x images, and PTFs needed to install or migrate the nodes are located in the /spdata/sys1/install directory.

In PSSP 2.2, new components are implemented that make the RS/6000 SP even more robust and reliable. These new components are often referred to as the new High Availability (HA) Infrastructure.

This section describes the new HA subsystems, daemons and directories.

1.2.1 Directory Structure

Directories under /spdata/sys1/install are manually created before the PSSP installation process is started. The installation process requires the lppsource, images, pssp, and pssplpp directories under /spdata/sys1/install. Any deviation from these names results in installation failure.

/spdata/sys1
This is the main directory for the System Data Repository (SDR), SP monitor, log management, partition layout files, microcode, and installation.

/spdata/sys1/install
This is the main directory for PSSP installation.

/spdata/sys1/install/aix414/lppsource
This directory contains the required AIX4.1 file sets (the standard name is “default,” not AIX414).

/spdata/sys1/install/aix42/lppsource
This directory contains all AIX 4.2 images that are needed to install and migrate a node to AIX 4.2. (only needed when using AIX4.2 on the nodes).

/spdata/sys1/install/images
This directory contains the spimg image for AIX and system backup images for the nodes. The spimg image is a single file containing a system backup (mksysb) image of a minimal AIX system, and is shipped with the PSSP code.

/spdata/sys1/install/pssp
This directory contains the Network Installation Management (NIM) configuration files for the nodes.

/spdata/sys1/install/pssplpp/PSSP-2.2
All PSSP 2.2 software images, PTFs, and SP system file sets reside in this directory.

/spdata/sys1/install/pssplpp/PSSP-2.1
PSSP 2.1 software images, PTFs, and SP system file sets reside in this directory.

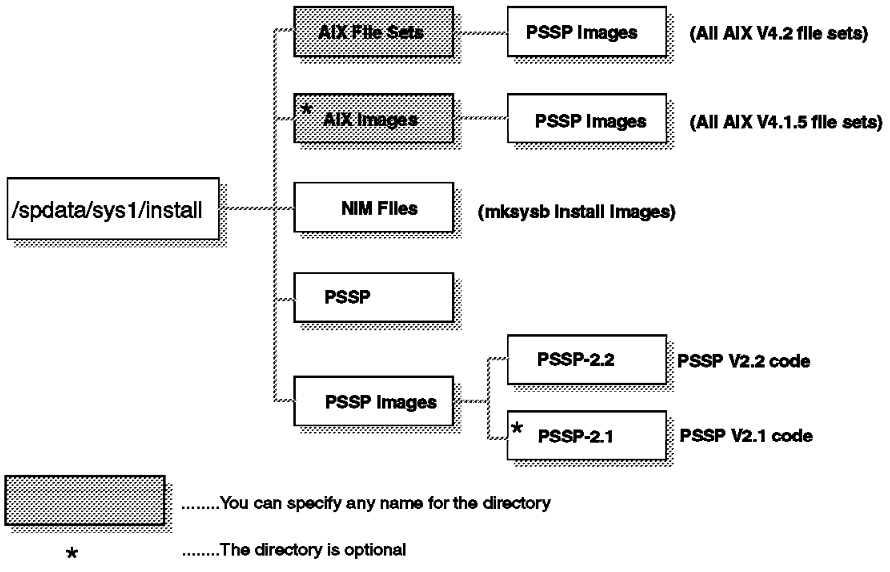


Figure 1. Directory Structure

1.2.2 The New High Availability Infrastructure

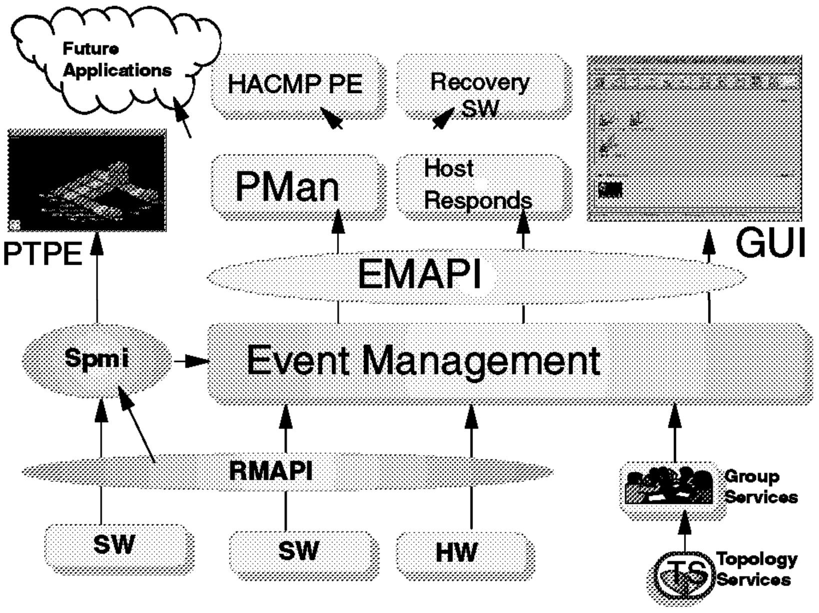


Figure 2. High Availability Infrastructure

Figure 2 provides an overview of the IBM HA Infrastructure components and the interrelationships that exist within their internal structures.

The purpose for adding these new components to PSSP is to guarantee a robust commercial environment on the SP system, that is, to offer high availability of the system on the one hand, and good performance on the other.

The main components of the new code are:

- Topology Services
- Group Services
- Resource Monitors
- Event Management
- Problem Management
- Application Programming Interfaces (API)

In this chapter we focus on the three components included in the ssp.ha file set. They are:

- Topology Services
- Group Services
- Event Management

For more information about other subsystems, refer to *RS/6000 Monitoring: Keeping It Alive*, SG24-4873 and *RS/6000 SP High Availibility Infrastructure*, SG24-4838.

What are the functions of these three subsystems?

Topology Services is a partition-sensitive subsystem that has many goals. One of them is to maintain the information about the topology of the networks and the accessibility of the communication adapters and processors by querying the infrastructure. The main lists used by Topology Services are:

- Machine list
- Adapter list
- Connectivity list

Other duties of Topology Services are:

- Heartbeat monitoring
- Providing the routing information for *Reliable Messaging* (on an application level using the connectivity list)
- Building a foundation for the HA Infrastructure

Group Services (GS) provides for coordinating and monitoring changes to the state of the subsystem running on the nodes belonging to the GS domain. This is mostly a set of nodes within a system partition. Any hagsd in the domain can ask to become a member of a group (provider) or to monitor the group (subscriber). If the provider is the subscriber, then we are talking about "supplicants."

While the GS subsystem is being initialized, each process in a subsystem asks to join the group. As a member of the group it can receive different types of messages from GS called notifications, specifically: proposal notifications, final notifications, announcement notifications, liveness notifications and query notifications.

The information about the state of the nodes is called *processor membership*. The information about the state of the communications networks is maintained in *network membership*. These two groups are predefined system membership groups. Providers can change the membership list by joining or leaving the group. They can also specify the level of consistency associated with the modification. Providers can leave the group voluntarily or involuntarily (when the provider process at the node on which it was running fails).

You may now have the impression of redundancy of functions and ask "why do we need GS"? The reason for having GS provide system membership groups is to support reliable recovery actions and have strict consistency between these actions. It is important that all processes involved in these actions have exactly the same information about the changes being performed. And while the group members have the same knowledge about the order of notifications, it is not necessary to replicate this function in the TS layer.

Event Management matches the information about the state of system resources with the information about resource conditions that are of interest to the client programs.

The system partition-sensitive subsystems provide the following services:

hats The Topology Services subsystem, which provides information to other PSSP subsystems about the state of the nodes and adapters on the SP system.

hb	The heartbeat subsystem, which communicates with several PSSP subsystems as part of providing information about the state of the nodes and adapters on the SP system.
hags	The Group Services subsystem, which provides a general purpose facility for coordinating and monitoring changes to the state of a client program that is running on a set of nodes in an SP system.
haem	The Event Management subsystem, which compares information about the state of system resources provided by resource monitors with requests for information by client programs, and creates events that notify the clients when a match occurs.
hr	The Host_Responds subsystem, which provides information about the state of the nodes and adapters on the SP system.
pman	The Problem Management subsystem, which allows the installation to be notified when events related to Problem Management occur.
emon	The Emonitor subsystem, which monitors the status of nodes in a switch network, bringing a node back into the switch network when necessary. This subsystem is not started with the `syspar_ctrl` command, but by using the `Estart -m` command.
sp_configd	The SNMP support portion of the Problem Management subsystem, which creates and sends SNMP traps when selected types of errors are recorded.
emcond	The Event Management subsystem loads the SDR with predefined events or conditions of interest. These conditions are used by the SP Perspectives GUI.

1.2.3 Information about the Daemons of the HA Subsystems

The HA subsystems run under the control of the System Resource Controller (SRC) and have associated SRC group and subsystem names. Each subsystem has a control script. Each subsystem is associated with one or more daemons.

For each system partition, one instance of a subsystem is running on the Control Workstation and one instance is running on each node belonging to the partition. On the Control Workstation there are multiple instances of the subsystems, one for each partition that is defined.

Table 1 shows the information about the HA subsystems related to the names of the components used by them:

Table 1. HA Subsystems

Subsystem Name		Subsystem	Daemons	SRC Group	Control Script
CWS	Node				
hats.<syspar_name>	hats	Topology Services	hatsd	hats	hatsctrl
hags.<syspar_name> hagsglsm.<syspar_name>	hags hagsglsm	Group Services	hagsd hagsglsmd	hags hags	hagsctrl hagsctrl
haem.<syspar_name>	haem	Event Management	haemd	haem	haemctrl

The subsystem names on the Control Workstation end with the system partition name, as follows:

```
Subsystem         Group            PID     Status
 hb.sp2en3        hb               15474   active
 hb.sp2en0        hb               22154   active
 hb.sp2en1        hb               22702   active
 hb.sp2en2        hb               23820   active
 hr.sp2en3        hr               22848   active
 hr.sp2en0        hr               23388   active
 hr.sp2en1        hr               21662   active
 hr.sp2en2        hr               17334   active
 hags.sp2en0      hags             61782   active
 hagsglsm.sp2en0  hags             25654   active
 hags.sp2en1      hags             26698   active
 hagsglsm.sp2en1  hags             24158   active
 hags.sp2en2      hags             27502   active
 hagsglsm.sp2en2  hags             20134   active
 hags.sp2en3      hags             62512   active
 hagsglsm.sp2en3  hags             62800   active
 pman.sp2en3      pman             61232   active
 pmanrm.sp2en3    pman             17690   active
 pman.sp2en1      pman             27216   active
 pmanrm.sp2en1    pman             21092   active
 pman.sp2en2      pman             60280   active
 pmanrm.sp2en2    pman             19316   active
 pman.sp2en0      pman             13462   active
 pmanrm.sp2en0    pman             48688   active
 haem.sp2en3      haem             44774   active
 haem.sp2en0      haem             40570   active
 haem.sp2en1      haem             21948   active
 haem.sp2en2      haem             28876   active
 hats.sp2en0      hats             11248   active
 hats.sp2en1      hats             26558   active
 hats.sp2en2      hats             29898   active
 hats.sp2en3      hats             61052   active
 sdr.sp2en1       sdr              23220   active
 sdr.sp2en2       sdr              34492   active
 sdr.sp2en3       sdr              41924   active
 sdr.sp2en0       sdr              35952   active
 Emonitor.sp2en3  emon                     inoperative
 Emonitor.sp2en0  emon                     inoperative
 Emonitor.sp2en1  emon                     inoperative
 Emonitor.sp2en2  emon                     inoperative
```

Note

Even though the SDR subsystem appears in the output for the system partition, the SDR is not managed by the Syspar Controller. Also, the Emonitor subsystem is not started by the `syspar_ctrl` command. Therefore, its status is inoperative until it is started by the `Estart -m` command.

On the node there is only one instance of the subsystem running. That is why there is no need to use the partition name as part of the subsystem name running on the node:

```
 hats             hats             8822    active
 hags             hags             19572   active
 hagsglsm         hags             18896   active
 pman             pman             15502   active
 pmanrm           pman             9290    active
 Emonitor.sp2n16                           inoperative
```

Note

If PSSP 2.1 is running on the node, you will see an entry like the following for the heartbeat daemon (hbd) in the previous output:

```
hb              hb              11276   active
```

Figure 3 shows the HA daemons running on the Control Workstation and on the nodes belonging to the partitions with different levels of the PSSP software.

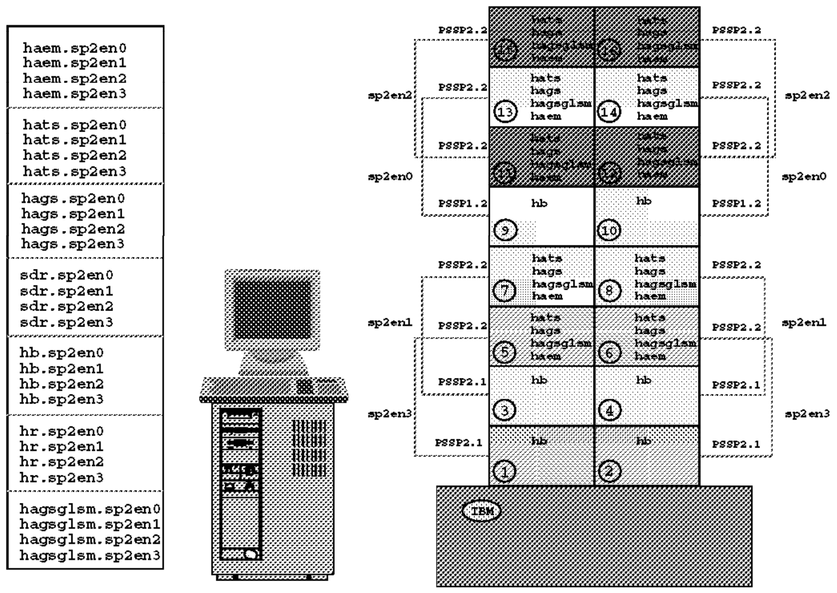


Figure 3. HA Daemons and Coexistence

1.2.3.1 hatsd

The hats daemon is initially started on the Control Workstation by the System Resource Controller (SRC), regardless of the level of the system partition. If hatsd fails, the SRC will respawn it automatically. The SP_NAME environment variable causes selection of the correct topology configuration. There is one hatsd per node and it is associated with the system partition to which the node belongs. On the Control Workstation, there are multiple instances of each subsystem, one for each system partition.

Initialization of the Topology Services daemon does the following:

- Gets the IP address of the partition using the `spget_syspar` command.
- Gets the name of the partition using the `spget_syspar -n` command.
- Sets the full name of the subsystem (such as, hats.sp2en2).
- Checks for the existence of the /var/ha/run directory, and if it does not exist, creates it.
- Checks for the existence of the /var/ha/log directory, and if it does not exist, creates it.

- Creates the machines.lst file for this partition using the information from the SDR and the netstat command, as follows:

```
*Timestamp(DDHHMMSS)=2010/14/96M51
TS_Frequency=1
TS_Sensitivity=4
TS_FixedPriority=38
Network SPether
*
*Node Type Address
    0 en0  192.168.3.37
   11 en0  192.168.3.11
   12 en0  192.168.3.12
   15 en0  192.168.3.15
   16 en0  192.168.3.16
Network SPswitch
*
*Node Type Address
   11 css0 192.168.13.11
   12 css0 192.168.13.12
   15 css0 192.168.13.15
   16 css0 192.168.13.16
```

- Removes the oldest log files.
- Enables IP source routing using the command:

  ```
  # no -o nonlocsrcroute=1
  ```

  This option is required to support Reliable Messaging and should not be removed, commented out or disabled.
- Starts the daemon itself.

Important

The hats script takes advantage of the `no -o nonlocsrcroute=1` command. It enables IP source routing. Do not change this "no" option because TS and other subsystems depend on this setting; if it is changed, they may not work properly.

It is also important to know from where the daemon takes the input, where the output goes, and what the error log files of this daemon are. Information about the definitions of standard input, standard output, standard error, and flags that are set for the hats daemon can be obtained by using the following command:

```
# lssrc -S -s hats
#subsysname:synonym:cmdargs:path:uid:auditid:standin:standout:standerr:action
:
mullti:contact:svrkey:svrmtype:priority:signorm:sigforce::display:waittime
:grpname:
hats:::/usr/lpp/ssp/bin/hats:0:0:/dev/console:/var/ha/log/hats.sp2en2:
/var/ha/log/hats.sp2en2:-R:-Q:-K:0:0:20:0:0:-d:30:hats:
```

Let us look at some information in this output:

```
subsysname       hats
synonym          there is none
cmdargs          there are none
path             /usr/lpp/ssp/bin/hats
uid              0
stdin            /dev/console
stdout           /var/ha/log/hats.sp2en2
stderr           /var/ha/log/hats.sp2en2
action           -R means that the system will be restarted every time
                 after it terminated abnormally
multi            -Q means that multiple instances of the subsystems are  not
                 allowed to run at the same time and the subsystem is not
                 to share the same Interprocess Communication (IPC) queue.
contact          -K means that the subsystem uses sockets as its
                 communication method
priority         20
group name       hats
```

1.2.3.2 hagsd

The Group Services daemon provides services for coordinating and monitoring the changes to the state of client programs that run on a set of nodes within an RS/6000 SP system. Like other HA daemons, hagsd is normally started by an entry in the /etc/inittab file via the `startsrc` command. If necessary, the Group Services daemon can be started using the `hagsctrl` command or the `startsrc` command directly.

If the hags daemon fails, the SRC will respawn it automatically. There is one hagsd per node and it is associated with the system partition to which the node belongs. On the Control Workstation, there are multiple instances of the daemon, one for each system partition. For more information about the behavior of hagsd and its relationship with Topology Services, see 1.3.1.1, “What Happens If hatsd Dies?” on page 26.

The initialization of the Group Services daemon does the following:

- Gets the number of the node on which it is running by using the `/usr/lpp/ssp/install/bin/node_number` command. Node 0 is the Control Workstation.
- Fetches the name of the system partition from the Syspar SDR class.
- Tries to connect the TS subsystem. If the connection cannot be established because the TS subsystem is not running, it is scheduled to be retried again. This continues until the connection to TS is established. Meanwhile, the GS daemon writes an AIX error log entry every minute and no clients may connect to the GS subsystem.
- Establishes connections with the SRC subsystem so that the status can be returned in response to SRC commands.
- Establishes Group Services, which is a set of nodes within the SP system partition in which the GS daemon is executing. At this point one of the GS nodes is nominated to be a GS nameserver. Until the domain is established, no client can request to join or subscribe to a group.
- Enters the main control loop. In this loop the GS daemon waits for requests from GS clients, messages from other GS daemons, messages from TS, and requests from SRC for status.

What about the definitions of standard input, standard output, and standard error? Use the following command:

```
# lssrc -S -s hags
#subsysname:synonym:cmdargs:path:uid:auditid:standin:standout:standerr:
action:multi:contact:svrkey:svrmtype:priority:signorm:sigforce:display:
waittime:grpname:
hags:::/usr/lpp/ssp/bin/hagsd:0:0:/dev/console:/var/ha/log/gs.default.sp2en2:
/var/ha/log/gs.default.sp2en2:-R:-Q:-K:0:0:20:0:0:-d:30:
hags:
```

Let us extract the most important information from this output:

subsysname	hags
synonym	there is none
cmdargs	there are none
path	/usr/lpp/ssp/bin/hagsd
uid	0
stdin	/dev/console
stdout	/var/ha/log/gs.default.sp2en2
stderr	/var/ha/log/gs.default.sp2en2
action	-R means that the system will be restarted every time after it terminated abnormally
multi	-Q means that multiple instances of the subsystems are not allowed to run at the same time and the subsystem is not to share the same Interprocess communication (IPC) queue.
contact	-K means that the subsystem uses sockets as its communication method
priority	20
group name	hags

1.2.3.3 haemd

The Event Management daemon, haemd, runs on each node of a system partition and on the Control Workstation. If there is more than one system partition, then multiple daemons run on the Control Workstation, one per system partition.

The operational domain of the Event Management subsystem is a system partition. The Control Workstation takes part in each domain. Like other HA daemons, haemd gets started by an entry in the /etc/inittab file using the `startsrc` command. If necessary, haemd can be started by using the `haemctrl` command or the `startsrc` command directly. The Event Management subsystem is contained in the ssp.ha file set, and the EMAPI libraries are contained in the ssp.clients file set.

The initialization process of the Event Management daemon does the following:

- Gets the number of the node on which it is running by using the `/usr/lpp/ssp/install/bin/node_number` command. Node 0 is the Control Workstation.
- Fetches the name of the system partition and the EMCDB version string from the Syspar SDR class.
- Performs actions that are necessary to become a daemon. This includes establishing communications with the SRC subsystem so that it can return status in response to SRC commands.

- Removes from the registration cache all the subdirectories for local EM clients that no longer exist. That is, if the process ID in the subdirectory name cannot be found, it removes that subdirectory.
- Tries to connect to the Group Services subsystem. If the connection cannot be established because the subsystem is not running, it is scheduled to be retried in five seconds. This continues until the connection to GS is established. Meanwhile, Event Manager daemon initialization continues.
- Enters the main control loop. In this loop, the Event Manager daemon waits for requests from EM clients, messages from resource monitors and other EM daemons, messages from the GS subsystem, and requests from the SRC for status. It also waits for internal signals that indicate a function that was previously scheduled should now be executed, for example, retrying a connection to GS.

  What about the definitions of standard input, standard output, and standard error? Use the following command:

```
# lssrc -S -s haem
#subsysname:synonym:cmdargs:path:uid:auditid:standin:standout:standerr:
action:multi:contact:svrkey:svrmtype:priority:signorm:sigforce:display:
waittime:grpname:
haem:::/usr/lpp/ssp/bin/haemd:0:0:/dev/console:/dev/null
:/var/ha/log/em.default.sp2en2:
-R:-Q:-K:0:0:20:0:0:-d:120:haem:
```

Let us extract the most important information from this output:

susbsysname	haem
synonym	there is none
cmdargs	there are none
path	/usr/lpp/ssp/bin/haemd
uid	0
stdin	/dev/console
stdout	/dev/null
stderr	/var/ha/log/em.default.sp2en2
action	-R means that the system will be restarted every time after it terminated abnormally
multi	-Q means that multiple instances of the subsystems are not allowed to run at the same time and the subsystem is not to share the same Interprocess communication (IPC) queue.
contact	-K means that the subsystem uses sockets as its communication method
priority	20
group name	haem

Important for Performance

The Event Manager daemon has an internal limit of 256 open file descriptors. Practically, this means that the number of EM client sessions, local or remote, is limited to about 225. This is the limit of the file descriptors per daemon and not the limit of the number of EM clients in the operational domain.

1.2.4 How to Manage the Partition-Sensitive Subsystems

There are at least three methods for controlling and managing the new HA subsytems:

1. Using the control commands, that is, `hatsctrl`, `hagsctrl`, `haemctrl`, `emonctrl`, `hbctrl`, `hrctrl`, `pmanctrl`, and `sp_configdctrl`

   These Korn shell scripts manage the corresponding daemons as the SRC subsystems. Their location is the /usr/lpp/ssp/bin directory. They provide a variety of controls for operating the corresponding subsystem, and they all use the same options for the same action. Let us have a look at the options of the *hatsctrl* script:

   ```
   hatsctrl -a | -s | -r | -k | -d | -c | | -t | -o | -h | -?

   -a      add topology services
   -d      delete topology services
   -k      stop topology services
   -c      clean topology services(delete from all partitions)
   -r      refresh topology services configuration
   -s      start topology services
   -t      turn on tracing for topology services
   -o      turn off tracing for topology services
   -h|?    this help message
   ```

2. Using the SRC commands

   System partition-sensitive subsystems operate under the control of the SRC and have associated SRC group and subsystem names. Have a look at Table 1 on page 8. For managing and controlling partition-sensitive subsystems, you can also use the SRC commands `startsrc -g <subsystem_name>` to start a subsystem, and `stopsrc -g <subsystem_name>` to stop it. To get more information about the subsystem, you can use the `lssrc` command:

```
# lssrc -g hags
Subsystem         Group            PID     Status
 hagsglsm         hags             18896   active
 hags             hags             19572   active
# lssrc -S -s hats
#subsysname:synonym:cmdargs:path:uid:auditid:standin:standout:standerr:
action:
mullti:contact:svrkey:svrmtype:priority:signorm:sigforce::display:waittime
:grpname:
hats:::/usr/lpp/ssp/bin/hats:0:0:/dev/console:/var/ha/log/hats.sp2en2:
/var/ha/log/hats.sp2en2:-R:-Q:-K:0:0:20:0:0:-d:30:hats:
# lssrc -ls haem
Subsystem         Group            PID     Status
 haem             haem             7404    active

Trace flags set:  None
 Configuration Data Base version: 854067075,797629440,0(SDR)

 Daemon started on 02/22/97 at 18:16:04.849851392
    running 0 days, 4 hours, 6 minutes and 1 seconds
 Daemon connected to group services: TRUE
 Daemon has joined peer group:       TRUE
 Daemon communications enabled :     TRUE
 Peer count:                         4

 Logical Connection Information
  Type   LCID   FD  Node/PID  Start Time
 peer       0            0
 local      2   12    15606   Sat Feb 22 18:16:20 1997
 local      3   14    10124   Sat Feb 22 18:28:14 1997

 Resource Monitor Information
   Resource Monitor Name      Type     FD    PID  Locked
 IBM.PSSP.harmld            server     15    6668    No
 IBM.PSSP.harmpd            server     11   17396    No
 IBM.PSSP.hmrmd             server     -1      -1    No
 IBM.PSSP.pmanrmd           client     13      -2    No
 Membership                 internal   -1      -2    No
 Response                   internal   -1      -2    No
 aixos                      internal   -1      -2    No

 Highest file descriptor in use is 15
.....
```

3. Using the `syspar_ctrl` commands

   The `syspar_ctrl` command is a Perl script. It acts as an interface to the system partition-sensitive subsystems, except the SDR, supporting the functions that are shared by all the subsystems. These functions can be, for example, to add/delete or start/stop a subsystem, and can be executed on the Control Workstation and the nodes.

   The options passed to this script are passed directly to the underlying subsystem′s control scripts (such as `hatsctrl`, `hagsctrl`, `haemctrl`, and so on).

   By default, all the control scripts listed in the /usr/lpp/ssp/config/cmi/syspar_subsystems file are called.

   Let us have a look at this file:

```
.....
# <subsystem_name>      <full_path_ctrl_script>
hats                    /usr/lpp/ssp/bin/hatsctrl
hb                      /usr/lpp/ssp/bin/hbctrl
hags                    /usr/lpp/ssp/bin/hagsctrl
haem                    /usr/lpp/ssp/bin/haemctrl
hr                      /usr/lpp/ssp/bin/hrctrl
pman                    /usr/lpp/ssp/bin/pmanctrl
emon                    /usr/lpp/ssp/bin/emonctrl
sp_configd              /usr/lpp/ssp/bin/sp_configdctrl
emcond                  /usr/lpp/ssp/bin/emconditionctrl
spdmd                   /usr/lpp/ptpe/bin/spdmdctrl
```

Invoking the syspar_ctrl command without specifying the susbsystem on which the operation should be done, will perform this operation for all partition-sensitive subsystems on this node. Following is an example how to use this command for a selected subsystem:

```
# syspar_ctrl -k haem
0513-044 The stop of the haem Subsystem was completed successfully.
```

The syspar_ctrl script is most often invoked through other PSSP commands, such as spapply_config, sprestore_config, and rc.sp, but it can also be invoked from the command line.

Following are the options you can use with the syspar_ctrl command, and their meaning:

`syspar_ctrl [-G] [-V] {-a|-d|-s|-k|-t|-o|-c|-h|-A|-D|-E|-R} [subsystem_name]`

-G Global: Invokes the appropriate underlying partition-sensitive subsystem′s control scripts for each system partition.

-V Verbose: Runs this script in verbose mode, which will print out the calls to the underlying control scripts and other important data.

-a Add: Adds all the subsystems, or if a subsystem_name is provided, just adds the specified subsystem. Each subsystem′s control script is invoked with the -a option, which results in adding the subsystem to the SRC subsystems /etc/inittab and /etc/services.

-d Delete: Deletes all the subsystems, or if a subsystem_name is provided, only the specified subsystem. Each subsystem′s control script is invoked with the -d option, which causes the subsystem to be deleted from the SRC subsystems /etc/inittab and /etc/services.

-s Start: Starts all the subsystems, or if a subsystem_name is provided, only the specified subsystem. Each subsystem′s control script is invoked with the -s option, which causes any daemons associated with this particular subsystem to be started.

-k Kill or stop: Stops all the subsystems, or if a subsystem_name is provided, only the specified subsystem. Each subsystem′s control script is invoked with the -k option, which causes any daemons associated with this subsystem to be stopped.

-t Trace on: Turns on the trace for all the subsystems, or if a subsystem_name is provided, only for the specified subsystem. Each subsystem′s control script is invoked with the -t option, which causes the trace in any daemon associated with this subsystem to be turned on.

Note

It is recommended that you only turn on the trace for a particular subsystem, otherwise the amount of data produced will rapidly fill up the /var file system.

-o Trace off: Turns off the trace for all the subsystems, or if a subsystem_name is provided, only for the specified subsystem. Each subsystem′s control script is invoked with the -o option, which causes each subsystem′s control script to turn the associated subsystem daemon′s trace off.

-c Clean: Cleans up after all the subsystems, or if a subsystem_name is provided, just after the specified subsystem. Each subsystem′s control script is invoked with the -c option, which causes each subsystem′s control script to stop the subsystem′s daemons and remove any entry made by this subsystem in SRC, /etc/inittab and /etc/services.

-A Add and start: Adds and starts all the subsystems, or if a subsystem_name is provided, only the specified subsystem. Each subsystem′s control script is invoked with the -a option followed by the -s option, which provides the same function as first calling syspar_ctrl with the -a option followed by the -s option.

-D Stop & delete: Stops and deletes all the subsystems, or if a subsystem_name is provided, only the specified subsystem. Each subsystem′s control script is invoked with the -k option followed by the -d option.

-E Examine: Examines all the subsystems, or if a subsystem_name is provided, only the specified subsystem in the Syspar Controller subsystems file. Each subsystem control script in the subsystems file is examined and displayed. Invalid entries are noted. An entry is invalid when the control script for a subsystem does not exist at the specified location, or does not have the correct read and execute permissions.

-R Restore: Restore all the subsystems, or if a subsystem_name is provided, only the specified subsystem. Stops and deletes, then adds and starts each subsystem. All subsystems are stopped and deleted before they are added and started. Each subsystem′s control script is invoked with the -k option followed by the -d option, then with the -a option followed by the -s option. This is a convenience option that provides the same function as first calling syspar_ctrl with the -D option followed by the -S option.

1.2.5 Getting Information about HA-Subsystems and Their Status

This is only a brief review of a set of commands to be used to get more detailed information about partition-sensitive subsystems. All these commands, and their options and outputs, are described in *RS/6000 SP High Availability Infrastructure*, SG24-4838, as well as in *PSSP Command and Technical Reference*, GC23-3900.

1.2.5.1 A List of the Defined Subsystems

Use the following command to get a list of the defined subsystems:

```
# syspar_ctr l -E
  hats  /usr/lpp/ssp/bin/hatsctrl
  hb  /usr/lpp/ssp/bin/hbctrl
  hags  /usr/lpp/ssp/bin/hagsctrl
  haem  /usr/lpp/ssp/bin/haemctrl
  hr  /usr/lpp/ssp/bin/hrctrl
  pman  /usr/lpp/ssp/bin/pmanctrl
  emon  /usr/lpp/ssp/bin/emonctrl
  sp_configd  /usr/lpp/ssp/bin/sp_configdctrl
  emcond  /usr/lpp/ssp/bin/emconditionctrl
```

The order in which the subsystems are listed, from top to bottom, is the order in which they are added or started by the syspar_ctrl command. This command deletes or stops the subsystems in the reverse order, from bottom to top. Normally, you are not supposed to do anything to manage the system partition-sensitive subsystems. In general, they are all added and started during installation and use the standard mechanisms to restart themselves when they are not explicitly stopped.

Note

Do not forget that the syspar_ctrl command can be used only in a PSSP 2.2 environment.

1.2.5.2 More Information about a Subsystem

To get more information:

```
# lssrc -ls hats.sp2en2
Subsystem         Group            PID     Status
 hats.sp2en2      hats             29898   active
  NODES_DEFINED = 5, IN_GROUP = 5, GROUP STATUS = Stable
  Myself = (192.168.3.37, 1122670240), GL = (192.168.3.37, 1124129351)

  HB Interval = 1 secs HB Sensitivity = 4 missed beats
  CWS = 192.168.3.37
```

1.2.5.3 The path, stdin, stdout, stderr Flags Set for a Subsystem

To look at the flags:

```
# lssrc -S -s hagsglsm
#subsysname:synonym:cmdargs:path:uid:auditid:standin:standout:standerr:
action:multi:contact:svrkey:svrmtype:priority:signorm:sigforce:display:
waittime:grpname:
hagsglsm:::/usr/lpp/ssp/bin/hagsglsmd:0:0:/dev/console:/var/ha/log/glsm.d
efault.sp2en3:/var/ha/log/glsm.default.sp2en3:-R:-Q:-K:0:0:20:0:0:-d:30:
hags:
```

1.2.5.4 GS Domain Information

To get information about the GS domain:

```
# lssrc -ls hags.sp2en2
Subsystem         Group            PID     Status
 hags.sp2en2      hags             27502   active
 2 locally-connected clients.  Their PIDs:
 28876 20134
 HA Group Services domain information:
 Domain established by node 0.
 Number of groups known locally: 2
                   Number of   Number of local
 Group name        providers   providers/subscribers
 cssMembership           2           0          1
 ha_em_peers             5           1          0
```

```
# hagsns -s hags.sp2en2
We are: 0.6 domaindId = 0.6 noNS = 0 inRecovery = 0
NS::ENsState(7):kBecomeNS protocolInProgress = NS::ENsProtocol(0):kNoProtocol
outstandingBroadcast = NS::ENsBroadcast(0):kNoBcast
Process started on Jan 25 18:25:23, (29d 6:27:33) ago.  HB connection
took (0:1:3).
Initial NS certainty on Jan 25 18:26:46, (29d 6:26:10) ago, taking (0:0:20).
Our current epoch of certainty started on Jan 25 18:26:46, (29d 6:26:10) ago.
5 UP nodes: 0 11 12 15 16
1.1 ha_em_peers: GL: 0 seqNum: 0 theIPS: 0 11 12 16 15 lookupQ:
2.1 cssMembership: GL: 0 seqNum: 6 theIPS: 12 0 15 lookupQ:
```

1.2.5.5 List of Nodes in Each MetaGroup

For a list of the nodes in each MetaGroup:

```
# hagsmg -s hags.sp2en2
2.1 cssMembership: 0 12 15
1.1 ha_em_peers: 0 11 12 16 15
0.Nil ZtheNameServerXY: 0.6 11.0 12.2 16.22 15.4
0.Nil  theGROVELgroup:
```

1.2.5.6 Status of the Broadcast Services

For the status of the broadcast services:

```
# hagspbs -s hags.sp2en2
2.1 cssMembership: HWM 51 LWM 51 weAreTheGL
  pendingAckCount=0 kNotExpectingAcks  pendingRecoverCount=0
       0: HWM=1: lastType1=Nil
       12: HWM=51: lastType1=Nil
       15: HWM=51: lastType1=Nil
1.1 ha_em_peers: HWM 130 LWM 130 weAreTheGL
  pendingAckCount=0 kNotExpectingAcks  pendingRecoverCount=0
        0: HWM=0: lastType1=Nil
        11: HWM=130: lastType1=129
        12: HWM=130: lastType1=129
        16: HWM=130: lastType1=129
        15: HWM=130: lastType1=129
0.Nil ZtheNameServerXY: HWM 52 LWM 52 weAreTheGL
  pendingAckCount=0 kNotExpectingAcks  pendingRecoverCount=0
        0.6 kUp: HWM=0: lastType1=Nil
        11.0 kUp: HWM=52: lastType1=Nil
        12.2 kUp: HWM=52: lastType1=Nil
        16.22 kUp: HWM=52: lastType1=Nil
        15.4 kUp: HWM=52: lastType1=Nil
0.Nil  theGROVELgroup: HWM Nil LWM Nil
  pendingAckCount=0 kNotExpectingAcks  pendingRecoverCount=0
```

1.2.5.7 Client Processes Connected to GS

To see the client processes connected to GS:

```
# hagscl -ls hags.sp2en2
Client Control layer summary:
  Number of clients connected: 2
  Cumulative number of clients connected: 2
  Total number of client requests: 2
  Number of client hash table conflicts: 0

--------------------------------------------------
Client: socketFd[8] pid[28876]Total number of Clients: 2
 Client initialized: pid: 28876
Number of local providers/subscribers: 1/3
Responsiveness information for Client: socketFd[8] pid[28876]
Type[ type[HA_GS_PING_RESPONSIVENESS]] interval[120] response time limit[120]
Checks done/bypassed[20904/0] lastResponse[OK]]
Results(good/bad/late)[20904/0/0]
Membership list:
slot    info
0       [{provider}Member token[0] Client: socketFd[8] pid[28876]ProviderId[1/0]]
1       [{subscriber}Member token[1] Client: socketFd[8] pid[28876]]
2       [{subscriber}Member token[2] Client: socketFd[8] pid[28876]]
3       [{subscriber}Member token[3] Client: socketFd[8] pid[28876]]
--------------------------------------------------
Client: socketFd[9] pid[20134]Total number of Clients: 2
 Client initialized: pid: 20134
Number of local providers/subscribers: 1/1
Responsiveness information for Client: socketFd[9] pid[20134]
Type[ type[HA_GS_PING_RESPONSIVENESS]] interval[3630] response time limit[10]
Checks done/bypassed[696/0] lastResponse[OK]]
Results(good/bad/late)[0/0/0]
Membership list
slot    info
```

1.2.5.8 List of Providers and Subscribers of Each Group

To see the providers and subscribers of every group:

```
# hagsgr -s hags.sp2en2
  Number of: groups: 5
Group slot # [0] Group name[HostMembership] group state[Idle |]
Providers[[0/0][0/11][0/12][0/16][0/15]]
Local subscribers[[8/0]]

Group slot # [1] Group name[ha_em_peers] group state[Inserted |Idle |]
Providers[[1/0][1/11][1/12][1/16][1/15]]
Local subscribers[]

Group slot # [2] Group name[!SwitchGroupie!] group state[Idle |]
Providers[]
Local subscribers[]

Group slot # [3] Group name[enMembership] group state[Idle |]
Providers[[0/0][0/11][0/12][0/15][0/16]]
Local subscribers[[8/0]]

Group slot # [4] Group name[cssMembership] group state[Inserted |Idle |]
Providers[[0/12][0/15]]
Local subscribers[[8/0]]
```

1.2.5.9 List of Providers and Subscribers for a Specific Group

To see the providers and subscribers for a specific group:

```
# hagsgr -a ha_em_peers -s hags
  Number of: groups: 5
Group name[ha_em_peers] group state[Inserted |Idle |]
Providers[[1/0][1/11][1/12][1/16][1/15]]
Local subscribers[]
```

1.2.6 Relationship between the HA Components

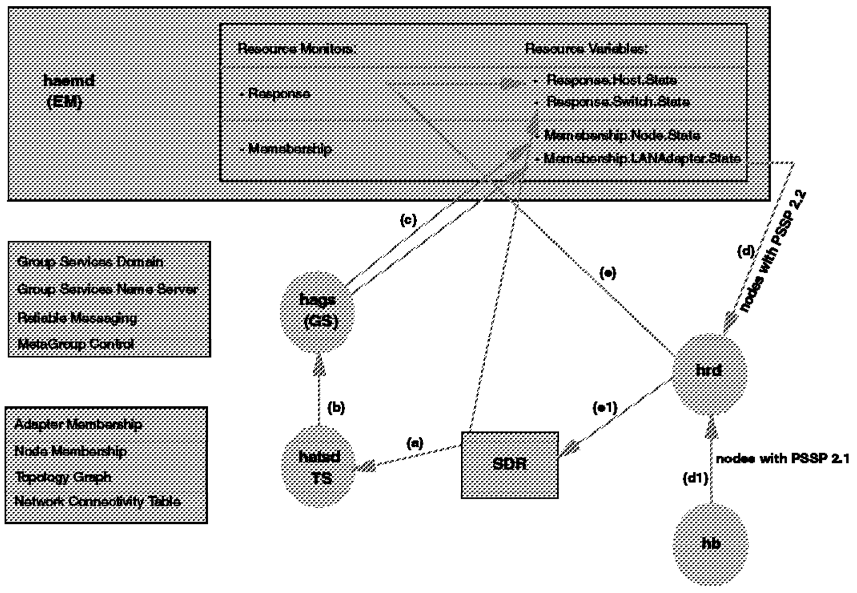


Figure 4. Relationship Between HA Components

Figure 4 shows the relationship between the components of the HA subsystems. The Topology Services subsystem gets the required data from the SDR and prepares the information that should be passed to Group Services. This information contains the data about Adapter Membership, Node Membership, the Topology Graph (a table including the node numbers and the network definitions on the nodes), and the Network Connectivity Table (containing the description of the routes in the system).

The GS subsystem maintains this information and extends it with data for its own use, and with data that will be used by other subsystems. The data needed for GS itself is information about the GS domain, main groups and their members, the GS Nameserver, MetaGroup control and Reliable Messaging. The data needed by other subsystems, like Event Management, consists, for example, of the resource variables of the Membership resource monitor:

- Membership.Node.State
- Membership.LANAdapter.State

These resource variables are passed to the Event Management subsystem. The host responds daemon takes the information about Membership.LANAdapter.State variable from the Event Management subsystem. Then it updates the Response.Host.State resource variable of the Response resource monitor. At the same time, the host responds daemon updates the host_responds class in the SDR. (For nodes running PSSP 1.2 or PSSP 2.1, the host responds daemon takes the information that should be passed to the Response.Host.State resource variable not from the Membership.LANAdapter.State resource variable, but from the heartbeat daemon.)

Note

The hr and hb daemons run on the Control Workstation for coexistence with PSSP 1.2 and PSSP 2.1, and the hb daemon runs on the nodes with these PSSP levels. On the nodes with PSSP 2.2, the HA daemons that are running are: hats, hags, hagsglsm, haem, pman, pmanrm, sp_configd, and Emonitor.

The flexibility of the subsystems offers the ability to tailor their usage to the best way of monitoring and recovering the corresponding parts of the system. A client subsystem can be informed about events affecting its own resources, or about general system conditions that require attention by the subsystem. When there is no need to coordinate the distributed actions, but the system needs to be informed of events that may occur on other nodes, the subsystem is able to take advantage of Event Manager. When the subsystem needs to respond to various conditions in a coordinated way, then the GS subsystem becomes very important. The support for the synchronized multiphase protocol execution enables the subsystem to ensure that all its members are responding to the condition in a proper way.

Likewise, a subsystem may interface with GS when it does not need the powerful event monitoring capability of Event Management, but simply needs coordination of its actions.

1.3 Problem Determination on the New High Availability Infrastructure

The first steps of the recovery procedure when having problems with the new HA daemons are:

- Check stdout and stderr for this daemon:

```
# lssrc -S -s hags
#subsysname:synonym:cmdargs:path:uid:auditid:standin:standout:standerr:
action:multi:contact:svrkey:svrmtype:priority:signorm:sigforce:display:
waittime:grpname:
hags:::/usr/lpp/ssp/bin/hagsd:0:0:/dev/console:/var/ha/log/gs.default.sp2
en3:/var/ha/log/gs.default.sp2en3:-R:-Q:-K:0:0:20:0:0:-d:30:hags:
```

- Check if there is a log file for this daemon:

```
# ls -lt /var/ha/log|grep <syspar_name>
.....
-rw-r--r--   1 root     system    769229 Feb 15 13:12 hags.sp2en3_0_12.sp2en3
.....
-rw-r--r--   1 root     system      4841 Feb 15 13:03 gs.default.sp2en3
.....
```

- Look at the log file:

```
# cat /var/ha/log/hags.sp2en3_0_12.sp2en3
.....
[TOD(Feb 15 12:39:37)]( TRACE_FYI ) in SSuppSocket::ReadData(void *p, int
readlen: Received signal [SIGTERM (15)]
[TOD(Feb 15 12:39:37)]( TRACE_FYI ) in SRCSocket::StopNormal(void): STOP
NORMAL request from SRC, stop_requested was.
[TOD(Feb 15 12:39:37)]( TRACE_FYI ) in DispatchControl::Dispatcher(void):
hagsreap process finished.  Removed [0] files.
```

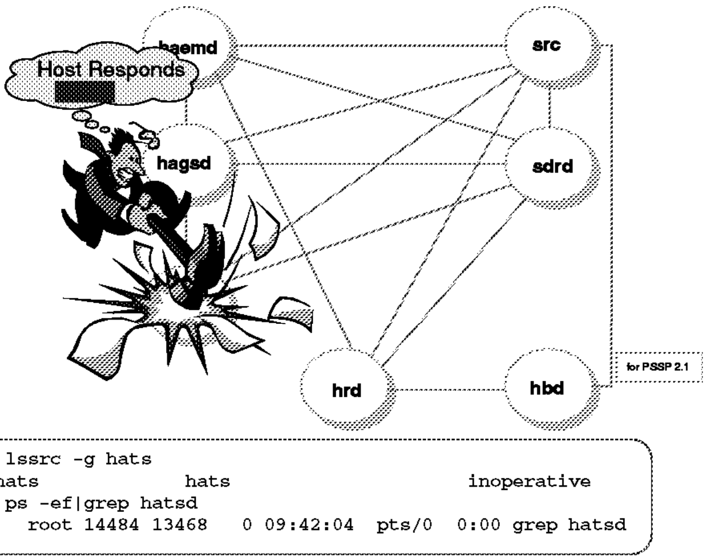
We see that hags has been stopped at the normal request of SRC.

Have a look at the log files on the Control Workstation:

```
#cat gs.default.sp2en3
hb_init:  hb_init_communication failed
Get_Instance(void):  Our Incarnation Number is 13
[TOD(Feb 15 18:27:47)]( TRACE_FYI ) in Prepare_SP_Node(NodeNum   *node_num,:
We were LNN in [12] incarnation (size [3]).  Force node ′failure′.
[TOD(Feb 15 18:27:47)]( TRACE_FYI ) in Prepare_SP_Node(NodeNum *node_num,:
We will stop hats for [45] seconds.
[TOD(Feb 15 18:27:47)]( TRACE_FYI ) in Prepare_SP_Node(NodeNum   *node_num,:
Message starting hats: [0513-059 The hats.sp2en3 Subsystem has been started.
Subsystem PID is 46842.
[TOD(Feb 15 18:27:47)]( TRACE_FYI ) in Prepare_SP_Node(NodeNum *node_num,:
Hats restarted.  Let′s continue.
.....
[TOD(Feb 15 18:27:47)]( TRACE_FYI ) in main:  Our NodeId is 0.13
.....
```

We see the next incarnation of hagsd that has been started on this system (0.13 means the 13th incarnation of hagsd on the Control Workstation).

1.3.1.1 What Happens If hatsd Dies?



```
# lssrc -g hats
 hats            hats                        inoperative
# ps -ef|grep hatsd
    root 14484 13468   0 09:42:04  pts/0  0:00 grep hatsd
```

Figure 5. What Happens If hatsd Dies?

We know that hagsd died when:

1. The hats subsystem is inoperative and the hatsd process is not running:

```
# lssrc -g hats
 hats             hats                        inoperative
# ps -ef|grep hatsd
    root 14484 13468   0 09:42:04  pts/0  0:00 grep hatsd
```

2. The output of `SDRGetObjects host_responds` shows 0 for the node where hatsd died:

```
# SDRGetObjects host_responds
node_number  host_responds
          1           1
          2           1
          5           0
          6           1
```

3. The spmon GUI for host_responds turns to red for this node.

What is the behavior of the system?

The death of hatsd may have many causes. It could happen because of a "software collision," but also because of defective hardware. The host_response value will be 1 as soon as one of the following connections works: Ethernet or Switch. That means host_responds refers to the heartbeat:

```
# lssrc -s hats
Subsystem         Group            PID     Status
 hats             hats             17896   active
```

If hatsd dies on the node ("dies" means it was not stopped using the `stopsrc -s hats` or `hatsctrl -k` commands), SRC tries to restart the hats subsystem.

If you enter the same command after the daemon has died, and SRC succeeded in restarting hats, you will see that hats is still active but has another PID, as follows:

```
# lssrc -s hats
Subsystem         Group            PID     Status
 hats             hats             12682   active
```

Let us look at the log files. The /var/ha/log/hats.17.190733.sp2en3 file on the node shows:

```
02/17  19:07:34 hatsd[0]: High Availability Topology Services Daemon.
02/17  19:07:34 hatsd[0]: Using socket Port number = 10006.
02/17  19:07:34 hatsd[0]: Using server socket = /var/ha/soc/hats/server_socket.sp2en3.
02/17  19:07:34 hatsd[0]: System Configuration File: /var/ha/run/hats.sp2en3/machines.lst.
02/17  19:07:34 hatsd[0]: Node 0 adapter 0 Name 192.168.3.37 Address 192.168.3.37.
02/17  19:07:34 hatsd[0]: Node 5 adapter 0 Name 192.168.3.5 Address 192.168.3.5.
02/17  19:07:34 hatsd[0]: Node 6 adapter 0 Name 192.168.3.6 Address 192.168.3.6.
02/17  19:07:34 hatsd[1]: Node 5 adapter 1 Name 192.168.13.5 Address 192.168.13.5.
02/17  19:07:34 hatsd[1]: Node 6 adapter 1 Name 192.168.13.6 Address 192.168.13.6.
02/17  19:07:34 hatsd[0]: Adapter[0] Name = 192.168.3.6, Address = 192.168.3.6.
02/17  19:07:34 hatsd[1]: Adapter[1] Name = 192.168.13.6, Address = 192.168.13.6.
02/17  19:07:35 hatsd[0]: Type: en0 Address: 192.168.3.6, BROADCAST.
.....
        My Leader is              (192.168.3.6:0x4308f2c7).
        My Crown Prince is        (192.168.3.6:0x4308f2c7).
        My upstream neighbor is   (192.168.3.6:0x4308f2c7).
        My downstream neighbor is (192.168.3.6:0x4308f2c7).
.....
02/17  19:07:36 hatsd[0]: Received a Group Proclaim from (192.168.3.6:0x4308f2c7)
in group (192.168.3.6:0x4308f2c7).
02/17  19:07:36 hatsd[0]: Received a HB_GET_CONFIG_INFO request client PID = 17042.
02/17  19:07:36 hatsd[0]: Received a HB_SUBSCRIBE request from client PID = 17042.
02/17  19:07:36 hatsd[0]: Subscription type [Hb_All_Nodes_Subscription] for event [Hb_All_Events].
02/17  19:07:36 hatsd[0]: New Interest entry created at 0x20089c48.
02/17  19:07:41 hatsd[0]: New Interest entry created at 0x20089cc8.
02/17  19:07:42 hatsd[0]: Received a Group Proclaim from (192.168.3.37:0x430646a7)
in group (192.168.3.37:0x4308f2c9).
02/17  19:07:42 hatsd[0]: Sending JOIN request to (192.168.3.37:0x430646a7)
in group (192.168.3.37:0x4308f2c9).
02/17  19:07:44 hatsd[0]: Received a PTC request from (192.168.3.37:0x430646a7)
in group (192.168.3.37:0x4308f2ce).
02/17  19:07:44 hatsd[0]: Received a COMMIT request from (192.168.3.37:0x430646a7)
in group (192.168.3.37:0x4308f2ce).
02/17  19:07:44 hatsd[0]: Sending a COMMIT ACK response to (192.168.3.37:0x430646a7).
02/17  19:07:44 hatsd[0]: Sending adapter [Hb_Join] notifications.
02/17  19:07:44 hatsd[0]: Sending adapter [Hb_New_Group] notifications.
02/17  19:07:44 hatsd[0]: My New Group ID = (192.168.3.37:0x4308f2ce) and is Stable.
        My Leader is              (192.168.3.37:0x430646a7).
        My Crown Prince is        (192.168.3.6:0x4308f2c7).
        My upstream neighbor is   (192.168.3.37:0x430646a7).
        My downstream neighbor is (192.168.3.37:0x430646a7).
02/17  19:07:45 hatsd[0]: Sending node [Hb_Join] notifications.
02/17  19:07:45 hatsd[0]: Sending Notification packet for [hagsd_pm_client] subscription.
02/17  19:07:45 hatsd[0]: Sending node [Hb_New_Group] notifications.
02/17  19:07:45 hatsd[0]: Sending Notification packet for [hagsd_pm_client] subscription.
02/17  19:07:46 hatsd[0]: Received a HB_GET_CONFIG_INFO request from client PID = 13980.
```

This log file was created by hatsd. In the first phase, it lists all data such as:

- The socket port number (10006)
- The server socket name (/var/ha/soc/hats/server_socket.sp2en3)
- The system configuration file (/var/ha/run/hats.sp2en3/machines.lst)
- The IP addresses of the Ethernet and Switch adapters

  This data was taken from the machines.lst file and contains Ethernet and Switch IP addresses for all nodes that belong to this partition and run PSSP 2.2.

  We can also see that there are three systems that can belong to the group: the Control Workstation, Node 5, and Node 6.

Remote daemon-to-daemon communication uses UDP port numbers. They are stored in the SDR in the class Syspar_ports:

```
# SDRGetObjects Syspar_ports
subsystem    port
hats              10006
hags              10007
haem              10008
```

In the second phase, the node created a *singleton* group, which means a group that consists of only one member, the node itself, and this node is the GS nameserver, Crown Prince, and its own neighbor. Afterwards this node received requests such as:

- `HB_GET_CONFIG_INFO`
- `HB_SUBSCRIBE`

Both came from a process with PID = 17042. This process was the GS daemon, hagsd, running on this node.

In the third and last phase, the node sent a join request to the GS nameserver (in this case the Control Workstation), and after getting a commitment and sending an acknowledge response to the GS nameserver, Node 6 became a member of the group again.

Now if you follow the corresponding log file on the Control Workstation, for example /var/ha/log/hats.15.182838.sp2en3, you see the same procedure from the point of view of the Group Leader. You also see that Node 5 is dead, and the group that was built contains only the Control Workstation and Node 6:

```
.....
02/17  19:07:42 hatsd[0]: Received a JOIN request from (192.168.3.6:0x4308f2c7).
02/17  19:07:42 hatsd[0]: This is the first JOIN request!
02/17  19:07:42 hatsd[0]: Base group:
       0 (192.168.3.37:0x430646a7)
02/17  19:07:42 hatsd[0]: Other group:
       0 (192.168.3.6:0x4308f2c7)
02/17  19:07:42 hatsd[0]: Merged group:
       0 (192.168.3.37:0x430646a7)     1 (192.168.3.6:0x4308f2c7)
02/17  19:07:44 hatsd[0]: Sending PTC to new membership:
       0 (192.168.3.37:0x430646a7)     1 (192.168.3.6:0x4308f2c7)
02/17  19:07:44 hatsd[0]: PTC's sent = 2, PTC's received = 1.
02/17  19:07:44 hatsd[0]: Received the final PTC response, committing group.
02/17  19:07:44 hatsd[0]: Deleting Current Group, (192.168.3.37:0x4308f2c9),
from the NCT.
02/17  19:07:44 hatsd[0]: Received a COMMIT BROADCAST request from
(192.168.3.37:0x430646a7) in group (192.168.3.37:0x4308f2ce).
02/17  19:07:44 hatsd[0]: Other group:
       0 (192.168.3.37:0x430646a7)     1 (192.168.3.6:0x4308f2c7)
.....
       My Leader is             (192.168.3.37:0x430646a7).
       My Crown Prince is       (192.168.3.6:0x4308f2c7).
       My upstream neighbor is  (192.168.3.6:0x4308f2c7).
       My downstream neighbor is (192.168.3.6:0x4308f2c7).
02/17  19:07:45 hatsd[0]: Sending node [Hb_Join] notifications.
02/17  19:07:45 hatsd[0]: Sending Notification packet for [hagsd_pm_client] subscription.
02/17  19:07:45 hatsd[0]: Sending node [Hb_New_Group] notifications.
02/17  19:07:45 hatsd[0]: Sending Notification packet for [hagsd_pm_client] subscription.
```

This example shows the behavior of the system in case the hats subsystem can be restarted.

But what happens if hatsd cannot run? **What is the consequence to the system if hatsd is missing?**

The consequence of missing hatsd on a node is that the node does not pass any *heartbeats* or *pulses* any more. If the number of missing heartbeats reaches some limit (threshold), the adapter is considered dead, and a message about its death is sent to the Group Leader and all members of the Adapter Membership group by its neighbor who is the one expecting a heartbeat packet every second by default.

The Group Leader checks whether the sender of this message is a valid member of the group. If this verification fails, the message is ignored. Otherwise, the Adapter Membership group is recreated and the dead adapter does not belong to the group any more. On the other side, each node does the same verification about the sender of the message and if the verification passes, each node changes its state to a singleton Adapter Membership group, declaring itself to be the Adapter Membership Group Leader and neighbor at the same time and having the status heartbeat_group_unstable. Then the Adapter Membership Group Leader changes the status of the nodes running hatsd to heartbeat_group_stable and recommits the group with its new status. The same commits are done for options such as node_connectivity and group_connectivity.

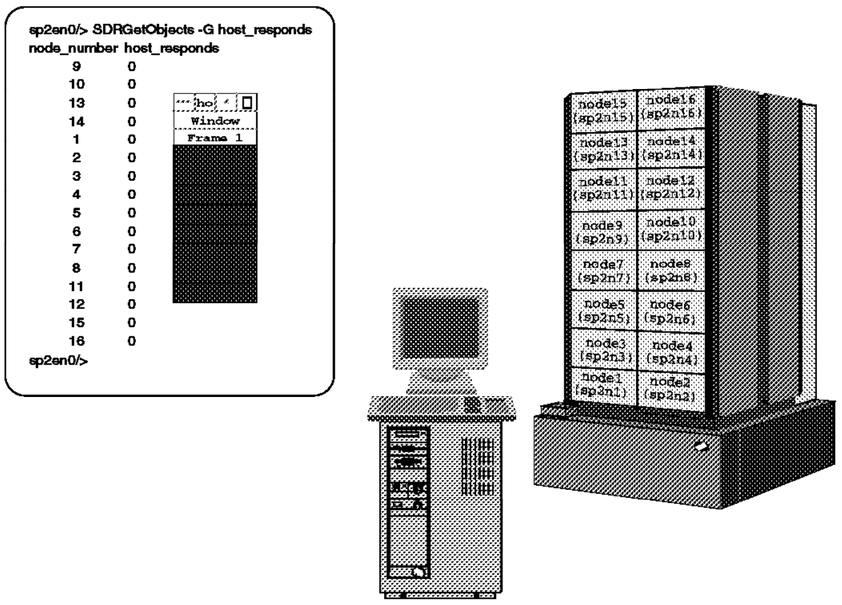
Following the maintained information about current node connectivity in an SP system, the node with the dead adapter is considered to be down and does not participate in a list of reachable nodes, which is the basis for the view of the topology of the entire system.

Most of the problems with HA subsystems are related to host_responds and switch_responds (for switch problem determination, see Chapter 5, "Diagnosing Switch and Partitioning Problems" on page 213). Host_responds problems result mostly from the fact that for some reason, hatsd is not running. But sometimes, on checking the system, all required HA daemons seem to be running and host_responds for the node(s) is still 0. Let us have a look at a similar situation on the system.

Symptoms:
host_responds does not work.
SDRGetObjects host_responds is 0 for all nodes.

Environment:
AIX 4.1.4, the Control Workstation, and the nodes migrated to PSSP 2.2.

Problem Description:



```
sp2en0/> SDRGetObjects -G host_responds
node_number host_responds
        9       0
       10       0
       13       0
       14       0
        1       0
        2       0
        3       0
        4       0
        5       0
        6       0
        7       0
        8       0
       11       0
       12       0
       15       0
       16       0
sp2en0/>
```

Figure 6. Problems with host_responds

After migration of the Control Workstation and the nodes to PSSP 2.2, the values of the host_responds field in the host_responds class remain 0 for all nodes. During the migration, problems occurred doing Step 10 of the installation guide and executing `syspar_ctlr -r -G`. This command did not terminate, and after trying to interrupt it, the Control Workstation died and had to be rebooted.

Problem Determination:

The first step was to stop and restart the partition-sensitive subsystems on the Control Workstation, using the commands `syspar_ctrl -k` and `syspar_ctrl -s`, and if these terminated successfully, to have a closer look at the hats subsystem using the command `lssrc -l -s hats.<syspar_name>`. These actions were completed successfully.

The output of the `lssrc -l -s hats.sp2en2` command issued on the Control Workstation was:

```
# lssrc -l -s hats.sp2en2
Subsystem         Group            PID     Status
 hats.sp2en2      hats             29898   active
  NODES_DEFINED = 5, IN_GROUP = 4, GROUP STATUS = Stable
  Myself = (192.168.3.37, 1122670240), GL = (192.168.3.37, 1124203422)
  HB Interval = 1 secs HB Sensitivity = 4 missed beats
  CWS = 192.168.3.37
```

The command `lssrc -l -s hats` on the node hangs.

The /var/ha/log/hats.sp2en2 log file on the critical node contains entries such as:

```
/usr/lpp/ssp/rcmd/bin/rsh: 0041-002 Host sp2en0.msc.itso.ibm.com
is not registered for Kerberos service.
trying normal rsh (/usr/bin/rsh)
rshd: 0826-813 Permission is denied.
```

So it seems to be a Kerberos problem. Normally, there should be a ticket of "never" on each node. The hats daemon takes the ticket and does an `rcmdtgt` to make it "never." In our case, hatsd seems to use the .rhosts file instead of the recommended Kerberos ticket. The next step in determining the problem was to check all Kerberos files. They were all OK.

What is the next area to look at? Let us check the network configuration! After all tests with commands like `ping` and `netstat -ni`, and, last but not least, verifying the hostname resolution, it was no longer hard to identify the problem: a "mixed environment" - using /etc/hosts on the Control Workstation, and the domain nameserver /etc/resolv.conf on the nodes.

Conclusion:

Figure 7 on page 32 shows the result of our problem determination in this case.

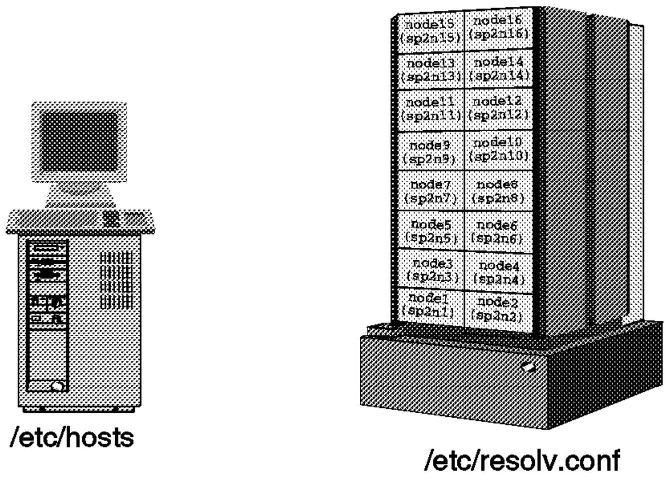


Figure 7. Result of Missing hatsd Problem Determination

As mentioned before, the Control Workstation used the local /etc/hosts and the nodes were using the /etc/resolv.conf file. The hatsd daemon had problems with getting a ticket because the nodes could not rsh back to the Control Workstation using the Kerberos ticket. There are only two possibilities: you can either use /etc/resolv.conf on the Control Workstation *and* the nodes, or use /etc/hosts on the Control Workstation *and* the nodes (take a look at the script.cust script).

Solution: In case /etc/hosts is used:

1. Remove /etc/resolv.conf from the nodes.
2. Check /etc/hosts on the nodes.
3. Restart hats on each node.
4. Restart the hr daemon on the Control Workstation.

The relationship between the host response and the time synchronization on the SP system.

Does the lack of synchronization of the system clocks on the SP system affect the host response? This question is the starting point for our problem determination of this case.

What happens if the time of the node and its time server are not synchronized? Let us say that the time on the node has been changed. What is the consequence for the system?

First, the host response is still working, but Kerberos, of course, gets into trouble, because it can only handle a 5-minute difference, and an rsh between the Control Workstation and the node is no longer possible. As host response refers to the heartbeat, the value of the host_responds SDR class for this node is still 1. Although hatsd registers the change of time, the state of the connection remains established. In the hatsd log file /var/ha/log/hats... you can see some information about being late in sending a heartbeat, but this does not affect the host response.

Let us go back to the situation with the host response and the wrong time on the node. If we reboot the node now, we lose the host response at the moment when the communication adapter is initialized. The header of the packets that are being sent and received while establishing the connection between the communication adapters, contain all the information necessary to deliver a packet. Part of this information is a time stamp. During the attempts to establish the connection, the time stamps are compared, and in our case the adapter of the node having a wrong time stamp is considered “down” and removed from the Adapter MembershipList. This information is sent to all nodes and is visible in the log files.

Now let us imagine that we do not reboot the node and our status is still: host response is OK and the time is not synchronized. Let us stop the hats daemon and restart it instead of rebooting the system.

The result is the same as before, because as soon as hatsd is stopped, you lose the host response, and as soon as you start hatsd again, the attempt to establish the connection between the adapters is made, and at this moment the time stamps must match. The behavior of the system confirms the fact that the time stamps are compared by establishing the communication between the adapters, and once the communication is established, the mismatch of the time stamps is registered but does not affect the communication. This explains the fact that after changing the time on the node, the host response was still OK.

So whether you reboot the node or stop/restart the daemons, the effect is the same: the node loses its host response. The solution is to set up the right time-of-day on the node.

Before we finish this case, let us have a look at the behavior of the hats and hags daemons after restarting hatsd (or rebooting of the system). If you check the existence of the daemons, you can see that they are active again. But let us have a closer look at their activities. hatsd is running, but the node is in singleton mode. The node without host_responds left the Adapter Membership group and created a singleton group (this node is the only member of the group). It will remain in the singleton group as long as the connection to its communication adapters cannot be established. The log files of hagsd show that the group has changed and the node without the host response does not belong to this group any more.

Let us now come back to the initial question of this unit: does the lack of synchronization of the system clocks affect the host response? The answer should be: not as long as hatsd is running.

1.3.1.2 What Happens If hagsd Dies?

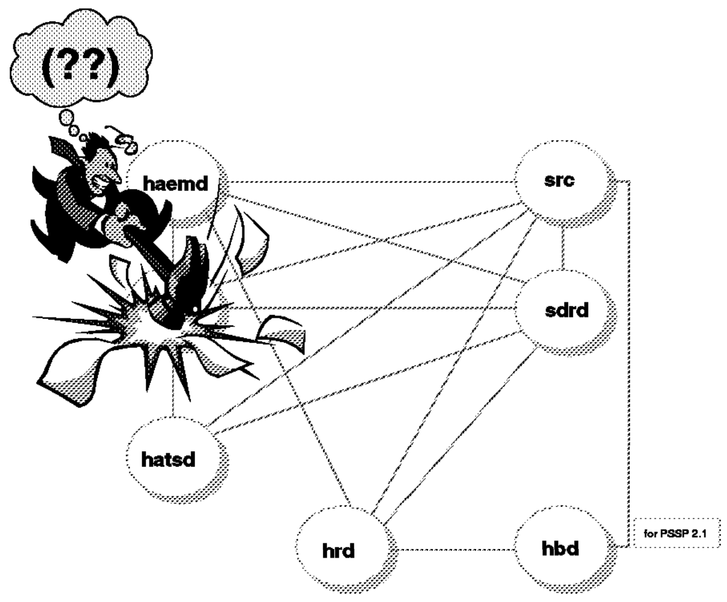


Figure 8. What Happens If hagsd Dies?

How do we recognize that hagsd died?

Similar to the hats and haem subsystems, hags is created as an SRC subsystem using the flags -R, -Q and -K. You can see that these flags are set for hags with the command:

```
# lssrc -S -s hags
#subsysname:synonym:cmdargs:path:uid:auditid:standin:standout:standerr:
action:multi:contact:svrkey:svrmtype:priority:signorm:sigforce:display:
waittime:grpname:
hags:::/usr/lpp/ssp/bin/hagsd:0:0:/dev/console:/var/ha/log/gs.default.sp2
en3:/var/ha/log/gs.default.sp2en3:-R:-Q:-K:0:0:20:0:0:-d:30:hags:
```

This output gives you global information about hags, and in the last line you can see that the flags -R, -Q, -K are set for this daemon. Their meaning is as follows:

-R	The subsystem is restarted every time after it terminated abnormally.
-K	The subsystem uses sockets as its communication method.
-Q	Multiple instances of the subsystems are not allowed to run at the same time, and the subsystem is not to share the same Interprocess Communication (IPC) queue.

So if hags terminates abnormally, the SRC tries to restart it again. If this succeeds, you may not notice that hags died, but the daemon process will have another PID and you can see the incarnation number of the daemon (/var/ha/lck/hags.tid.<syspar_name>):

```
# ls -la /var/ha/lck/hags.tid.sp2en3 (on the node5)
total 3
drwxr-xr-x   2 root     system       512 Feb 18 14:41 .
drwxr-xr-x   8 root     system      1024 Feb 15 18:28 ..
----------   1 root     system         0 Feb 18 14:41 hagsd.in32
```

Now let us see whether the incarnation number is correct:

```
# hagsmg -s hags  (on the node5)
2.1 cssMembership: 5 6 0
1.1 ha_em_peers: 0 6 5
0.Nil ZtheNameServerXY: 0.Nil 6.Nil 5.32
0.Nil  theGROVELgroup:
```

The line describing the MetaGroup,
`0.Nil ZtheNameServerXY: 0.Nil 6.Nil 5.32`,
shows us that hagsd on Node 5 was started 32 times.

If the incarnation number of hags is large, as in our case, there must be something wrong with the environment. For more information, you can look at the corresponding log files. There are at least two you should look at on the node (there are also some on the Control Workstation):

- /var/ha/log/gs.default.<syspar_name>
- /var/ha/log/hags_<nodenumber>_<incarnation_number>.<syspar_name>

Let us have a look at the first one, /var/ha/log/gs.default.sp2en3:

```
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in Get_Instance(void):  Our Incarnation Number is 32
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in Prepare_SP_Node(NodeNum   *node_num,:
No mark file[/var/ha/lck/hags.status.sp2en3] previous incarnation [31]
nothing special.
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in main:  Our NodeId is 5.32
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in Establish_Resource_Limits(void):
Successfully modified priority from [61] to [38] for pid [19186]
.....
```

You can see that the new incarnation of hagsd was started on Node 5 (5.32), and the new hagsd process running with PID 19186 became a higher priority (38).

The second log file, /var/ha/log/hags_5_32.sp2en3, shows:

```
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in NS::NS_Status: We are: 5.32 domaindId = 0.Nil noNS = 1 inRecovery
= 0
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in NS::NS_Status: NS::ENsState(0)::kUncertain
protocolInProgress
= NS::ENsProtocol(0):kNoProtocol outstandingBroadcast = NS::ENsBroadcast(0):kNoBcast
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in NS::NS_Status: Process started on Feb 18 14:41:20, (0:0:0)
ago.
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in NS::NS_Status: 0 UP nodes:
.....
( TRACE_FSM ) in NS::ChangeState: Leaving  nsState NS::ENsState(0):kUncertain
[TOD(Feb 18 14:41:20)]Recursive call to NS::ChangeState
( TRACE_FSM ) in NS::ChangeState: Entering nsState NS::ENsState(2):kAscend
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in NS::Init: NS::Init complete.  We are 5.32
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in NS::Init: HB connect complete.hagsd process started
on Feb
18 14:41:20.  HB connect took (0:0:0)
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in SSuppConnSocket::SSuppConnSocket(int sock_fd,: Opened
supplicant
connection socket, path[/var/ha/soc/hagsdsocket.sp2en3]
.....
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in MGCommitInsertP1:  0.Nil ZtheNameServerXY: 0.Nil 6.Nil 5.32
[TOD(Feb 18 14:41:20)]Recursive call to NS::ChangeState
( TRACE_FSM ) in NS::ChangeState: Entering nsState NS::ENsState(6):kCertain
.....
[TOD(Feb 18 14:41:22)]( TRACE_FYI ) in MGApplyInsertP1ToGroup:  1.1 ha_em_peers. Old: 0 6.
New: 5
[TOD(Feb 18 14:41:22)]( TRACE_FYI ) in PBLazyAck::main: Sent node 0 an ACK in Container at
0x30051778
 (D=0.13 G=1.1 HWM=277 LWM=275 ACK)
[TOD(Feb 18 14:41:22)]( TRACE_FYI ) in pbs_inhale_input_message: Node 0 sent us Container at 0x3005748
(D=0.13 G=1.1 HWM=278 LWM=277 Type-3 MG-BCAST MGMsgType=2)
[TOD(Feb 18 14:41:22)]( TRACE_FYI ) in MGCommitInsertP1:  1.1 ha_em_peers 0 6 5
[TOD(Feb 18 14:41:22)]( TRACE_FYI ) in SGroup::MetaGroupInsertDone(NodeNum _node): Group
name[ha_em_peers]
Node number [0] inserted into meta-group.
[TOD(Feb 18 14:41:22)]( TRACE_FYI ) in SGroup::MetaGroupInsertDone(NodeNum _node): Group
name[ha_em_peers]
Node number [6] inserted into meta-group.
.....
[TOD(Feb 18 14:41:22)]( TRACE_FYI ) in SDaemonMessage::HandleFirstPhaseBroadcast(Boolean
_nPhase):
Protocol first-phase (of N) bcast received.
[TOD(Feb 18 14:41:22)]( TRACE_FYI ) in SDaemonMessage::UpdateProtocol(Boolean
_createProtocol,:
GL told us to create a join protocol!
[TOD(Feb 18 14:41:22)]( TRACE_FYI ) in SDaemonMessage::UpdateProtocolMembership(char *
pMsgPtr):
Grabbed attributes from protocol broadcast.
[TOD(Feb 18 14:41:22)]( TRACE_FYI ) in SDaemonMessage::UpdateProtocolMembership(char *
pMsgPtr):
Protocol bcast w/membership. Current/new[2/1] providers.
[TOD(Feb 18 14:41:22)]( TRACE_FYI ) in SDaemonMessage::UpdateProtocolMembership(char *
pMsgPtr):
Being primed[[1/0/(14)][1/6/(42)]]
[TOD(Feb 18 14:41:22)]( TRACE_FYI ) in
SJoinProtocol::SJoinProtocol():Joiners[[1/5/(192)]
Destruct  join protocol [0x30048468], cancel sticky_submit,  container[0x3004a8e8].
.....
[TOD(Feb 18 16:23:00)]( TRACE_FYI ) in MGGroup::DisplayAll(): 1.1 ha_em_peers: 0 6 5
[TOD(Feb 18 16:23:00)]( TRACE_FYI ) in MGGroup::DisplayAll() : 0.Nil ZtheNameServerXY: 0.Nil 6.Nil
5.32
[TOD(Feb 18 16:23:00)]( TRACE_FYI ) in MGGroup::DisplayAll(): 0.Nil theGROVELgroup:
```

This log file shows us the phases that Node 5 went through, beginning with the abnormal termination of the hags daemon, leaving the group, getting itself into an “uncertain” state, starting hagsd again, sending a join request to the Group Leader, and finally becoming a member of the group again.

The hats daemon has protocol in the /var/ha/log/hats.18.121759.sp2en3 log file on Node 5:

```
02/18  14:41:20 hatsd[0]: Received a HB_GET_CONFIG_INFO request from client PID = 19186.
02/18  14:41:20 hatsd[0]: Received a HB_SUBSCRIBE request from client PID = 19186.
```

The process with PID 19186 is the newly started hagsd.

The situation on Node 5 has been noticed by all nodes in the group, and the log file on Node 6, /var/ha/log/hags_6_6.sp2en3, shows us the situation as seen from that node:

```
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in pbs_node_down: Node 5 down, G=1.1, ha_em_peers, groupGL=0
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in MGApplyDeleteToGroup: 0.Nil ZtheNameServerXY: 5
.....
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in MGApplyDeleteToGroup:  1.1 ha_em_peers: 5
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in SGroup::HandleMetaGroupDelete(NodeNum_node):
Group name[ha_em_peers]Handling node number [5] deleted from the meta-group.
```

The log file on the Control Workstation, /var/ha/log/hags.sp2en3_0_13.sp2en3, shows us the actions that were taken by the Group Leader:

```
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in pbs_inhale_input_message:Node 5 sent us
Container at 0x3005b458 (D=65535.Nil G=0.Nil HWM=Nil LWM=Nil SUPP-MSG plLen=10)
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in NS::NsIncoming GrovelMsg: from 5 Grovel
nodeIncarnation = 32 my nsState = NS::ENsState(7):kBecomeNS.
( TRACE_FSM_FYI ) in NS::NsIncoming GrovelMsg:  Got a non-stale Grovel message
from node 5 in my domain. I′m deleting and inserting this node.
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in NS::DriveNextProtocol: About to start NS
Delete protocol for DeletionList:  5.31.
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in pbs_node_down: Node 5 down, G=1.1,
ha_em_peers, groupGL=0, weAreTheGL
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in MGDeleteBuild:  1.1 ha_em_peers: 5
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in NS::Sent: NS  Delete  protocol completed.
[TOD(Feb 18 14:41:20)]( TRACE_FYI ) in NS::DriveNextProtocol:About to start NS
Insert protocol for InsertionList:  5.32.
```

We see that, for some reason, Node 5 had a non-stale status for a while, so the Group Leader (GL, here the Control Workstation) deleted Node 5 from the group, and after the new incarnation of hagsd was started on Node 5, the GL added this node to the group again.

What about hagsd terminating normally? How do we recognize that?

The hags subsystem is inoperative and the hagsd process is not running, as seen in the following:

```
#
lssrc -s hags
 hags            hags                      inoperative
# ps -ef|grep hagsd
    root 14996 13468   0 09:30:28  pts/0  0:00 grep hagsd
```

Reading the log file on the node, /var/ha/log/hags_5_32.sp2en3, we see that hagsd stopped at a normal request from the SRC:

```
[TOD(Feb 18 21:33:20)]( TRACE_FYI ) in SSuppSocket::ReadData(void *p, int readlen: Received
signal
[SIGTERM (15)]
[TOD(Feb 18 21:33:20)]( TRACE_FYI ) in SRCSocket::StopNormal(void): STOP NORMAL request from
SRC,
stop_requested was.
[TOD(Feb 18 21:33:20)]( TRACE_FYI ) in DispatchControl::Dispatcher(void):
hagsreap process finished. Removed [0] files.
```

The information about the status of the hags subsystem, taken from another member of the group, looks like this:

```
# lssrc -ls hags
Subsystem         Group            PID     Status
 hags             hags             17042   active
 2 locally-connected clients.  Their PIDs:
 13980 18780
 HA Group Services domain information:
 Domain established by node 0.
 Number of groups known locally: 1
                   Number of   Number of local
 Group name        providers   providers/subscribers
 cssMembership          2           1          0
 ha_em_peers            2           1          0
```

The GS nameserver is the Control Workstation, there are two providers in the group ha_em_peers (the Control Workstation and Node 6), the process with PID 13980 is hagsd, and the process with PID 18780 is hagsglsmd, which maintains the information for the switch group.

Let us have a look at the nodes belonging to the group:

```
#hagsmg -s hags
2.1 cssMembership: 5 6 0
1.1 ha_em_peers: 0 6 5
0.Nil ZtheNameServerXY: 0.Nil 6.6 5.32
0.Nil  theGROVELgroup:
```

The list of the members of the group did not change, even though Node 5 left the group. The group information in the GS domain (entered from the Control Workstation) gives us the same details:

```
#hagsgr -s hags.sp2en3
  Number of: groups: 5
Group slot # [0] Group name[HostMembership] group state[Not Inserted |]
Providers[]
Local subscribers[]

Group slot # [1] Group name[!SwitchGroupie!] group state[Idle |]
Providers[]
Local subscribers[]

Group slot # [2] Group name[ha_em_peers] group state[Inserted |Idle |]
Providers[[0/0][0/6][0/5]]
Local subscribers[[9/0]]

Group slot # [3] Group name[enMembership] group state[Idle |]
Providers[[0/0][0/5][0/6]]
Local subscribers[[9/0]]

Group slot # [4] Group name[cssMembership] group state[Needs Priming |Not Inserted |]
Providers[[0/5][0/6]]
Local subscribers[[9/0]]
```

So we see that if the system does not use the GS and hagsd is inoperative, it is not easy to notice that the daemon is not running, unless you search the log files or explicitly use the `lssrc -g hags` command, perhaps using dsh to get this information about all nodes in the system.

What is the behavior of the system?

Now what about host_responds while hagsd is inoperative? They will remain as they were; for example:

```
# SDRGetObjects host_responds
node_number  host_responds
          1           1
          2           1
          5           1
          6           1
```

The host_responds refer to the heartbeat, and hatsd is still active. As long as hagsd is inoperative on Node 5, the node is excluded from the group.
This was the situation on Node 5:

```
#lssrc -s hags
Subsystem         Group            PID     Status
 hags             hags                     inoperative
#lssrc -s hats
Subsystem         Group            PID     Status
 hats             hats             19632   active
```

Now what about the opposite situation?

```
#lssrc -s hags
Subsystem         Group            PID     Status
 hags             hags             19555   active
#lssrc -s hats
Subsystem         Group            PID     Status
 hats             hats                     inoperative
```

This “combination” is much more critical for the system. The node with the missing hatsd loses the host_responds:

```
# SDRGetObjects host_responds
          1           1
          2           1
          5           1
          6           0
```

The hagsd seems to be active, but in reality its only activity is to fill the /var/ha/log/gs.default.<syspar_name> log file with the same message about the failing heartbeat. Let us have a look at this logfile:

```
# cat /var/ha/log/gs.default.sp2en3
hb_init:  hb_init_communication failed
hb_init:  hb_init_communication failed
hb_init:  hb_init_communication failed
hb_init:  hb_init_communication failed
hb_init:  hb_init_communication failed
and so on ...
```

This file is extended with these error messages as long as hatsd is not running. Have a look at the size of the logfile:

```
# ls -l gs.default.sp2en3
-rwxr-xr-x   1 root     system      2106 Feb 18 22:54 gs.default.sp2en3
# ls -l gs.default.sp2en3
-rwxr-xr-x   1 root     system      2145 Feb 18 22:54 gs.default.sp2en3
# ls -l gs.default.sp2en3
-rwxr-xr-x   1 root     system      2184 Feb 18 22:54 gs.default.sp2en3
# ls -l gs.default.sp2en3
-rwxr-xr-x   1 root     system      2574 Feb 18 22:54 gs.default.sp2en3
```

The size of the log file grows with time, and if the /var file system reaches 100%, the system will have problems.

GS logs will be trimmed automatically if they occupy more than 5 MB of disk space. Up to the 5 MB point, GS will keep as many logs as it can. The daemon will automatically trim the current log to keep it no larger than 5000 lines.

The TS logs, due to an error, are not automatically trimmed. You need IX64893 (part of PSSP PTF10) and then these will remove all but the most recent 2 hats logs and will trim the current log every 5000 lines.

If desired, you can manually remove all but the most recent (current) logs. Once you have installed IX64893, we recommend you not manually remove any logs from this directory, if you report any problems with the HA daemons the support center will need the data from the logs to diagnose problems. With this fix, the subsystems will occupy, under normal conditions, about 8 MB of disk space in /var.

What is the consequence to the system of missing hags daemons?

Daemons running for GS provide services for coordinating and monitoring changes to the state of a subsystem running on a set of nodes, and services for synchronizing the consistent recovery of multiple subsystems.

With this facility offered by GS, the members of the groups are able to perform synchronized actions in response to failures or other conditions important to the groups. GS informs all members of the group automatically if a member, or the node on which the member is running, fails.

On the other hand, the members themselves can take advantage of the facility for detecting other events of interest to the subsytem and use GS for coordinating their actions in responding to the events.

There are some reserved groups created by GS that consist of all processors and networks in the system. Subsystems may subscribe to these groups to be informed about the node and/or adapter failures. To detect these types of problems, GS uses Topology Services.

Now if hagsd is not running, these services can no longer be provided for this node. As in a clockwork, hags is one of the components of the whole mechanism and if it fails, all the other components, such as Event Management, may be affected or confused by this failure (see also 1.2.6, “Relationship between the HA Components” on page 23).

Note

Who can use GS? Any process of multicomputer applications or subsystems can be a GS client. GSAPI lets you make the application act as a GS client. PSSP 2.2 contains two GS clients:

- Event Management
- Recoverable Virtual Shared Disk

HACMP/ES and DB2 Parallel Edition are currently planned.

We have discussed the situation where hagsd is inoperative. Now let us have a look at the problem determination of a similar situation, where hagsd cannot start.

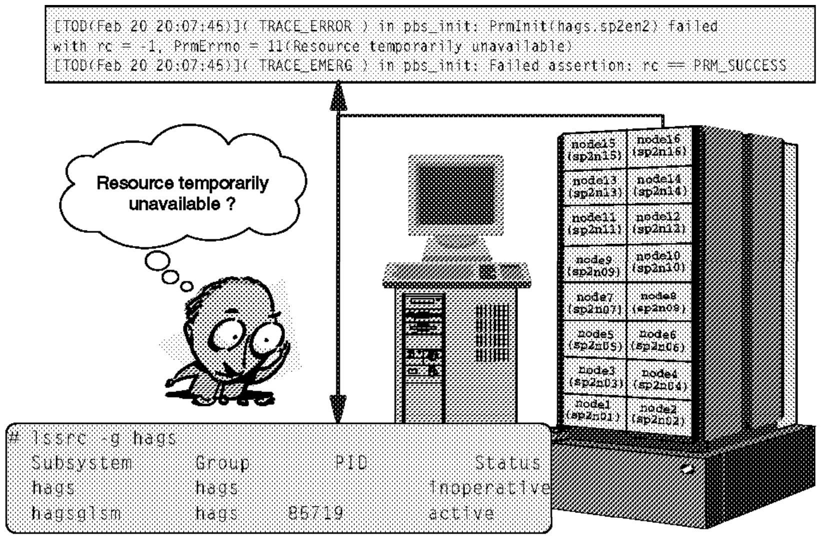
Symptoms:

hagsd is not running.

Environment:

AIX 4.2 and PSSP 2.2 running on the Control Workstation and all nodes in this partition.

Problem Description:



```
[TOD(Feb 20 20:07:45)]( TRACE_ERROR ) in pbs_init: PrmInit(hags.sp2en2) failed
with rc = -1, PrmErrno = 11(Resource temporarily unavailable)
[TOD(Feb 20 20:07:45)]( TRACE_EMERG ) in pbs_init: Failed assertion: rc == PRM_SUCCESS
```

```
# lssrc -g hags
  Subsystem       Group         PID         Status
  hags            hags                    inoperative
  hagsglsm        hags     85719          active
```

Figure 9. Problems Starting hagsd

The problem is that hagsd is not running. This fact was not discovered for a while because our system did not use GS for other than the default tasks and we did not have any monitor for the AIX Error log entries. Normally, you will be informed that hagsd is not running through the AIX Error log facility. While controlling the systems from the Control Workstation, we saw that hagsd was inoperative and hagsglsm was running.

Problem Determination:

The first step is to restart the daemon. The output of this action was as follows:

```
# hagsctrl -s
0513-059 The hags subsystem has been started. Subsystem PID is 16942.
0513-059 The hagsglsm subsystem has been started. Subsystem PID is 18896.
```

It seems to work this time, but let us check if the daemons are really running:

```
# lssrc -g hags
Subsystem         Group            PID     Status
 hagsglsm         hags             18896   active
 hags             hags                     inoperative
```

We see that hags tries to start, even gets a PID, but dies a few seconds later.

The next step is to look at the log files. Let us start with /var/hags_<node number>_<incarnation number>.<syspar_name>:

```
.....
[TOD(Feb 20 20:07:45)]( TRACE_ERROR ) in pbs_init: PrmInit(hags.sp2en2) failed
with rc = -1, PrmErrno = 11(Resource temporarily unavailable)
TRACE_EMERG ) in pbs_init: Failed assertion: rc == PRM_SUCCESS.
```

The file tells us that while the daemon was being initialized, it could not use some resources because they were temporarily unavailable.

Let us have a look at the next log file on the node, /var/ha/log/gs.default.<syspar_name>:

```
Assertion failed: rc == PRM_SUCCESS, file  ../../../../../../srcssp/availability
/pgs/pgsd/pbs/pbs_init.C, line 173
.....
    n    CumSize       MB    weight       size name
    1       1061    0.001      0.00       1061 hags_16_21.sp2en2
    2       2742    0.003      0.48       1681 hags_16_20.sp2en2
    3    1207688    1.152      0.53    1204946 core_16.20
.....
```

We see that a core file was created in directory /var/ha/run/hags.<syspar_name>. This core file also has some information about a resource that was already in use:

```
.....
invalid signal or routine specified for SRC signal communication
.....
failed to dup() SRC socket to desired file descriptor
.....
invalid parameters passed to dae_init_exclusive() routine
an instance of the daemon is already running; multiple instances not allowed
.....
```

This log file gives us the same information: an instance of the daemon is already running.

Let us recall the process of initializing the GS daemon. First it gets the number of the node on which it is running by using the /usr/lpp/ssp/install/bin/node_number command. Next it fetches the name of the system partition from the Syspar SDR class and ensures that the port number is set in the /etc/services file. If there is no port number in the SDR and the hagsctrl script is running on the Control Workstation, the script obtains a port number. If the hagsctrl script is running on the node and there is no port number in the SDR, the script ends with an error.

Next, hagsd tries to establish a communication with its neighbors using the UDP port specified in the SDR. All the local connections between GS and its clients are made through Unix Domain Sockets located in the /var/ha/soc directory. Let us have a look at the Syspar_ports class in the SDR:

```
# SDRGetObjects Syspar_ports
subsystem    port
hats              10009
hags              10010
haem              10011
```

Now let us see if the 10010 port is defined in /etc/services for hagsd:

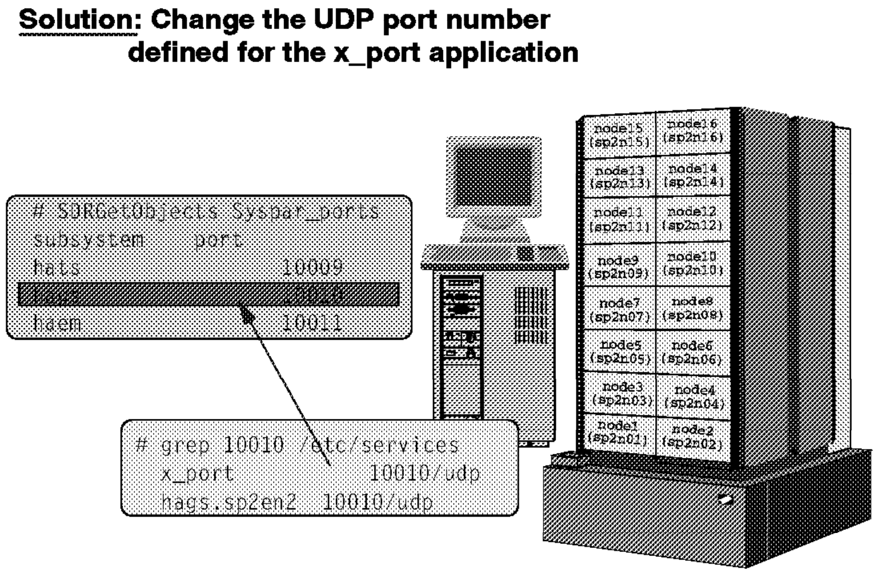
```
# grep 10010/udp /etc/services
x_port         10010/udp
hags.sp2en2    10010/udp
```

This is our problem!

Another application called x_port is allowed to use the same port number as our hagsd. At the time we wanted to start our hagsd, the x_port application was active on this port.

Conclusion:

Figure 10 on page 45 shows the result of our problem determination in this case.



```
# SDRGetObjects Syspar_ports
subsystem     port
hats                 10009
hags                 10010
haem                 10011
```

```
# grep 10010 /etc/services
  x_port              10010/udp
  hags.sp2en2  10010/udp
```

Figure 10. Result of Missing hagsd Problem Determination

The UDP port was already used by another application.

Solution:

Change the UDP port number defined for the x_port application. Make sure again that you do not choose a port number that can be used by some other application.

Note

If the UDP port that the application is using cannot be changed, change the Syspar_port SDR class to a different port number for Group Services.

1.3.1.3 What Happens If haemd Dies?

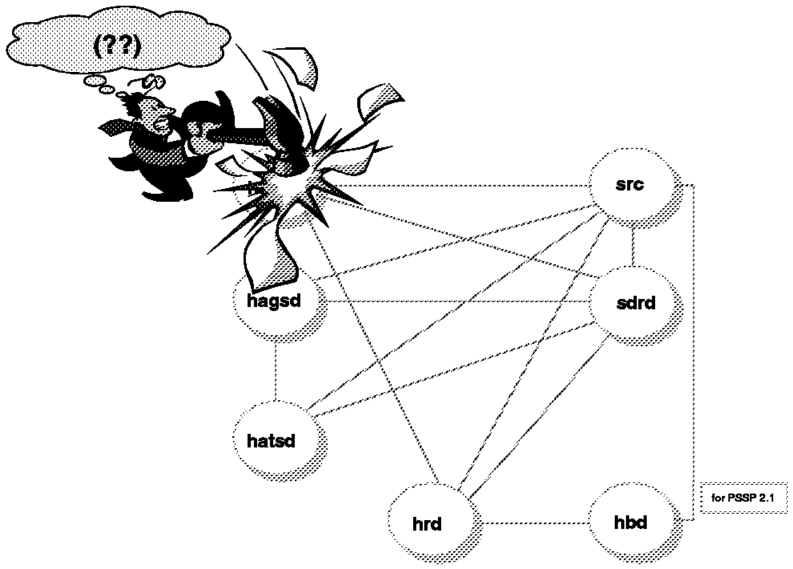


Figure 11. What Happens If haemd Dies?

How do I recognize it?

Like other partition-sensitive subsystems, the Event Management subsystem normally does not require any additional administrative intervention. It recovers automatically from temporary failures.

So as with other HA daemons, if haemd dies, SRC tries to restart it. You may not notice this if you do not use the EM subsystem, because haemd will still be active, but it will have another PID. But you can find out why the daemon died from the output of the `errpt` command, which shows you whether its death was on purpose or by accident.

In the following output of `errpt -a`, we see that haemd was killed.

```
LABEL:          HA001_TR
IDENTIFIER:     99FA80C7

Date/Time:       Wed Feb 19 19:50:09
Sequence Number: 3346
Machine Id:      000041005700
Node Id:         sp2n06
Class:           S
Type:            UNKN
Resource Name:   haemd
Description
SOFTWARE

Probable Causes
SUBSYSTEM

Failure Causes
SUBSYSTEM

        Recommended Actions
        NONE

Detail Data
DETECTING MODULE
LPP=PSSP,Fn=emd.c,SID=1.26,L#=364,

DIAGNOSTIC EXPLANATION
haemd: Received termination signal SIGTERM, program is exiting.
```

Another error report shows us that haemd was terminated by the SRC:

```
LABEL:          HA001_TR
IDENTIFIER:     99FA80C7

Date/Time:       Sun Feb 23 14:53:20
Sequence Number: 9458
Machine Id:      000204085700
Node Id:         sp2n16
Class:           S
Type:            UNKN
Resource Name:   haemd

Description
SOFTWARE

Probable Causes
SUBSYSTEM

Failure Causes
SUBSYSTEM

        Recommended Actions
        NONE

Detail Data
DETECTING MODULE
LPP=PSSP,Fn=emd.c,SID=1.26,L#=508,
DIAGNOSTIC EXPLANATION
haemd: Received ″normal″ stop indication from SRC, program is exiting.
```

What is the consequence to the system of missing haemd?

The Event Management Subsystem is a collection of software components like resource monitors, Event Manager daemons, and *client programs*. Client programs are applications or other subsystems that wish to receive event information when resources change state. The Event Manager daemon notes the client programs that have subscribed and notifies them when events occur in which they are interested. The data transfer from resource monitors to the Event Manager daemon is performed by the Performance Toolbox/6000 System Performance Measurement Interface. If an event is generated, it is sent to the registered clients through the Event Management Application Programming Interface (EMAPI). The EMAPI interface passes the data to clients like the Problem Management daemon (pmand), host responds daemon (hrd), or the Perspectives GUI.

If haemd is inoperative, no monitoring of the system, subsystem and application resource can be performed and no events can be generated when the resources change state. So if you use pman error notification or Perspectives and haemd is missing, you get error messages regarding the absence of the EM daemon.

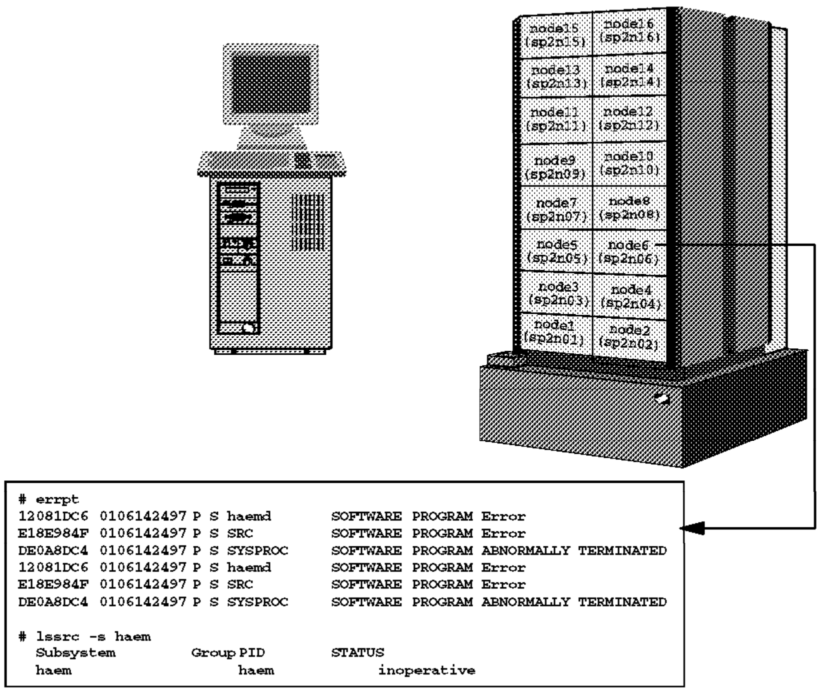
In this section we describe a problem determination path for the recovery of the haemd that is not running.

Symptom: haemd is not running.

Environment:

AIX 4.2 and PSSP 2.2 on the Control Workstation, AIX 4.1.4 and PSSP 2.2 on the node.

Problem Description



```
# errpt
12081DC6  0106142497 P S haemd       SOFTWARE PROGRAM Error
E18E984F  0106142497 P S SRC         SOFTWARE PROGRAM Error
DE0A8DC4  0106142497 P S SYSPROC     SOFTWARE PROGRAM ABNORMALLY TERMINATED
12081DC6  0106142497 P S haemd       SOFTWARE PROGRAM Error
E18E984F  0106142497 P S SRC         SOFTWARE PROGRAM Error
DE0A8DC4  0106142497 P S SYSPROC     SOFTWARE PROGRAM ABNORMALLY TERMINATED

# lssrc -s haem
  Subsystem        Group PID        STATUS
  haem                  haem             inoperative
```

Figure 12. Problem with haemd on Node 6

There are many entries in the AIX error log on the node and haemd is inoperative. After trying `haemctrl -s` on the node, the following error message appears:

```
haemd:  2521-025 Cannot copy /etc/ha/cfg/em.sp2en3.cdb from CWS.
```

Problem Determination:

The first step of the problem determination is to see whether there is a /etc/ha/cfg/em.sp2en3.cdb file on the node. In our case, the file existed. What is the meaning of this file? During initialization of the Event Manager daemon on the Control Workstation, haemd checks the SDR Syspar class for the correct version of the configuration database (EMCDB):

```
SDRGetObjects Syspar syspar_name ip_address code_version haem_cdb_version
syspar_name  ip_address   code_version haem_cdb_version
sp2en3       192.168.3.40 PSSP-2.2     854065238,626525952,0
```

Then haemd copies the compiled configuration file from /spdata/sys1/ha/cfg to its active directory, /etc/ha/cfg. When the Event Manager daemon starts up on the node, it uses the /usr/lpp/ssp/install/bin/haemrcpcdb Korn shell script to copy the EMCDB from the Control Workstation to the local node, using the RCP protocol. In our case, the EMCDB file existed and was the right version, so the problem was probably the `rcp` command. But let us have a look at the log file /var/ha/log/em.default.<syspar_name> before continuing with our suspect, `rcp`. The file consisted of one line:

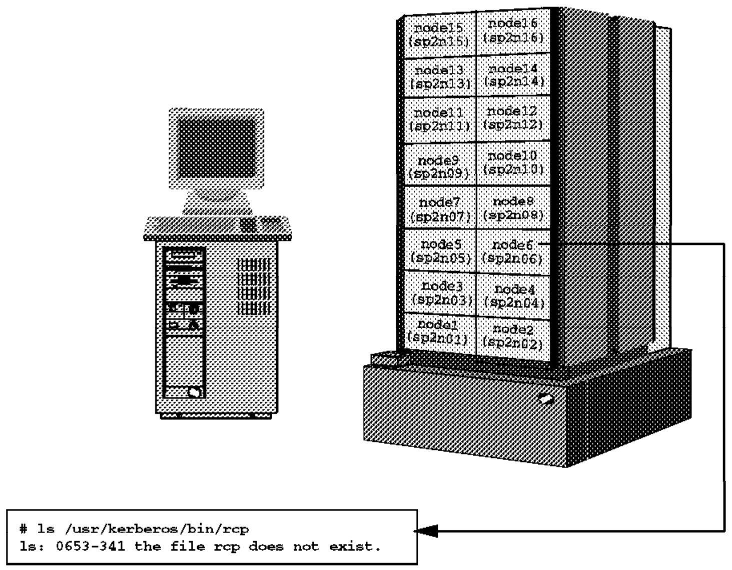
```
# cat /var/ha/log/em.default.sp2en3
ksh:  /usr/kerberos/bin/rcp:  not found.
```

Let us now look in the /usr/kerberos/bin directory:

```
# cd /usr/kerberos/bin
# ls -la
total 1352
drwxr-xr-x   2 bin      bin          512 Jan 22 18:36 .
drwxr-xr-x   4 root     security     512 Jan 22 12:15 ..
-r-xr-xr-x   1 bin      bin        81088 Oct 14 20:05 add_principal
-r-xr-xr-x   1 bin      bin       100428 Oct 14 20:05 kadmin
-r-xr-xr-x   1 bin      bin         5736 Oct 14 20:04 kdestroy
-r-xr-xr-x   1 bin      bin        83170 Oct 14 20:04 kinit
-r-xr-xr-x   1 bin      bin        21980 Oct 14 20:04 klist
-r-xr-xr-x   1 bin      bin        79756 Oct 14 20:05 kpasswd
-r-xr-xr-x   1 bin      bin        42622 Oct 14 20:04 ksrvtgt
-r-xr-xr-x   1 bin      bin       251136 Oct 14 20:05 ksrvutil
```

We see that the link of rcp from /usr/lpp/ssp/rcmd/bin/rcp is missing.

Conclusion:



```
# ls /usr/kerberos/bin/rcp
ls: 0653-341 the file rcp does not exist.
```

Figure 13. Result of Missing haemd Problem Determination

The problem was that the link in /usr/kerberos/bin was missing. After recreating the link, haemd started successfully.

Solution:

Recreate the rcp link in /usr/kerberos/bin via the command:

```
# cd /usr/kerberos/bin
# ln -s /usr/lpp/ssp/rcmd/bin/rcp rcp
```

Another aspect of having problems with haemd is related to running out of *paging space* on a *switchless* SP system, where haemd is the process that is consuming the paging space.

APAR IX64764 fixes this problem (PTF set 4). If you do not have this APAR installed, or do not want to install it, here is a recommendation for a workaround:

Workaround

The memory leak in haemd can be prevented by removing the definition of a resource variable from the EMCDB. It seems that haemd does a check and cannot get a response from `SDRGetObjects Switch` and retries it every 10 seconds in an endless loop.

To work around this situation, do the following:

1. Remove the resource variable definition for IBM.PSSP.Response.Switch.state from the SDR:

```
# SDRDeleteObjects EM_Resource_Variable rvName==IBM.PSSP.Response.Switch.state
```

2. Build a new EMCDB file:

```
# haemcfg
```

3. Stop the Event Management daemons on the CWS and *all* nodes:

```
# haemctrl -k
# dsh -a haemctrl -k
```

4. Verify that no haemd daemons are active:

```
# lssrc -g haem
# ps -ef|grep haem
# dsh -a lssrc -g haem
# dsh -a ps -ef|grep haem
```

5. Exit from any instance of Perspectives.
6. Remove the switchResponds condition from the SDR:

```
# SDRDeleteObjects EM_Condition name==switchResponds
```

7. Restart Event Management daemons on the CWS and *all* nodes:

```
# haemctrl -s
# dsh -a haemctrl -s
```

8. Restart Perspectives.

After completing these steps, the EM daemons are using a new version of the EMCDB. This can be verified after executing the preceding steps and then using the following command:

```
# lssrc -ls haem
.....
 Configuration Data Base version: 854067075,797629440,0(SDR)
.....
```

1.3.1.4 What Happens If hrd Dies?

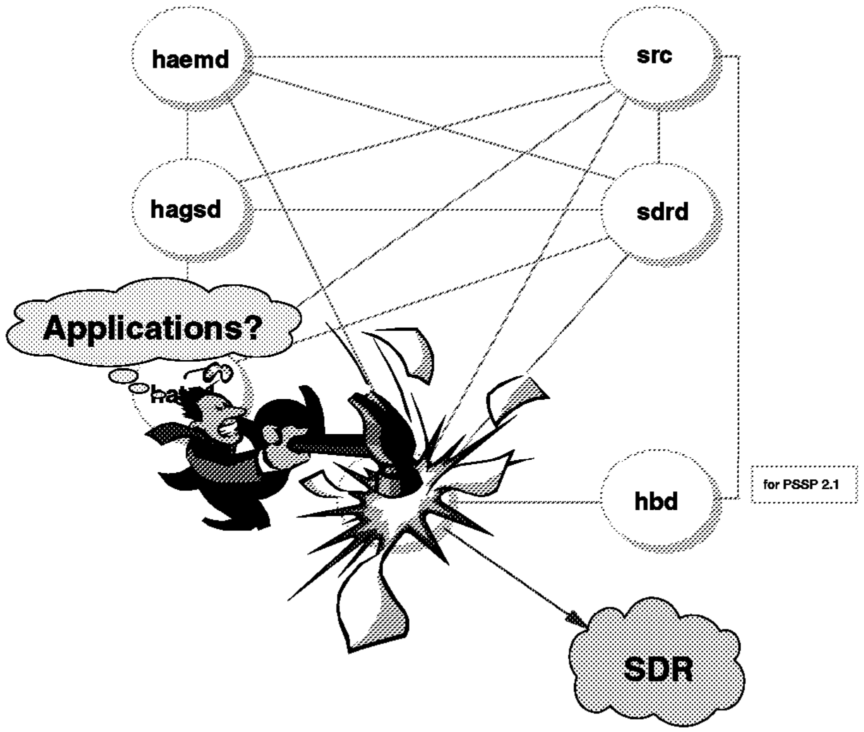


Figure 14. What Happens If hrd Dies?

The host responds daemon updates the host_responds class in the SDR. If there are any changes, it is hrd that maintains them in the SDR. The daemon is invoked by the /usr/lpp/ssp/hr Korn shell script, which is under SRC control, and respawns automatically.

The hrd daemon can perform two kinds of monitoring:

- Using the system heartbeat (hbd)
- Using ping (running ping to many nodes asynchronously, and also issuing `snmpinfo`, calling each node to detect a situation where ping succeeds but the node is not responding to user requests).

By setting `HR_FPING` in the /usr/lpp/ssp/bin/hr script, you decide whether you are going to use the heartbeat (`HR_FPING=0`) or ping (`HR_FPING=1`).

The host responds daemon runs only on the Control Workstation. It obtains the data from the Event Manager layer for the nodes running PSSP 2.2, and from hbd for the nodes with PSSP 2.1 or older.

If hrd dies, the SDR is not updated for nodes with PSSP level 1.2 and 2.1 that belong to this partition and whose host_responds status changed.

Let us take, as example, the situation where hrd on the Control Workstation is inoperative and host_responds are set to 1 for all nodes in the partition:

```
# lssrc -s hr.sp2en3
Subsystem         Group            PID     Status
 hr.sp2en3        hr                       inoperative
# SDRGetObjects host_responds
node_number  host_responds
          1            1
          2            1
          5            1
          6            1
```

It was not necessary to change the host_responds value yet, and the fact that hrd is not running for this partition is not visible yet. But what happens if one of the nodes running PSSP 2.1 or older loses the host_responds? No update will be done in the SDR and the value for host_responds for this node remains 1; this does not match the real situation:

```
# dsh -w sp2n02 lssrc -s hb
sp2n02: Subsystem         Group            PID     Status
sp2n02:  hb               hb                       inoperative

#SDRGetObjects host_responds
node_number  host_responds
          1            1
          2            1
          5            1
          6            1
```

1.3.1.5 What Happens If SRC (/usr/sbin/srcmstr) Dies?

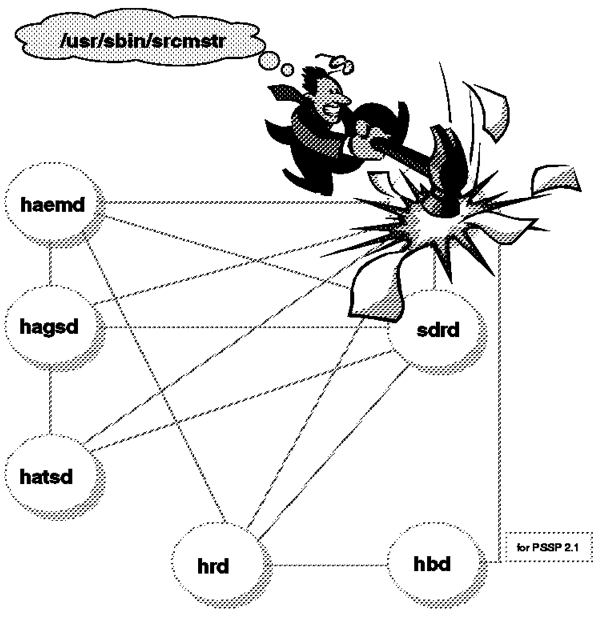


Figure 15. What Happens If SRC (/usr/sbin/srcmstr) Dies?

We previously discussed daemon initialization and mentioned the relationship between the daemons and the SRC subsystem. Normally, the HA daemons establish connection with the SRC that is necessary for getting status information about the daemons in response to SRC commands. If the process /usr/sbin/srcmstr is not running on the system, neither a query about the subsystem status nor the execution of the control commands is possible:

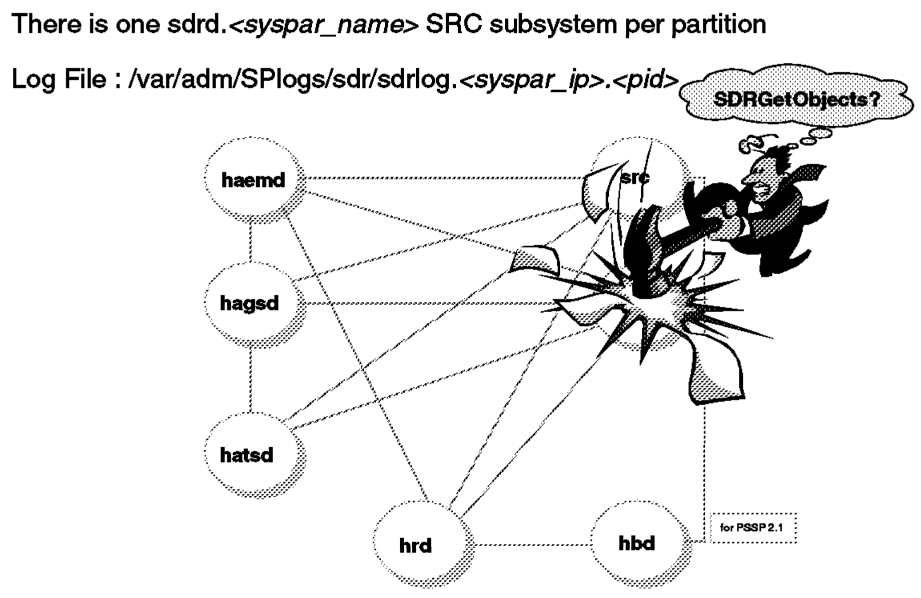
```
# ps -ef|grep src
    root 16026 17414   1 12:59:28  pts/1  0:00 grep src
#
# lssrc -s hags
0513-001 The System Resource Controller daemon is not active.

# hatsctrl -r
0513-001 The System Resource Controller daemon is not active.
```

And what is the consequence if the SRC process dies? Daemons related to the SRC, such as the HA deamons hatsd, hagsd, and haemd, die as well! Let us see which processes on the system depend on the SRC:

```
#cat /etc/inittab
.....
nim:2:wait:/usr/bin/startsrc -g nim >/dev/console 2>&1
sdrd:2:once:/usr/bin/startsrc -g sdr
.....
qdaemon:2:wait:/usr/bin/startsrc -sqdaemon
writesrv:2:wait:/usr/bin/startsrc -swritesrv
.....
hardmon:2:once:/usr/bin/startsrc -s hardmon
sysctld:2:once:/usr/bin/startsrc -s sysctld
splogd:2:once:/usr/bin/startsrc -s splogd
hats:2:once:/usr/bin/startsrc -g hats > /dev/console 2>&1
hb:2:once:/usr/bin/startsrc -g hb > /dev/console 2>&1
hr:2:once:/usr/bin/startsrc -g hr
hags:2:once:/usr/bin/startsrc -g hags > /dev/console 2>&1
haem:2:once:/usr/bin/startsrc -g haem > /dev/console 2>&1
pman:2:once:/usr/bin/startsrc -g pman >/dev/console 2>&1
sp_configd:2:once:/usr/bin/startsrc -s sp_configd
.....
```

If the SRC process is not already running when booting the system, the subsystems and processes depending on the SRC will not start.

1.3.1.6 What Happens If the sdr.<syspar_name> Daemon Dies?



There is one sdrd.*<syspar_name>* SRC subsystem per partition

Log File : /var/adm/SPlogs/sdr/sdrlog.*<syspar_ip>.<pid>*

Figure 16. What Happens If sdr.<syspar_name> Dies?

The SDR cannot be accessed any more, that is, hrd cannot update the SDR host_responds class, if the resource variable Response.Host.state changes.

The Event Manager daemon uses the SDR primarily as a repository of configuration information. In addition, on the Control Workstation, the daemon polls the SDR for information that it uses to create instances of the IBM.PSSP.Response.Switch.State resource variable. This information is taken from the switch_responds class. If the sdr.<syspar_name> daemon is not running, these operations cannot be performed.

While the sdr.<syspar_name> daemon is not running, the other HA subsystems cannot be configured any more (if needed). The UDP port number is fetched from the SDR.

The SDR daemon writes information to a log named /var/adm/SPlogs/sdr/sdrdlog.<syspar_ip_addr>.<pid>, where syspar_ip_addr is the IP address of the system partition and pid is the process ID of the SDR daemon (sdrd process). This log contains the date and time the process started, as well as problems encountered by the daemon during the operation. The content of the log file is:

```
Invocation: /usr/lpp/ssp/bin/sdrd 192.168.3.39
   pid = 12120
   time = Sat Jan 25 18:24:21 1997


Thu Jan 30 13:11:24 1997
:Client sp2n16:10802:1 disconnected. RC=110 from SDRRead (line 347)
Errno=73
Wed Feb 12 12:19:20 1997
:Client sp2n15:5188:1 disconnected. RC=110 from SDRRead (line 347)
Errno=73
```

The output of spmon -d if the sdr.<syspar_name> daemon is missing is:

```
# spmon -d
1. Checking server process
   Process 18750 has accumulated 10 minutes and 30 seconds.
   Check ok

2. Opening connection to server
```

1.3.1.7 What Happens If the GS Nameserver Dies?

GS guarantees that each node has the same information about the list of nodes, so that all nodes know which one is next on the list. When the GS nameserver dies, the next node on the list gets the nomination message from the nodes and becomes a new GS nameserver as soon as it sends and commits the message back to the nodes. If the previous GS nameserver comes back, it joins the MetaGroup and is put at the end of the list.

How can we determine which machine is a GS nameserver?

Log on to the Control Workstation and issue the following command:

```
lssrc -ls hags.partition_name
```

```
# lssrc -ls hags.sp2en2
Subsystem         Group            PID     Status
 hags.sp2en2      hags             27502   active
 2 locally-connected clients.  Their PIDs:
 28876 20134
 HA Group Services domain information:
 Domain established by node 0.
 Number of groups known locally: 2
                   Number of   Number of local
 Group name        providers   providers/subscribers
 cssMembership          2           0           1
 ha_em_peers            5           1           0
```

Let us have a look at the HA GS domain information. The message `Domain established by node 0` means that the domain was established and the GS nameserver for this domain is Node 0, which is the Control Workstation.

Now we know which is the GS nameserver, and we took a look at the list of the nodes that belong to this domain. The following command, entered from the node, shows us the members of each group (Ethernet and Switch) and their order in the list:

```
hagsmg -s hags.sp2en2
```

```
2.1 cssMembership: 15 0 16
1.1 ha_em_peers: 0 15 11 16 12
0.Nil ZtheNameServerXY: 0.Nil 15.Nil 11.Nil 16.1 12.2
0.Nil  theGROVELgroup:
```

The line:

```
0.Nil ZtheNameServerXY: 0.Nil 15.Nil 11.Nil 16.1 12.2
```

shows us a list of nodes that are in the domain called MetaGroup, listing all the nodes in the partition that are running the GS deamon.

The syntax of the line is: node_number.incarnation_number, where incarnation_number is the number of times the GS daemon has been started on this node. So, in our example, `12.2` means that hagsd was started twice on Node 12. `15.Nil` means that the node on which this command was executed does not know the incarnation_number for Node 15. This is working as designed because only the GS nameserver knows the incarnation_number for every node. So if you enter the same command, but this time from the GS nameserver, which in our case is the Control Workstation, you get the following output:

```
# hagsmg -s hags.sp2en2
2.1 cssMembership: 15 0 16
1.1 ha_em_peers: 0 15 11 16 12
0.Nil ZtheNameServerXY: 0.6 15.1 11.0 16.1 12.2
0.Nil  theGROVELgroup:
```

As mentioned before, the next node on the list becomes the nameserver if the GS nameserver dies, Now let us check the lists again:

```
# lssrc -ls hags
Subsystem         Group            PID     Status
 hags             hags             18344   active
 2 locally-connected clients.  Their PIDs:
 16828 19922
 HA Group Services domain information:
 Domain established by node 15.
 Number of groups known locally: 2
                    Number of   Number of local
 Group name         providers   providers/subscribers
 cssMembership           2           1           0
 ha_em_peers             5           1           0
```

```
# hagsmg -s hags.sp2en2
2.1 cssMembership: 15 0 16
1.1 ha_em_peers: 15 11 16 12 0
0.Nil ZtheNameServerXY: 15.1 11.0 16.1 12.2 0.7
0.Nil  theGROVELgroup:
```

Node 15 is now the GS nameserver and the previous one, the Control Workstation, has been put at the end of ha_em_peers and the MetaGroup lists.

1.3.2 HA Directory Structure

1.3.2.1 /usr/lpp/ssp/bin

Programs used by the PSSP software.

1.3.2.2 GSAPI Shared Library

/usr/lpp/ssp/lib/libha_gs.a
/usr/lpp/ssp/lib/libha_gsr.a

1.3.2.3 GS Client Sample Programs

/usr/lpp/ssp/samples/hags

1.3.2.4 /var/ha/lck (Lock Files)

```
drwxr-xr-x   4 root     system       512 Jan 25 18:37 .
drwxr-xr-x   6 root     system       512 Jan 25 18:06 ..
----------   1 root     system         0 Jan 25 18:37 em.RMIBM.PSSP.harmld.sp2en2
-rwsrws---   1 root     system         0 Jan 25 18:37 em.RMIBM.PSSP.harmldSHM
----------   1 root     system         0 Jan 25 18:37 em.RMIBM.PSSP.harmpd.sp2en2
----------   1 root     system         0 Jan 25 18:17 em.RMIBM.PSSP.pmanrmd.sp2en2
----------   1 root     system         0 Jan 25 18:17 em.haemd.sp2en2
-rw-r--r--   1 root     system        20 Feb 21 07:54 hags.partition.save
drwxr-xr-x   2 root     system       512 Feb 21 07:54 hags.tid.sp2en2
-rw-r--r--   1 root     system        20 Feb 20 20:07 hagsglsm.partition.save
drwxr-xr-x   2 root     system       512 Feb 20 20:07 hagsglsm.tid.sp2en2
```

1.3.2.5 /var/ha/log (Log Files)

```
drwxr-xr-x   2 root     system      1024 Feb 21 07:54 .
drwxr-xr-x   6 root     system       512 Jan 25 18:06 ..
-rwxr-xr-x   1 root     system         0 Feb 23 20:54 em.default.sp2en2
-rwxr-xr-x   1 root     system       636 Feb 20 20:07 glsm.default.sp2en2
-rw-r--r--   1 root     system      1681 Feb 20 19:39 hags_16_16.sp2en2
-rw-r--r--   1 root     system      1680 Feb 20 19:39 hags_16_17.sp2en2
-rw-r--r--   1 root     system       551 Feb 20 20:06 hagsglsm_16_11.sp2en2
-rw-r--r--   1 root     system     20885 Feb 24 13:55 hagsglsm_16_12.sp2en2
-rw-rw-rw-   1 root     system     12107 Feb 24 12:59 hats.20.184753.sp2en2
-rw-rw-rw-   1 root     system      6581 Jan 25 18:24 hats.25.181729.sp2en2
-rwxr-xr-x   1 root     system         0 Feb 20 18:47 hats.sp2en2
```

1.3.2.6 /var/ha/run (Run-Time Directories for Each System Partition)

```
drwxr-xr-x  18 root     system       512 Jan 23 20:05 .
drwxr-xr-x   6 root     system       512 Jan 22 17:40 ..
drwxr-xr-x   7 root     system       512 Feb 05 14:06 haem.sp2en0
drwxr-xr-x   5 root     system       512 Jan 25 18:25 haem.sp2en1
drwxr-xr-x   6 root     system       512 Feb 11 09:43 haem.sp2en2
drwxr-xr-x   5 root     system       512 Jan 25 18:25 haem.sp2en3
drwxr-xr-x   2 root     system       512 Jan 22 17:49 hags.sp2en0
drwxr-xr-x   2 root     system       512 Jan 22 20:37 hags.sp2en1
drwxr-xr-x   2 root     system       512 Jan 23 20:04 hags.sp2en2
drwxr-xr-x   2 root     system       512 Jan 23 19:35 hags.sp2en3
drwxr-xr-x   2 root     system       512 Jan 22 17:49 hagsglsm.sp2en0
drwxr-xr-x   2 root     system       512 Jan 22 20:37 hagsglsm.sp2en1
drwxr-xr-x   2 root     system       512 Jan 23 20:04 hagsglsm.sp2en2
drwxr-xr-x   2 root     system       512 Jan 23 19:35 hagsglsm.sp2en3
drwxr-x---   2 root     system       512 Feb 05 14:07 hats.sp2en0
drwxr-x---   2 root     system       512 Jan 25 18:26 hats.sp2en1
drwxr-x---   2 root     system       512 Jan 25 18:26 hats.sp2en2
drwxr-x---   2 root     system       512 Jan 25 18:26 hats.sp2en3
```

1.3.2.7 /var/ha/soc (Socket Files)

```
drwxr-xr-x   3 root     system       512 Feb 18 23:14 .
drwxr-xr-x   6 root     system       512 Jan 22 18:40 ..
srw-------   1 root     system         0 Feb 18 23:14 em.RMIBM.PSSP.harmld.sp2en3
srw-------   1 root     system         0 Feb 18 23:14 em.RMIBM.PSSP.harmpd.sp2en3
srw-rw-rw-   1 root     system         0 Jan 23 09:37 em.clsrv.sp2en1
srw-rw-rw-   1 root     system         0 Feb 18 23:14 em.clsrv.sp2en3
srw-rw-rw-   1 root     system         0 Jan 23 09:37 em.rmsrv.sp2en1
srw-rw-rw-   1 root     system         0 Feb 18 23:14 em.rmsrv.sp2en3
srw-------   1 root     system         0 Jan 23 09:37 hagsdsocket.sp2en1
srw-------   1 root     system         0 Feb 18 23:14 hagsdsocket.sp2en3
drwxr-x---   2 root     system       512 Feb 18 12:17 hats
srw-------   1 root     system         0 Feb 18 23:14 pman
```

1.4 Coexistence

Problem determination of HA daemons in a coexistence environment traces back to the reciprocal action between hatsd and hb daemons. There are multiple instances of the hats daemon running for every partition on the Control Workstation, as follows:

```
# lssrc -g hats
Subsystem         Group            PID     Status
 hats.sp2en1      hats             26558   active
 hats.sp2en2      hats             29898   active
 hats.sp2en0      hats             11248   active
 hats.sp2en3      hats             61052   active
```

In addition to these processes, there are also multiple instances of hr and hb daemons running on the Control Workstation for every partition:

```
# lssrc -g hr
Subsystem         Group            PID     Status
 hr.sp2en3        hr               22848   active
 hr.sp2en0        hr               23388   active
 hr.sp2en1        hr               21662   active
 hr.sp2en2        hr               17334   active
# lssrc -g hb
Subsystem         Group            PID     Status
 hb.sp2en3        hb               15474   active
 hb.sp2en0        hb               22154   active
 hb.sp2en1        hb               22702   active
 hb.sp2en2        hb               23820   active
```

On the nodes running PSSP 2.2, the required HA daemons should be running:

```
# lssrc -a
.....
 hats             hats             8822    active
 pman             pman             15388   active
 pmanrm           pman             9290    active
 sp_configd                        8560    active
 hagsglsm         hags             18896   active
 hags             hags             19572   active
 haem             haem             14862   active
```

The nodes that are running PSSP 1.2 or 2.1 only have the hb daemon active:

```
# lssrc -a
.....
 hb               hb               12044   active
.....
```

For the nodes running at PSSP level 1.2 or 2.1, the host responds daemon (hrd) obtains information about the state of the nodes from the PSSP heartbeat daemon (hbd).

1.4.1 Common Problems

Most of the problems that occur after the installation/migration of the system to PSSP 2.2 are related to host_responds and switch_responds.

1.4.1.1 host_responds (GUI)

One of the known host_responds problems in a partition with PSSP 2.2 and 2.1 is that the node with PSSP 2.2 has host_responds set to 0 even though hatsd is running. Refreshing the subsystem with the `hatsctrl -r` command was successful. Now the confusion is complete: spmon -g hostResponds is red, spmon -d host_responds is “yes,” and Perspectives is green. It looks like the spmon process does not reconnect with hrd after refreshing and does not report the hr changes to the GUI. The best way to make sure about host_responds is to obtain it with the `SDRGetObjects host_responds` command.

1.4.1.2 host_responds (Missing Prerequisites)

After the installation/migration of the system, there is no host_responds for a node, or nodes. This problem is usually related to missing prerequisites for the PSSP file set ssp.ha. Mostly, it occurs when the perfagent.server file set does not exist in the /spdata/sys1/install<aix_version>lppsource directory. If this is the case, then load this file set into the directory and customize your nodes.

1.4.1.3 host_responds (Shared Memory)

Another aspect of having problems with host_responds is related to the usage of shared memory. There are two ways for passing the resource variables, depending on their status. One way is to pass the data as messages, the other is to use the shared memory. (The EM subsystem uses the segment of shared memory with an ID starting with 0x78.) Let us look at the output of the `ipcs -m` command on the Control Workstation:

```
# ipcs -m
IPC status from /dev/mem as of Sun Feb 23 21:11:31 EST 1997
T     ID     KEY        MODE       OWNER    GROUP
Shared Memory:
m      0 0x5806a933 --rw-rw-rw-     root   system
m  45057 0x6709056e --rw-r--r--     root   system
m  45058 0x6809056e --rw-r--r--     root   system
m  36867 0x670904de --rw-r--r--     root   system
m  36868 0x680904de --rw-r--r--     root   system
m   4101 0x670904e9 --rw-r--r--     root   system
m   4102 0x680904e9 --rw-r--r--     root   system
m   4103 0x670904fd --rw-r--r--     root   system
m   4104 0x680904fd --rw-r--r--     root   system
m      9 0x78050203 --rw-rw-rw-     root   system
m     11 0x0d06e045 --rw-rw-rw-     root   system
```

After the installation/migration of the system and the nodes, the host_responds window of spmon remains grey. In this case it is pretty certain that you did not reboot the Control Workstation after the installation/migration. There are some problems with performing a synchronization on IPCS, and in case of a large configuration this error may occur. To correct this problem, reboot your Control Workstation.

1.4.1.4 host_responds (after Repartitioning)

We highly recommend that you reboot all affected nodes after migrating and repartitioning the system.

If you did not reboot the nodes, and they run a PSSP version below 2.2, the host_responds window of spmon for these nodes turns to red. The host responds daemon (hrd) on the Control Workstation will run in a compatibility mode to be able to communicate with the nodes running PSSP 1.2 or 2.1. If you did not reboot the nodes after repartitioning the system, then refresh the hb daemon on the nodes; the spmon host_responds window will turn to green for these nodes.

Chapter 2. Installation and Customization

The installation and customization process of an SP system can be divided into three stages:

1. Initial setup

   This stage refers to the initial steps needed to install and configure the Control Workstation. We must install AIX and any prerequisite for PSSP. Then we have to set up the Control Workstation in order to prepare it for installing PSSP.

   After the PSSP code has been installed, we need to acquire the frame information, set up the environment information, and populate the SDR with IP addresses, hostnames, and details about AIX and PSSP levels we want to install in the nodes.

   Finally, we are ready to run setup_server for the first time, which is the second stage of the process.

2. setup_server and friends

   When setup_server runs for the first time, it installs and creates all the necessary file sets and resources to make the Control Workstation a boot/install server (BIS). NIM resources such as the SPOT, lppsource, and mksysb are created and allocated to the nodes. All the files needed to install the nodes are generated and the Control Workstation is prepared to respond to the network boot request coming from the nodes.

   This network boot is the third and last stage of the install process.

3. Network installation and customization

   This is the only supported way to install SP nodes. The nodes must boot from the network, and be installed by the boot/install server. After AIX has been installed, the customization process takes place, in which PSSP code is installed and configured on the node.

This chapter deals with each of these three stages, pointing out problems and proposing solutions. It contains many examples and procedures that will help you to understand how to approach and solve installation and customization problems.

2.1 Initial Setup

One of the first steps prior to installing the Control Workstation is to verify software and hardware prerequisites:

- Verify the network and TTY connection
- Download the install images to the Control Workstation

And after installation of PSSP 2.2 on the Control Workstation, do the following:

- Enter data in Kerberos
- Execute the install_cw script
- Start system partition-sensitive subsystems (syspar_ctrl)
- Enter data in the System Data Repository (SDR)

- Verify PSSP 2.2 basic functions

The *IBM RS/6000 SP Installation and Migration Guide*, GC23-3898, provides the necessary information for a successful installation. However, problems often arise during the installation and customization process. The following section provides additional information and hints for some of the main installation steps, along with suggestions for a successful installation. It also provides some good and known problem examples that will help you understand how to approach, diagnose, and solve an installation problem.

2.1.1 Network and TTY Connection

The Control Workstation and all nodes are connected via the SP Ethernet and a serial cable (RS-232C). The Control Workstation monitors the status of each node through the serial link. The SP Ethernet is used for network installation, remote command execution, subsystem communication, and node monitoring, among other tasks.

The serial link is also used to send a sequence for power on/off to each node. These two communication lines are the basis for the SP operation.

To make sure they work properly, you should check at least the following prerequisites:

- Hardware connections for the Control Workstation, frames and nodes are established properly.

  All of these should be verified by a hardware support representative. But sometimes, in particular when the Control Workstation has more than one Ethernet card, select en0 as your default SP Ethernet adapter. Although any Ethernet adapter is supported as part of the SP Ethernet network, many problems have been seen when the default adapter used is other than en0.

- The latest PTFs for TTY and Ethernet are applied.

  The TTY connection is used heavily by the hardware monitor daemon, hardmon. Without PTFs, the TTY connection might hang. The first Ethernet driver for AIX V4.1 caused the NFS daemon to hang. A later version is recommended for the following file sets:

  – For AIX V4.1

bos.rte.tty.4.1.4.12	(base TTY support and commands)
devices.mca.8ef2.com.4.1.4.1	(common integrated Ethernet software)

  – For AIX V4.2

bos.rte.tty.4.2.0.8	(base TTY support and commands)

    Please contact your local Software Service Center or use FixDist to get the latest PTF for these file sets.

- Preparation for hostname resolution is completed.

  All hostnames of network interfaces for the nodes and Control Workstation must be resolved by /etc/hosts or Domain Name Server (DNS). If the system has more than one partition, all aliased addresses and hostnames for the Control Workstation must also be resolved.

- File sets for NFS are installed.

  bos.net.nfs.server and bos.net.nfs.client are needed.

2.1.2 Disk Space and Installation Images

To support different AIX and PSSP versions, the directory structures for the installation images of PSSP 2.2 and PSSP 2.1 differ slightly. The following figure shows the directory structure of PSSP 2.2 under /spdata/sys1/install:

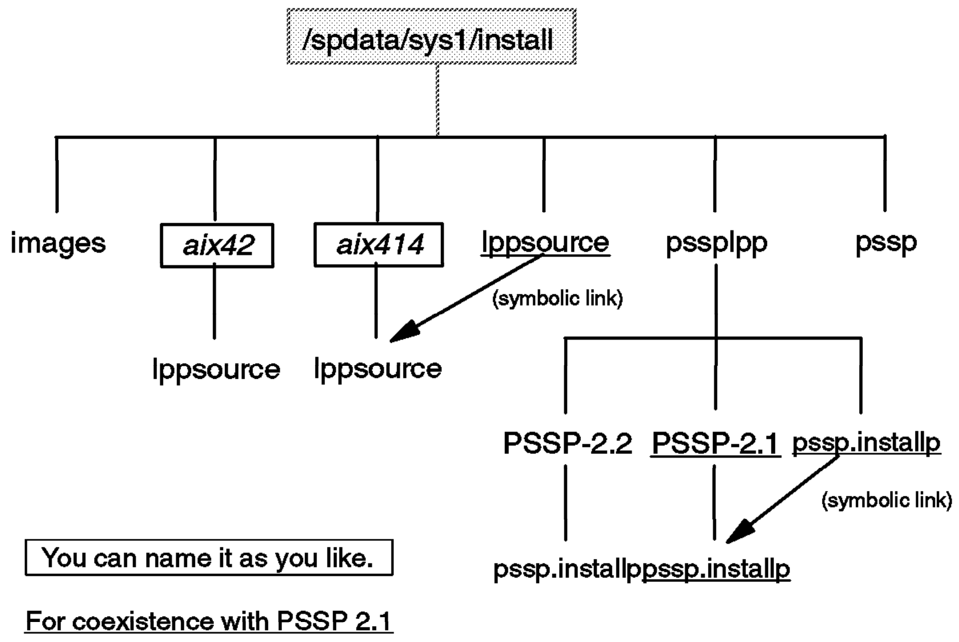


Figure 17. Structure of the Install Directory

The following is a minimal checklist for disk space and images:

- The /spdata file system structure and free space.

  See Figure 17. For free space, see *IBM RS/6000 SP Installation and Migration Guide*, GC23-3898.

- The /var and /tmp file systems have enough free space.

  /var needs at least 20MB of free space. If no problem occurs, most of it is not used. But when a problem does occur, especially one having to do with the Switch, a lot of error information is logged here. Therefore, a large amount of free space is recommended.

  /tmp needs at least 16MB of free space.

- /tftpboot needs 25MB per AIX version. That is, if you are going to install AIX Version 4.1 and AIX Version 4.2 on nodes, you need 50MB of free space in the /tftpboot directory.

- The lppsource directory contains all AIX file set images, necessary PTFs, and the perfagent.server file set.

  PTFs for AIX in the lppsource directory are automatically applied when node installation is done. To avoid PTF failures due to missing prerequisites and to make software maintenance easier, it is recommended that all AIX file set images be downloaded to the lppsource directory. To check necessary PTFs, see *Read This First*, which is shipped with the PSSP code. Whenever the contents of the lppsource directory are changed, execute `inutoc` to refresh the .toc file.

The perfagent.server file is a prerequisite for ssp.ha, which is the basis for High Availability Infrastructure.

- The latest PTFs for PSSP 2.2 are applied.

  To check the latest version of PTFs, contact your local Software Service Center. Or, search for it with FixDist.

  Note

  PTFs for PSSP are not applied automatically to nodes even if you put them in the pssplpp directory. To install them, use the script.cust script.

- The mksysb image for node installation does not contain NIM master file sets and the /etc/niminfo.prev file.

  For general RS/6000 (not SP), the mksysb image cannot be used for cloning one machine to another. Because device drivers are installed selectively, the image cannot be applied to all platforms. In other words, for cloning purposes, all device drivers should be contained in the image.

  But for SP nodes, the mksysb image for SP does not have to include device drivers. When nodes are installed, the missing device drivers are installed from the /spdata/sys1/install/<lppsource_name>/lppsource directory by the Network Install Manager (NIM). So, the directory should contain all the device driver file sets needed to configure all the devices present on the nodes.

  If the mksysb image contains the bos.sysmgt.nim.master file set, and there also is the /etc/niminfo.prev file, NIM customization will fail. Because the information in /etc/niminfo.prev is used for NIM customization, the Control Workstation is not recognized as a NIM master.

2.1.3 Kerberos

The Control Workstation is a single control point for all SP nodes. All commands to maintain nodes should be issued from the Control Workstation, which is a kind of client-server programming model. A client process requests execution of some command on a remote machine, and a server process accepts it.

On UNIX, the rsh client program and the rshd server program are used for this. All users in the .rhosts file on the server are permitted to execute commands from a remote host. But if this method is used on SP, it becomes a security exposure.

On the SP, the user that requests to execute commands remotely needs to be authorized by a trusted third party. Kerberos is used for this. The root user should be registered in the Kerberos database as an administrative user via the `setup_authent` script, which configures the Kerberos database and run time environment. The following are the main tasks of `setup_authent`:

- Create /etc/krb.conf
- Create /etc/krb.realms
- Create the principal.pag and principal.dir files in /var/kerberos/database

  These are both Kerberos database files. The keys and all principal names are kept there.

- Create /.k
- Update /etc/inittab and start the Kerberos daemons

  The following two entries are added to /etc/inittab and executed:

```
kadm:2:respawn:/usr/lpp/ssp/kerberos/etc/kadmind -n
kerb:2:respawn:/usr/lpp/ssp/kerberos/etc/kerberos
```

- Create /tmp/tkt0
- Create /.klogin
- Create /etc/krb-srvtab

The check points for Kerberos are as follows:

- The kadmin and kerberos daemons are running.

  To maintain the Kerberos database and get an authentication ticket from Kerberos, both daemons need to be running.

- `klist` shows correct information.

  `klist` shows the following output, for example:

```
# klist
Ticket file:    /tmp/tkt0
Principal:      root.admin@MSC.ITSO.IBM.COM

  Issued           Expires         Principal
Jan 30 14:43:58  Mar 1 14:43:58  krbtgt.MSC.ITSO.IBM.COM@MSC.ITSO.IBM.COM
```

  If `klist` does not show proper data, run the `kinit` command again to get the ticket.

```
# kinit root.admin
kerberos Initialization for "root.admin"
Password:
#
```

  Then re-execute `klist` to verify. If it still does not get the ticket, issue the `setup_authent` command again. Before node installation, re-execution of this command does not cause problems.

  Once the ticket has been obtained, it is cached in the /tmp/tkt0 file. When the file is removed, `kinit` should be executed again to get a new ticket.

2.1.4 The install_cw Command

After execution of this command, PSSP 2.2 functions are configured on the Control Workstation. The following are the main tasks of `install_cw`:

- Check that the Kerberos configuration files already exist.
- Define node number information for the SP ODM object.

  The definition can be displayed via the `odmget` command, as follows:

```
# odmget -q "name=sp" CuAt
CuAt:
        name = "sp"
        attribute = "node_number"
        value = "0"
        type = "R"
        generic = "DU"
        rep = "s"
        nls_index = 24
#
```

- Execute the /usr/lpp/ssp/install/bin/post_process.

  This script is responsible for many tasks. The main ones are the following:

  - Create the /spdata/sys1/spmon/hmacls file.

    It contains permission specifications for users to execute various hardware operations. It is used by the hardware monitor daemon, hardmon.

  - Add SDR and hardware monitor subsystems to System Resource Controller (SRC) and /etc/inittab, and start them.

    The following three entries are added to inittab:

    ```
    sdrd:2:once:/usr/bin/startsrc -g sdr
    hardmon:2:once:/usr/bin/startsrc -s hardmon
    sp:2:wait:/etc/rc.sp > /dev/console 2>&1
    ```

    The first two lines are necessary to accomplish single point of control of nodes from the Control Workstation. The hardmon daemon monitors node hardware status through the serial link. For SDR information, see 2.1.6, “System Data Repository” on page 77 and 2.1.11, “Case Study 2 – spbootins and SDR” on page 80.

    The last line is the rc.sp script. Its main task is customization of the machine. When executed, it decides whether customization is needed by querying the System Data Repository. If it is needed, it executes the pssp_script script for customization.

  - Create the /etc/SDR_dest_info file.

    It provides connection addresses for SDR clients on the Control Workstation. See 2.1.6, “System Data Repository” on page 77 for more details.

  - Locate clock configuration files for the switch in the /etc/SP directory.

    One of these is selected for switch clock source setting during installation.

  - Execute the `SDR_init` command.

    Via this command, all necessary classes are defined in the SDR, after which you can store configuration information through SMIT or a variety of PSSP commands.

- Set the authent_server attribute in the SP class of the SDR.

  The `SDRGetObjects` command gets the information:

```
# SDRGetObjects SP authent_server
authent_server
ssp
#
```

The value of the attribute is ssp, if Kerberos from PSSP 2.2 is used as authentication server.

- Set up the spmon logging daemon, splogd.

  This daemon calls the shell script SDR_config when the hardware type is changed. The /spdata/sys1/spmon/hwevents file contains the definition for the daemon, as follows:

  ```
  frame slot variable operator value time function
  ......
  *     *    type     =        *     b    /usr/lpp/ssp/install/bin/SDR_config -l
  ......
  ```

  This line means that whenever the hardware type value of Frame/Node/Switch is changed, SDR_config is executed by splogd. The hardware type can be obtained by the following commands:

  For frame:

  ```
  # spmon -G -l /SP/frame/*/type/value
  ```

  For nodes:

  ```
  # spmon -G -l /SP/frame/*/*/type/value
  ```

  SDR_config uses these commands internally. See 2.1.11, “Case Study 2 – spbootins and SDR” on page 80 for details.

To verify the install_cw procedure, see 2.1.7, “Verification of PSSP 2.2 Basic Functions” on page 77.

2.1.5 The syspar_ctrl Command

PSSP 2.2 provides a new command, syspar_ctrl, that manages all of the partition-sensitive subsystems. For example, syspar_ctrl -A adds all necessary daemons to the System Resource Controller (SRC) and starts them.

On the Control Workstation, syspar_ctrl -A executes all of the following commands:

```
/usr/lpp/ssp/bin/hatsctrl -a -s         # for hatsd
/usr/lpp/ssp/bin/hbctrl -a -s           # for hbd
/usr/lpp/ssp/bin/hagsctrl -a -s         # for hagd
/usr/lpp/ssp/bin/haemctrl -a -s         # for haemd
/usr/lpp/ssp/bin/hrctrl -a -s           # for hrd
/usr/lpp/ssp/bin/pmanctrl -a -s         # for pman and pmanrmd
/usr/lpp/ssp/bin/emonctrl -a -s         # for Emonitor
/usr/lpp/ssp/bin/sp_configdctrl -a -s   # for sp_configd
/usr/lpp/ssp/bin/emconditionctrl -a -s  # for loading SDR data for EM
/usr/lpp/ptpe/bin/spdmdctrl -a -s       # for spdmd
```

To get status information about the subsystems, use the following command:

```
# syspar_ctrl -G -E
Syspar Controller-managed subsystems and control scripts:
  hats  /usr/lpp/ssp/bin/hatsctrl
  hb  /usr/lpp/ssp/bin/hbctrl
  hags  /usr/lpp/ssp/bin/hagsctrl
  haem  /usr/lpp/ssp/bin/haemctrl
  hr  /usr/lpp/ssp/bin/hrctrl
  pman  /usr/lpp/ssp/bin/pmanctrl
  emon  /usr/lpp/ssp/bin/emonctrl
  sp_configd  /usr/lpp/ssp/bin/sp_configdctrl
  emcond  /usr/lpp/ssp/bin/emconditionctrl
Invalid Entry: spdmd  /usr/lpp/ptpe/bin/spdmdctrl
Examine Failed, the subsystem control script either
  does not exist or cannot be read or executed.
#
```

Note

Unless Performance Toolbox Parallel Extensions (PTPE) for AIX is installed, `spdmdctrl` does not exist. PTPE is a feature of PSSP 2.2. To use PTPE, AIX Performance Toolbox V2.2 is needed.

Each of these commands is a shell script. They use standard AIX commands to start daemons and processes. For example, `hatsctrl -a -s` uses the `mkssys` command to add the subsystem definition to the SRC.

```
mkssys -s hats -a <syspar_ip_addr> -p /usr/lpp/ssp/bin/hats -u 0
      -o /var/ha/hats.<syspar_name> -e /var/ha/hats.<syspar_name>
      -R -Q -K -d -w 30 -G hats
```

After that, `mkitab` is used to bring up the daemon automatically when the system boots:

```
mkitab "hats:2:once:/usr/bin/startsrc -g hats > /dev/console 2>&1"
```

As seen in the example, once all options are registered in the SRC, the `startsrc` command starts the process properly, even if options are not specified each time, because `mkssys` saves the defined information in the SRCsubsys ODM class.

```
# export ODMDIR=/etc/objrepos
# odmget -q "subsysname=hats.sp2en0" SRCsubsys
SRCsubsys:
    subsysname = "hats.sp2en0"
    synonym = ""
    cmdargs = "192.168.3.37"
    path = "/usr/lpp/ssp/bin/hats"
    uid = 0
    auditid = 0
    standin = "/dev/console"
    standout = "/var/ha/log/hats.sp2en0"
    standerr = "/var/ha/log/hats.sp2en0"
    action = 1
    multi = 0
    contact = 3
    svrkey = 0
    svrmtype = 0
    priority = 20
    signorm = 0
    sigforce = 0
    display = 1
    waittime = 30
    grpname = "hats"
#
```

The stopsrc command is used to stop each subsystem, and lssrc is used to list the current status of each subsystem. The -l option provides more detailed information about each subsystem.

```
# lssrc -a
    ...             ...              ...     ...
 haem.sp2en0      haem             20636   active
 hats.sp2en0      hats             29044   active
 syslogd          ras              45110   active
 lpd              spooler                  inoperative
 gated            tcpip                    inoperative
    ...             ...              ...      ...

# lssrc -ls hats.sp2en0
Subsystem         Group            PID     Status
 hats.sp2en0      hats             29044   active
  NODES_DEFINED = 3, IN_GROUP = 3, GROUP STATUS = Stable
  Myself = (192.168.3.37, 112 2670239), GL = (192.168.3.37, 1123434725)
  HB Interval = 1 secs HB Sensitivity = 4 missed beats
  CWS = 192.168.3.37
#
```

When some problem occurs with these subsystems, you should do the following, in addition to the basic checks we have just described:

- Check the status of each subsystem.

  When syspar_ctrl -A -G succeeds, the following entries are added to inittab:

  ```
  hats:2:once:/usr/bin/startsrc -g hats > /dev/console 2>&1
  hb:2:once:/usr/bin/startsrc -g hb > /dev/console 2>&1
  hr:2:once:/usr/bin/startsrc -g hr
  hags:2:once:/usr/bin/startsrc -g hags > /dev/console 2>&1
  haem:2:once:/usr/bin/startsrc -g haem > /dev/console 2>&1
  pman:2:once:/usr/bin/startsrc -g pman >/dev/console 2>&1
  sp_configd:2:once:/usr/bin/startsrc -s sp_configd
  ```

And the hb, hr, hags, pman, haem, hats, emon group subsystems are added to the SRC. The following is a partial output of the `lssrc -a` command:

```
hb.sp2en0        hb              22154   active
hr.sp2en0        hr              23388   active
hags.sp2en0      hags            24862   active
hagsglsm.sp2en0  hags            25654   active
pman.sp2en0      pman            21288   active
pmanrm.sp2en0    pman            13382   active
haem.sp2en0      haem            20636   active
hats.sp2en0      hats            29044   active
Emonitor.sp2en0  emon                    inoperative
```

Emonitor (Switch monitoring daemon to maximize availability) is not executed automatically until the operator executes `Estart -m` from the command line.

Except for Emonitor, all subsystems should be active.

If one or more of the subsystems are not listed in the output of `lssrc -a`, re-execute the `syspar_ctrl -G -A` command and examine the error message. If there is an inoperative subsystem, besides Emonitor, its log files should contain some error information.

- Check the log file.

  The log files for HA subsystems are located in the /var/ha/log directory. The log files for the PMAN subsystem are located in the /var/adm/SPlogs/pman directory. For more details on the PMAN subsystem, see Chapter 4, "Problem Management" on page 187.

- Check shared memory.

  If the shared segment is messed up by some other process, the hats and haem subsystems cannot be executed because of failure of shared segment allocation. The `ipcs -mp` command shows the status of it. If there is no significant process except SP daemons, the following is a typical output of this command:

```
# ipcs -mp
IPC status from /dev/mem as of Thu Feb  6 10:40:34 EST 1997
T     ID     KEY        MODE       OWNER    GROUP  CPID  LPID
Shared Memory:
m      0 0x5806a933 --rw-rw-rw-     root   system 16606  2242
m   4097 0x670904be --rw-r--r--     root   system 29302 37172
m   4098 0x680904be --rw-r--r--     root   system 29302 37172
m      9 0x78050203 --rw-rw-rw-     root   system 21948 50558
m   8202 0x780904c4 --rw-rw-rw-     root   system 50558 62694
m     11 0x0d06e045 --rw-rw-rw-     root   system 39510 30370
```

CPID is the process ID for the creator of the shared memory entry. One hatsd creates two shared memories. If you use the system partitioned by two, in other words, two hatsd daemons are running on the Control Workstation, there are four shared memories for it. In this example, the entries that have a value starting with `0x6` in the `KEY` field are for hatsd. On the other hand, haemd uses only two shared memories even if the system is partitioned into more. The entries that have a value starting with `0x78` in the `KEY` field are for the haem subsystem.

2.1.6 System Data Repository

The SDR holds all the system parameters that are needed to install, customize, and maintain SP nodes. All data is located in the /spdata/sys1/sdr directory on the Control Workstation. All nodes can access that data, by mediation of the SDR server daemon (sdrd) on the Control Workstation.

A minimal check list for SDR is as follows:

- The SDR daemon, sdrd, is active.

  The `lssrc -a` command shows the status of sdrd. If it is not there, `install_cw` should be executed again.

- The /etc/SDR_dest_info file contains correct information about the SDR server.

  The entries in this file are as follows:

  ```
  default:192.168.3.37
  primary:192.168.3.37
  ```

  These two entries define the IP address for the server in the active partition (primary), and the IP address of the server in the default partition (default). The primary and default entries are always the same in the Control Workstation; however, on the nodes, the primary entry should point to the IP address of the IP alias defined for the partition.

For more information, refer to 2.1.11, "Case Study 2 – spbootins and SDR" on page 80.

2.1.7 Verification of PSSP 2.2 Basic Functions

Some verification scripts are provided with PSSP 2.2 to check if the critical subsystems are being installed and configured properly:

- `SDR_test`

  This command temporarily creates an SDR Class and adds attributes and values to it. Afterwards, it deletes them all. If this command succeeds, there is no problem with SDR operations.

- `spmon_itest`

  This command verifies the following:

  – hardmon runs.

  – sdrd runs.

  – The hmacls file exists and contains the right hostname.

  – SP_ports Class in SDR has information about hardmon.

  If this command succeeds, prerequisite conditions for node status monitoring and entering data to SDR are verified.

- `spmon_ctest`

  This command verifies the following:

  – hardmon runs.

  – sdrd runs.

  – Node Class in SDR exists and can be read.

  - Frame Class in SDR exists and can be read.
  - The spmon command gets hardware information about frame.
  - The hmacls file contains right permission to monitor the frame.

  If this command succeeds, the hardware monitoring function works properly.

- SYSMAN_test

  This command should be used after setup_server has been successfully executed. It verifies SP system management components. The main verification points on the Control Workstation are as follows:

  - The log directory /var/adm/SPlogs exists.
  - rc.sp is registered in inittab.
  - Configurations for kerberized rsh commands are correct.
  - The SMIT menu definition for SP is saved in the ODM.
  - The /etc/SP directory exists.

    This directory contains the definition files for the switch.
  - The /spdata/sys1/install/images directory exists.

    This directory contains the mksysb install image for nodes.
  - The IP address of the Control Workstation in the SDR is correct.

    It is the cw_ipaddrs attribute in the SP SDR class.
  - NFS is available.
  - TFTP service is available.
  - BOOTP service is available.
  - The /tftpboot directory contains the necessary files for NIM installation.
  - Mount access to the images directory is enabled for the nodes.
  - SP system administration tools are configured correctly.

    It verifies NTP, AMD, and File Collection.

Note

When SYSMAN_test is executed on the Control Workstation, it tries to do nodes configuration verification automatically. Naturally, it fails if nodes are not installed yet or not powered on.

2.1.8 For Successful Installation

Because it is the very first installation step, problems are likely to be caused by defective hardware or software. You should pay special attention to PTF information.

To get the latest PTF information, contact your local Software Support Center. It can also be found in the RS/6000 SP Support page on the World Wide Web:

```
http://www.rs6000.ibm.com/support/sp/
```

It can also be found in the SP forum on VM. To access it, execute the following command on your VM:

```
TOOLS SENDTO YKTVMH3 IBMUNIX IBMUNIX GET SP2 FORUM
```

You can also use the FixDist utility to get PTF information, and the PTF itself. See Appendix B, “How to Get FixDist” on page 275 for more details.

2.1.9 Case Study 1 – syspar_ctrl

The hats daemon cannot continue to run on the Control Workstation. The command `syspar_ctrl -s hats` can activate the subsystem, but it dies in a few seconds, as follows:

```
# lssrc -s hats.sp2en0
Subsystem         Group            PID     Status
 hats.sp2en0      hats                     inoperative
#
```

2.1.10 Problem Determination

The log file for the daemon is located in the /var/ha/log directory, which contains all the log files about HA daemons. To get the latest log entry about the hats subsystem, enter the following command:

```
# cd /var/ha/logs
# ls -lt | grep hats
-rw-rw-rw-  1 root   system     161 Feb 05 10:40 hats.05.104015.sp2en0
-rw-rw-rw-  1 root   system     161 Feb 05 10:40 hats.05.103815.sp2en0
......
#
```

By using the `-t` flag, the `ls` command lists the files ordered by time. The first file is the latest log file. The suffix means <date-of-month>.<hh-mm-ss>.<syspar-name>.

```
# cat hats.05.104015.sp2en0
02/05  10:40:15 hatsd[0]: High Availability Topology Services Daemon.
02/05  10:40:15 hatsd[0]: 2523-012 Unable to obtain socket port number from
/etc/services
#
```

According to the log, the cause of the problem is the definition in the /etc/services file. This file defines which network service uses which port number, for socket programs.

When `syspar_ctrl -G -A` is executed on the Control Workstation, it gets the port number information for each daemon from SDR and enters it in /etc/services. Therefore, even when `syspar_ctrl -G -A` is executed on each node, the same port number will be set in /etc/services, based on the SDR information.

The port number information is stored in the SDR when `syspar_ctrl -G -A` is executed for the first time. The Syspar_ports SDR class keeps the information.

```
# SDRGetObjects Syspar_ports
subsystem    port
hats              10000
hags              10001
haem              10002
#
```

The port number for hats.sp2en0 is 10000. The /etc/services file should contain the following line:

```
hats.sp2en0     10000/udp
```

In this case, this line was deleted by mistake. When the line was added manually, the daemon worked fine.

You can also delete and add hats subsystem definitions by using commands. This updates the /etc/services file.

```
# syspar_ctrl -D hats
# syspar_ctrl -A hats
```

-D This option means stop and delete. It deletes the definition for hats from the SRC, /etc/inittab, and /etc/services files.

-A This option means add and start.

Instead of `syspar_ctrl`, you can use `hatsctrl` for controling the hats daemon, as follows:

```
# hatsctrl -k
# hatsctrl -d
# hatsctrl -a
# hatsctrl -s
```

where -k stands for kill, -d for delete, -a for add, and -s for start.

2.1.11 Case Study 2 – spbootins and SDR

The `spbootins` command was executed from `smitty` with the following settings:

```
             Boot/Install/usr Server Information

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                              Entry Fields
 Start Frame                                  1
 Start Slot                                   1
 Node Count                                   16

 OR

 Node Group

 OR

 Node List

 Boot/Install Server Node Identifier           0
 Network Install Image Name                    bos.obj.ssp.42
 Destination Hard Disk(s)                      hdisk0
 Response from Server to bootp Request         install
 LPP Source Name                               aix42
 PSSP Level                                    PSSP-2.2

 /usr Server's Hostname or IP Address
 Gateway to /usr Server
 /usr Client Adapter Name

 Run setup_server on the Control Workstation?  no
```

These settings are equivalent to the following command:

```
spbootins -n 0 -i bos.obj.ssp.42 -h hdisk0 -r install -v aix42
          -p PSSP-2.2 -s no 1 1 16
```

Then the following error occurred:

```
                                COMMAND STATUS

Command: failed        stdout: yes           stderr: no
spbootins:  0022-046 Node to be changed - 14 - could not be found in
repository.
```

2.1.12 Problem Determination

This error message shows that node information is missing in the SDR. If the system has more than one partition, the SP_NAME environment variable should be checked. But if the system has only one partition system, the spmon -d command provides basic diagnostics information.

```
# spmon -d
1.  Checking server process
    Process 18750 has accumulated 4 minutes and 54 seconds.
    Check ok

2.  Opening connection to server
    Connection opened
    Check ok

3.  Querying frame(s)
    1 frame(s)
    Check ok

4.  Checking frames

       Controller  Slot 17 Switch  Switch    Power supplies
Frame  Responds    Switch  Power   Clocking  A   B   C   D
------------------------------------------------------------
  1      yes        yes     on        0      N/A N/A N/A N/A

5.  Checking nodes

-------------------------- Frame 1 ------------------------------------
Frame Node   Node       Host     Switch   Key    Env   Front Panel
Slot  Number Type Power Responds Responds Switch Fail     LEDs
-------------------------------------------------------------------------------
  1     1    thin   on    yes      yes    normal  no  LEDs are blank
  2     2    thin   on    yes      yes    normal  no  LEDs are blank
  3     3    thin   on    yes      yes    normal  no  LEDs are blank
  4     4    thin   on    yes      yes    normal  no  LEDs are blank
  5     5    thin   on    yes      yes    normal  no  LEDs are blank
  6     6    thin   on    yes      yes    normal  no  LEDs are blank
  7     7    thin   on    yes      yes    normal  no  LEDs are blank
  8     8    thin   on    yes      yes    normal  no  LEDs are blank
  9     9    thin   on    yes      yes    normal  no  LEDs are blank
 10    10    thin   on    yes      yes    normal  no  LEDs are blank
 11    11    thin   on    yes      yes    normal  no  LEDs are blank
 12    12    thin   on    yes      yes    normal  no  LEDs are blank
 13    13    thin   on    yes      yes    normal  no  LEDs are blank
 15    15    thin   on    yes      yes    normal  no  LEDs are blank
 16    16    thin   on    yes      yes    normal  no  LEDs are blank
#
```

The entry for Node 14 is missing in this output. It shows that the node is not known to PSSP 2.2. The SDR information should be checked to verify its consistency.

The SDR database is located under the /spdata/sys1/sdr directory. The Node class definition is located in the /spdata/sys1/sdr/partitions/<IP address>/Node file.

```
# cat /spdata/sys1/sdr/partitions/192.168.3.37/classes/Node
1=1 2=10005AFA18CF 3=1 4=1 5=1 6=0 7=3 8=5 9=1 10=sp2n01 .....
1=2 2=10005AFA121D 3=1 4=2 5=1 6=1 7=2 8=5 9=1 10=sp2n02 .....
1=3 2=10005AFA082C 3=1 4=3 5=1 6=2 7=0 8=6 9=1 10=sp2n03 .....
......
#
```

The specified values correspond to the attribute names defined by the file in the following directory:

```
# cat /spdata/sys1/sdr/defs/Node
pI1=node_number S2=hdw_enet_addr I3=frame_number I4=slot_number ......
......
```

Although SDR data is a text file, it should not be edited by an editor. All read/write requests need to be handled by the SDR daemon. For that purpose, PSSP 2.2 provides commands that have the string SDR as prefix.

```
# SDRGetObjects Node node_number==1 hdw_enet_addr frame_number slot_....
hdw_enet_addr frame_number slot_number
10005AFA18CF            1            1
#
```

All SDR commands get the IP address for the SDR daemon from the /etc/SDR_dest_info file. See 2.1.6, “System Data Repository” on page 77 for more information about this file.

Basic node information is stored in the Node class in the SDR automatically when frame information is stored in it, that is, when the frame information is stored in the SDR by `smitty frame_dialog`, and the re-initialization of SDR flag is on.

```
                       Frame Information

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                Entry Fields
* Start Frame                                   1
* Frame Count                                   1
* Starting Frame TTY port                       /dev/tty0
  Re-initialize the System Data Repository      yes
```

Re-initialization is accomplished via the `hmreinit` command. This command restarts hardmon and executes the `SDR_config` command to update the SDR. It saves the current hardware type information in the /var/adm/SPlogs/SPconfig/node_info file, as follows:

```
# cat /var/adm/SPlogs/SPconfig/node_info
/SP/frame/frame1/node1/type/value/97
/SP/frame/frame1/node2/type/value/97
/SP/frame/frame1/node3/type/value/97
/SP/frame/frame1/node4/type/value/97
/SP/frame/frame1/node5/type/value/97
/SP/frame/frame1/node6/type/value/97
/SP/frame/frame1/node7/type/value/97
/SP/frame/frame1/node8/type/value/97
/SP/frame/frame1/node9/type/value/97
/SP/frame/frame1/node10/type/value/97
/SP/frame/frame1/node11/type/value/97
/SP/frame/frame1/node12/type/value/97
/SP/frame/frame1/node13/type/value/97
/SP/frame/frame1/node15/type/value/97
/SP/frame/frame1/node16/type/value/97
/SP/frame/frame1/switch17/type/value/49
#
```

The numbers indicate the device type. You may find all the definitions in the SDR_config script as comments.

```
#Array that is indexed in with node type value and returns slots used
%slots_used=(33,  1,    #Envoy Node Supervisor ′21′x
                         65,  1,        #Very old nodes ′41′x
                         81,  2,        #Papago wide nodes (4.0v) ′51′x
                         82,  2,        #Papago wide nodes (4.0v) ′52′x
                         83,  2,        #Papago wide nodes (3.7v) ′53′x
                         84,  2,        #Papago wide nodes (3.7v) ′54′x
                         97,  1,        #Onieda thin nodes  ′61′x
                         98,  1,        #Onieda thin nodes  ′62′x
                         113, 2,        #Courier wide node (4.0v) ′71′x
                         115, 2,        #Courier wide node (4.0v) ′73′x
                         161, 4,        #R30/40 MP node ′A1′x
                         49, switch,    #switch
                         50, switch,    #PPP switch
                         129, switch);  #TBS switch
```

There is no information about Node 14 in the node_info file. `splstdata -n` shows the information about nodes in the SDR.

```
# splstdata -n
                  List Node Configuration Information

node# frame# slot# slots initial_hostname reliable_hostname default_route
      processor_type processors_installed
-------------------------------------------------------------------------
    1      1     1     1 sp2n01           sp2n01.msc.itso.   192.168.3.37
                  UP                    1
    2      1     2     1 sp2n02           sp2n02.msc.itso.   192.168.3.37
                  UP                    1
    3      1     3     1 sp2n03           sp2n03.msc.itso.   192.168.3.37
                  UP                    1
    4      1     4     1 sp2n04           sp2n04.msc.itso.   192.168.3.37
                  UP                    1
    5      1     5     1 sp2n05           sp2n05.msc.itso.   192.168.3.37
                  UP                    1
    6      1     6     1 sp2n06           sp2n06.msc.itso.   192.168.3.37
                  UP                    1
    7      1     7     1 sp2n07           sp2n07.msc.itso.   192.168.3.37
                  UP                    1
    8      1     8     1 sp2n08           sp2n08.msc.itso.   192.168.3.37
                  UP                    1
    9      1     9     1 sp2n09           sp2n09.msc.itso.   192.168.3.37
                  UP                    1
   10      1    10     1 sp2n10           sp2n10.msc.itso.   192.168.3.37
                  UP                    1
   11      1    11     1 sp2n11           sp2n11.msc.itso.   192.168.3.37
                  UP                    1
   12      1    12     1 sp2n12           sp2n12.msc.itso.   192.168.3.37
                  UP                    1
   13      1    13     1 sp2n13           sp2n13.msc.itso.   192.168.3.37
                  UP                    1
   15      1    15     1 sp2n15           sp2n15.msc.itso.   192.168.3.37
                  UP                    1
   16      1    16     1 sp2n16           sp2n16.msc.itso.   192.168.3.37
                  UP                    1
#
```

Node 14 information is also missing in this output. The serial connection for the node should be checked. If it is not loose, there may be a hardware problem with the node supervisor card. You should contact your local hardware support center.

2.1.13 Other Considerations

- The `installp` command

  The `inutoc` command produces a .toc file that is used by the `installp` command. The version of `installp` that reads the .toc file for installation should be the same version that created the .toc file. It should be considered when you copy AIX OS install images from another machine to the Control Workstation.

  One particular problem is present in the first version of the `installp` command for AIX 4.1.4.; it seems to have a defect when handling .toc files. The bos.rte.install.4.1.4.1 or later file set should be installed to avoid this known problem.

- Locale

  There are some locale-dependent programs in PSSP 2.2. You should use the C or English locale during installation.

- The `setup_server` command

  It assumes that the return values of commands are in C or English locale format. If `setup_server` is executed on other locales, it fails.

- The `spmon -g` command

  The label strings of each window are retrieved from the message catalogue of the C locale. When it is executed on other locales, it fails.

2.2 Problems with setup_server and Friends

This section covers problem determination related to problems of the setup_server script, which can originate in many parts of the configuration. It is useful to perform a sort of check list before you begin with problem determination. You could save time by finding your problems already in the check list.

2.2.1 Checklist Before You Start

1. The installation/migration of the AIX operating system has completed successfully.
2. Check the sizes of the file systems. Recommended sizes are:

   rootvg 2GB

   spdatavg 2GB

   The space required for /tftpboot depends on the number of AIX versions you are running on the SP. If you are using only one AIX version, let us say AIX 4.2, you need about 25MB of space for /tftpboot. If you run AIX 4.2 and AIX 4.1.4, you need twice as much space, that is, about 50MB. Check also if enough space is left in all file systems. For more information about settings of the file systems, see *RS/6000 SP: Installation and Migration Guide*, GC23-3898.
3. The environment paths for the administrator and other users have been properly set (that is, .profile and .kshrc).
4. The network connections work properly, that is, the hostnames, IP addresses, and netmasks have been correctly defined. The `ping` command works for every IP address of the running systems, and the hostname resolution for the nodes that are going to be installed works properly. (Consider if you are using Domain Name Server (DNS).)
5. The install images of each release of AIX have been placed in the proper directory, such as:

   ```
   /spdata/sys1/install/aix414/lppsource
   /spdata/sys1/install/aix415/lppsource
   /spdata/sys1/install/aix42/lppsource
   ```

   (In case the machine has been migrated from PSSP2.1 to PSSP2.2, check if a symbolic link for the old lppsource directory exists.)
6. The install images of the PSSP versions have been placed in the proper directory, such as:

   ```
   /spdata/sys1/install/pssplpp/PSSP-1.2
   /spdata/sys1/install/pssplpp/PSSP-2.1
   /spdata/sys1/install/pssplpp/PSSP-2.2
   ```

   (In case the machine has been migrated from PSSP2.1 to PSSP2.2, check if a symbolic link for pssp.installp in the "old" pssplpp directory exists.)
7. The mksysb images for AIX versions have been placed in the /spdata/sys1/install/images directory.
8. PSSP 2.2 has been installed on the Control Workstation. The prerequisite for the installation of ssp.ha and ssp.pman is the perfagent.server file set. If you

run AIX 4.1.4 or 4.1.5 on the Control Workstation, make sure that perfagent.server.2.1.4.4 has been installed on the Control Workstation and the install image of perfagent.server.2.1.4.4 exists in the corresponding lppsource directory.

If you run AIX 4.2 on the Control Workstation, make sure that perfagent.server.2.2.0.0 or later has been installed on the Control Workstation and the install image of perfagent.server.2.2.0.0 or later exists in the corresponding lppsource directory.

Table 2. Perfagent.server Versions on the Control Workstation

AIX Version	**PSSP Version**	**Perfagent.server Version**
4.1.4	2.1	2.1.4.4
	2.2	2.1.4.4
4.1.5	2.1	2.1.4.4
	2.2	2.1.4.4
4.2	2.2	2.2.0.0

Table 3. Perfagent.server Versions Running on the Node

		Node					
		AIX 4.1.4		AIX 4.1.5		AIX 4.2.0	
CWS	AIX 4.1.4	2.1.4.4	2.1.4.4	N/A	N/A	N/A	N/A
	AIX 4.1.5	2.1.4.4	2.1.4.4	2.1.4.4	2.1.4.4	N/A	N/A
	AIX 4.2.0	2.1.4.4	2.2.0.0	2.1.4.4	2.2.0.0	2.1.4.4	2.2.0.0
		PSSP 2.1	PSSP 2.2	PSSP 2.1	PSSP 2.2	PSSP 2.1	PSSP 2.2

If the checks did not pass, and you need more “how to” information, refer to *RS/6000 SP: Installation and Migration Guide*, GC23-3898, and *RS/6000 SP: System Planning Guide*, GC23-3902.

2.2.2 What Wrappers Are and How to Use Them

The design of the setup_server has changed in PSSP 2.2. The new version is divided into modular sections called wrappers. Wrappers are functions written in Perl; each is an independent script. The use of wrappers offers the following advantages for the recovery and debugging activities:

- It is easier to check input parameters, flags and output because each wrapper is an independent external command. This means that the description of the problem can be more exact.
- It is easier to point to where an error occurs. Each wrapper has its own error messages, and as soon as the problem has been located, it can be picked up from there by using the individual wrapper. It is not necessary to execute the whole setup_server script again.

Let us have a look at the flow of the setup_server script that shows us the wrappers that are available, and their main purpose:

```
General flow for setup_server       Component command equivalent

----------------------------------  -------------------------------
1 Parse command line parameters.    None
2 Get information from SDR and ODM  None
  (node number).
3 Check prerequisites.              None
4 Configure PSSP services on this   services_config
  node.
5 If CWS then perform CWS-specific  setup_CWS
  tasks.
6 Get an authentication ticket.     kinit
7 If a NIM master but not a boot/   delnimmast -l <node_number>
  install server then unconfigure
  NIM and uninstall NIM file sets.
8 If not the CWS or boot/install    None
  server then exit.
-----------------------------------------------------------------
Note: Remaining code applies to CWS and boot/install servers only.
      We only support boot/install servers which are direct NIM
      clients of the Control Workstation.  We do not support
      "second-level" boot/install servers.
-----------------------------------------------------------------
 9 If any NIM clients are no longer delnimclient -s <server_node_numb>
   boot/install clients then delete
   them from the NIM configuration
   database.
10 Make this node a NIM master.     mknimmast -l <node_number>
11 Create TFTP access and srvtab    create_krb_files
   files on this master.
12 Make NIM interfaces for this     mknimint -l <node_number>
   node.
13 Add entry for server in /.rhosts Manually edit file if necessary.
   file on CWS.
14 Make the necessary NIM resources mknimres -l <node_number>
   on this master.
15 Make NIM clients of all of this  mknimclient -l <client_node_list>
   node's boot/install clients.
16 Make the config_info files for   mkconfig
   this master's clients.
17 Make the install_info files for  mkinstall
   this master's clients.
18 Export pssplpp file system to    export_clients
   all clients of this server.
19 Allocate the necessary NIM       allnimres -l <client_node_list>
   resources to each client.
20 Remove entry for server in       Manually edit file if necessary.
   /.rhosts file on CWS, if added.
21 Remove authentication ticket.    /bin/rm /tmp/tkt_rcmd
```

Now let us have a closer look at the wrappers. In this chapter we describe the wrappers and show their general flow and logic. For more information, see Appendix E, "Flowcharts of setup_server Wrappers" on page 287, which contains the flow charts for each wrapper.

2.2.2.1 services_config

This wrapper existed in the previous version of the setup_server and remains for that reason in the directory /usr/lpp/ssp/install/bin. It reads the SDR information stored in the SP class and checks which services will be run on the nodes. Then it calls the appropriate service config scripts. Possible services to run are:

- NTP services

- Print management services
- User management services
- AMD services
- File collection services

This script is called by /etc/rc.sp to set up designated services on nodes, and by setup_server to set up services on the Control Workstation. It calls init_ssp_envs, which is defined in ssp_functions.

Command Syntax: `services_config`. This command has no input parameters.

General flow and logic for `services_config`:

1. Get the version of the PSSP code.
2. Check the site_environment configuration. If one of the following options are configured, and ssp.sysman is not installed, get it from the Control Workstation and install it:
   - ntp_config != none
   - amd_config = true
   - print_config != false
   - usermgmt_config = true
   - filecoll_config = true
3. If the switch is installed and ssp.css is not installed, get it from the Control Workstation and install it.
4. Install ssp.jm if it has not been installed yet.
5. Call config scripts for required subsystems.

2.2.2.2 setup_CWS

This script reads the information from the SDR, checks prerequisites, and makes a setup of Control Workstation-specific items.

Command Syntax: `setup_CWS [-h]`
This command has no input parameters.

General flow and logic for `setup_CWS`:

1. Parse command line parameters.
2. Get information from the SDR.
3. Investigate prerequisites and conditions:

   Make sure this is the Control Workstation.

   Make sure that the required Kerberos files exist on the system:
   - /etc/krb-srvtab
   - /etc/krb.conf
   - /etc/krb.realms
4. Set up Control Workstation-specific items.
   - Set up the environment for the authentication routines.

- If the Kerberos database is on the Control Workstation, dump the database and create a list of known rcmd principals. If there is no local database, create the list of known principals in the /etc/krb-srvtab files.
- Perform all Kerberos updates on the Control Workstation required for each SP node.
- Create/update files and directories on the Control Workstation.

2.2.2.3 delnimmast

This wrapper deletes the NIM master definition, that is, all NIM objects and the SPOT on the boot/install node or on the Control Workstation. The NIM file sets are also deinstalled.

Command Syntax: `delnimmast -l <node_number_list>`
Example: `delnimmast -l 0`
Node_number 0 signifies the Control Workstation.

General flow and logic for `delnimmast`:

1. Parse command line parameters.
2. Get information from SDR.
   For each node:
   a. Set some global variables.
   b. Check prerequisites and conditions:

      If a node is not a valid node, skip to the next node.

      If a node is not boot/install server, issue a message but continue processing.

      If a node cannot be contacted via `runitrc`, skip that node.
   c. If a node is configured as a NIM master, unconfigure it.
   d. If there are any of the NIM file sets, delete them.

2.2.2.4 delnimclient

This script deletes the NIM client definitions from the NIM master. It searches for the nodes passed from the command line, resets these nodes, deallocates the NIM resources that were allocated to them, and removes their client definitions on the NIM master.

Command Syntax:

```
delnimclient -h | -l <node_number_list> | -s <server_node_list>
```

-h Display command syntax.

-l <node_number_list> List of node numbers to remove as NIM clients. This is a required option. Node number 0 (the CWS) is not allowed.

-s <server_node_list> List of server (NIM master) nodes on which to delete all clients that are no longer defined as boot/install clients in the SDR.

Examples:

```
delnimclient -l 1-5
```

Delete the client definitions for Nodes 1,2,3,4, and 5.

```
delnimclient -s 0
```

Delete the client definitions for all nodes of server 0 (Control Workstation) that are no longer defined as its clients (in the SDR).

General flow and logic for `delnimclient`:

1. Get information from the SDR.
2. Parse command line parameters.

   If -l specified, then for each specified client node:

   a. Determine the client′s NIM master.
   b. Check prerequisites and conditions:
      - If CWS, then skip to the next node.
      - Specified node must be a valid SP node.
      - Specified node′s boot server must be a valid SP node.
      - Specified node′s boot server can be contacted via `dsh`.
      - If the client′s boot/install server is not configured as a NIM master, then issue a message (consider "job done") and skip the node.
      - Specified node is configured as a client on its boot/install server.
   c. Remove the client on the master; if an error occurs, issue a message (consider "job done") and keep going.

   Else if `-s` specified, then for each specified server node:

   a. Check prerequisites and conditions.
   b. Delete clients from NIM database which are no longer boot/install clients.

2.2.2.5 mknimmast

This wrapper creates a NIM master. To create a NIM master means that the NIM master file sets (bos.sysmgt.nim.master and bos.sysmgt.nim.spot) are installed and the system is configured as a NIM master by using the `nimconfig` command.

Command Syntax:

```
mknimmast -h | -l <node_number_list>
```

where:

-h Display command syntax.

-l <node_number_list> List of node numbers to define as NIM masters. This is a required option. Node number 0 indicates the Control Workstation.

-s <server_node_list> List of node numbers to define as NIM masters. This is a required option. Node number 0 signifies the Control Workstation

General flow and logic for `mknimmast`:

1. Parse command line parameters.
2. Get information from the SDR.
   For each node:

   a. Set some global variables.

b. Check prerequisites and conditions. Skip the node if any of the following are true:

   Node is not the Control Workstation or boot/install server.

   Node cannot be contacted via `dsh`.

   Node is already configured as a NIM master.

   The lppsource directory (from -v or default) does not exist.

c. If the NIM master file sets are not installed, then:

   Export the correct lppsource from the Control Workstation.

   Mount the lppsource directory from the Control Workstation.

   Install the NIM file sets.

   Unmount the lpp_source directory.

d. Activate the NIM master.

   Configure the NIM master on the node.

   Make the Control Workstation a NIM client of this NIM master.

2.2.2.6 create_krb_files

This wrapper creates/updates the /etc/tftpaccess.ctl file. It also creates the Kerberos definition for every node for which bootp_response is set to install, customize or migrate. If the server node is not the Control Workstation, then it creates the <hostname>-new-srvtab file on the Control Workstation, copies it to the local /tftpboot directory, and removes the file from the Control Workstation.

Whether the operation is done on the server node or the Control Workstation, the script searches for the existence of the /tftpboot/<hostname>-new-srvtab file and makes it read-only for nobody (TFTP).

Command Syntax: `create_krb_files [-h]`

-h Display command syntax.

General flow and logic for `create_krb_files`:

1. Parse command line parameters.
2. Get information from the SDR.
3. Check prerequisites.
4. Create/update the /etc/tftpaccess.ctl file on the server.
   For each client of this server:

   a. Create the srvtab file on the Control Workstation.

   b. Copy the srvtab file from the Control Workstation to the server (unless the Control Workstation is the server).

   c. If the /tftpboot/<hostname>-new-srvtab file exists, make it read-only for nobody (TFTP):
   ```
   chmod 400 /tftpboot/<hostname>-new-srvtab
   chown nobody /tftpboot/<hostname>-new-srvtab
   ```

2.2.2.7 mknimint

This wrapper creates network objects on the NIM master to serve NIM clients. If more than one NIM master is configured, the `mknimint` command creates network objects to find the Control Workstation. Why is it so important to find the Control Workstation? If there is more than one NIM master defined on the SP, the Control Workstation will remain as a resource server for the other ones. So if `mknimint` is executed on a boot/install server that is not the Control Workstation, it creates the Control Workstation as a NIM client for this boot/install server to make this server able to access the resources on the Control Workstation. That is the reason why the boot/install server (other than the Control Workstation), while executing `mknimint`, searches for all network interfaces of the Control Workstation using the `netstat` command (in fact, only Ethernet or Token Ring interfaces are retrieved), and creates NIM network objects, which are related to the interfaces of the Control Workstation. To make sure that the boot/install server can reach every network interface of the Control Workstation, the route definitions must be in place.

Command Syntax:

```
mknimint -h | -l <node_number_list>
```

where:

-h Display command syntax.

-l <node_number_list> List of node numbers to define as NIM masters. This is a required option. Node 0 is the Control Workstation.

General flow and logic for `mknimint`:

1. Parse command line parameters.
2. Get information from SDR.
   For each node:
   a. Set some global variables.
   b. Find all networks in NIM.
   c. Find all Ethernet networks on the system.
   d. If some of them are not defined in NIM yet, then:
      - Define the network to NIM.
      - Define to NIM the route to the master.
   e. If the node on which the operation is performed is not the Control Workstation, then:
      - Find all the Control Workstation interfaces via the `netstat` command.
      - Only the Ethernet and Token Ring “up” interfaces will be stored.
      - Define all the Control Workstation Token Ring and Ethernet networks to the boot/install server. Define only the Control Workstation networks that are not on the same subnet as the Ethernet interface on the boot/install server.
      - Take any of the defined Control Workstation networks and make the Control Workstation a client of the boot/install server. The Control Workstation can serve the lppsource resources to the clients of the boot/install server.

- Define the route info to the Control Workstation for all network resources.

2.2.2.8 mknimres

This wrapper creates NIM resources that are needed for operations on the system. Depending on the value of the bootp_response field, the resources that are involved in the processes with the given bootp_response and do not yet exist, will be created.

bootp_response	**Resources to be created**
install	pssplpp, lppsource, mksysb, spot
customize	pssplpp
disk	None
maintenance	lppsource, spot
diag	spot
migrate	pssplpp, lppsource, spot

If resources were already created, they are checked by using the `lsnim -t resource` command, as follows:

```
lsnim -t lpp_source
```

or

```
lsnim -t spot
```

So if a resource already exists, mknimres checks its Rstate attribute. If it is “ready for use,” the resource seems to be OK. It it is “not ready for use,” the mknimres wrapper terminates with an error message.

Note

Since mknimres creates all needed resources, it also creates the SPOTs. If your SP system contains High Nodes, then while creating the SPOT, the boot images for rs6smp are created in the /tftpboot directory. Make sure that the images of the following file sets are included in the corresponding lppsource:

devices.rs6ksmp.base.usr
bos.rte.mp.usr (AIX 4.1.x)
bos.mp.usr (AIX 4.2)

Command Syntax:

```
mknimres -h | -l <node_number_list>
```

where:

-h Display command syntax.

-l <node_number_list>

List of node numbers to be defined as NIM masters. This is a required option. Node 0 is the Control Workstation.

General flow and logic for mknimres:

1. Parse command line parameters.
2. Get information from SDR.
3. Investigate prerequisites and conditions:
    - NIM master must be the Control Workstation or boot server (see SDR).
    - Make sure you have dsh access to the node.
4. Delete all resources with Rstate "not ready for use."
5. Create all resources needed for the operation.
6. Copy pssp.installp to the boot/install server.

2.2.2.9 mknimclient

This wrapper creates NIM client definitions on the boot/install server. It searches for the processor type of a client (UP/MP). The processor type is used in creating the SPOT, in fact in creating the boot images for the nodes. If the client is not attached directly to the Control Workstation, the mknimclient command builds the client routes to the Control Workstation for using the lppsource.

Command Syntax:

```
mknimclient -h | -l <node_number_list>
```

where:

-h Display command syntax.

-l <node_number_list> List of node numbers to define as NIM clients. This is a required option. Node 0 (the CWS) is not allowed.

Example:

```
mknimclient -l 16
```

This command creates the NIM client definitions for Node 16 on the NIM master, serving Node 16.

General flow and logic for mknimclient:

1. Get information from SDR.
2. Parse command line parameters.

    For each specified node:

    a. Determine client's NIM master.
    b. Check prerequisites and conditions:
        - Client's server must have the NIM master file set installed.
        - Client's server must be configured as a NIM master.
        - Client and server must be on the same subnet.
        - If the client is already defined on the server, skip this node.
    c. Define the client on the master.

2.2.2.10 mkconfig

This script creates the /tftpboot/<hostname>.config_info files for every node that has bootp_response not set to "disk." These files are used during network installation of an AIX mksysb image. The information that has been retrieved from the SDR for every node (bootp_response != disk) is:

node_number
bootp_response
initial_hostname
default_route
switch_node_number
switch_number
switch_chip
switch_chip_port
slots_used
reliable_hostname.

This data is used in creating the config_info files. `mkconfig` uses reliable hostnames for creation of the files.

Here is an example of the /tftpboot/<hostname>.config_info file:

```
16 sp2n16 192.168.3.16 192.168.3.37 15 1 7 3 1 yes
en0 192.168.3.16 255.255.255.0 NA bnc
css0 192.168.13.16 255.255.255.0 NA ""
```

The fields of the config_info file contain the following:

- Node number
- Initial hostname
- Default route
- An entry for each adapter defined in the SDR, listing:
  - Adapter name
  - Adapter IP address
  - netmask
  - ring_speed (for Token Ring)
  - bnc_select (dix/bnc for Ethernet)

Command Syntax:

```
mkconfig
```

This command has no input parameters.

General flow and logic for `mkconfig`:

1. Investigate prerequisites and conditions:
   - NIM master must be the Control Workstation or boot/install server (see SDR).
   - Make sure you have `dsh` access to the node.

   For all nodes with bootp_response != disk:

a. Get SDR data for each node (see previous list).

b. Add the arp_enabled field to the definition of each client.

c. Create the /tftpboot/<reliable_hostname>.config_info file for each client.

2.2.2.11 mkinstall

This wrapper creates the /tftpboot/<hostname>.install_info file for each node with bootp_response not set to "disk." These files, like the config_info files, are used during installation of the nodes. mkinstall retrieves site environment data from the SP class of the SDR, gets the node information, and if the node's bootp_response is not set to "disk," the existing /tftpboot/<hostname>.install_info file is removed and a new install_info file is created for the node. mkinstall uses reliable_hostname.

The following is an example of the /tftpboot/<hostname>.install_info file:

```
#!/bin/ksh
export control_workstation="9.12.1.37 192.168.3.40 192.168.3.39 192.168.3
.38 192.168.3.37"
export cw_hostaddr="192.168.3.37"
export cw_hostname="sp2en0.msc.itso.ibm.com"
export server_addr="192.168.3.37"
export server_hostname="sp2en0.msc.itso.ibm.com"
export rel_addr="192.168.3.16"
export rel_hostname="sp2n16.msc.itso.ibm.com"
export initial_hostname="sp2n16"
export primary_auth_hostname="sp2en0.msc.itso.ibm.com"
export primary_auth_addr="192.168.3.37"
export authent_server="ssp"
export netinst_boot_disk="hdisk0"
export netinst_bosobj="bos.obj.ssp.42"
export remove_image="false"
export sysman="true"
export code_version="PSSP-2.2"
export proctype="UP"
export lppsource_name="aix42"
export cwsk4=""
export lppsource_hostname="sp2en0.msc.itso.ibm.com"
export lppsource_addr="192.168.3.37"
```

Command Syntax:

```
mkinstall
```

This command has no input parameters.

General flow and logic for mkinstall:

1. Get SP site environment from the SDR.
2. Find the name of the lppsource resource.
3. Find the server for the lppsource resource.
4. Find the if1 hostname of the lppsource server.
5. Get the realm name and the primary server hostname from the Kerberos config file /etc/krb.conf.
6. Get the IP address for the primary authentication server.
7. Check if there is any need to install ssp.sysman during node installation.

8. Find the PSSP version for each node.

9. Get the node data from the SDR.

10. If bootp_response is "disk," no activities are done for this node.

11. If bootp_response is not "disk," remove the existing install_info file and create a new one for the client.

2.2.2.12 allnimres

This wrapper allocates all necessary NIM resources to a client, depending on the client′s bootp_response (SDR). This includes executing the `bos_inst` command for allocation of the boot resource and nimscript resource. When this command is done, nodes are ready for netboot install, diagnostics, or maintenance. If a node′s bootp_response is set to "disk" or "customize," then all NIM resources are deallocated from the node. The bootp_response values available in PSSP 2.2 are:

- install
- customize
- disk
- maintenance
- diag
- migrate

Because each of these values generates a different flow of NIM operations, have a look at the flow chart of `allnimres` in Appendix E, "Flowcharts of setup_server Wrappers" on page 287 for a better understanding of which configuration actions are performed by which bootp_response type.

Command Syntax: `allnimres -l <list_of_nodes>`

Example: `allnimres -l 11,12,15,16`

This command allocates all resources to Nodes 11,12,15 and 16 according to the value of bootp_responds of these nodes.

General flow and logic for `allnimres`:

1. Get information from the SDR.
2. Parse command line parameters.

   For each specified node:

   a. Reset the node.
   b. Deallocate all resources from the node.
   c. Allocate the NIM resources necessary for the assigned bootp operation to be performed.
   d. Boot the node in the assigned bootp_response mode.

2.2.3 Problem Determination

This redbook endeavors to help in solving problems by showing the correct techniques for problem determination. Since we cannot analyze every specific problem that can occur running setup_server, we use a few sample problems and point out the most important steps that should be taken in these or similar situations.

Let us first put the problems into categories and then show a method for problem determination for each category:

1. Problems related to Kerberos.
2. Problems related to the SDR data.
3. Problems related to NIM performing resources and objects operations (create, change, delete, allocate, deallocate, reset, and so on).

There are two ways of solving NIM problems in the setup_server script. One of them is to use the wrappers, the other to use the NIM commands directly. You can use whichever way you prefer, depending on what is better in your case.

2.2.3.1 Problems Related to Kerberos

There are some situations where Kerberos does not work. Most of these cases originate in:

- Problems with Kerberos daemons
- Problems with not having a valid ticket
- The wrong information in:
  - Kerberos configuration files (/etc/krb.conf, /etc/krb.realms)
  - .klogin file (remote principals)
  - Service key file (/etc/krb-srvtab)
- TCP/IP or/and hostname resolution problems
- PTF levels

This chapter does not cover problem determination on Kerberos itself. We intend to point to setup_server problems having connection with Kerberos. The following scenario is an example for this sort of problem. (For information related to Kerberos problems, refer to *RS/6000 SP: Problem Determination Guide SG24-4778*.)

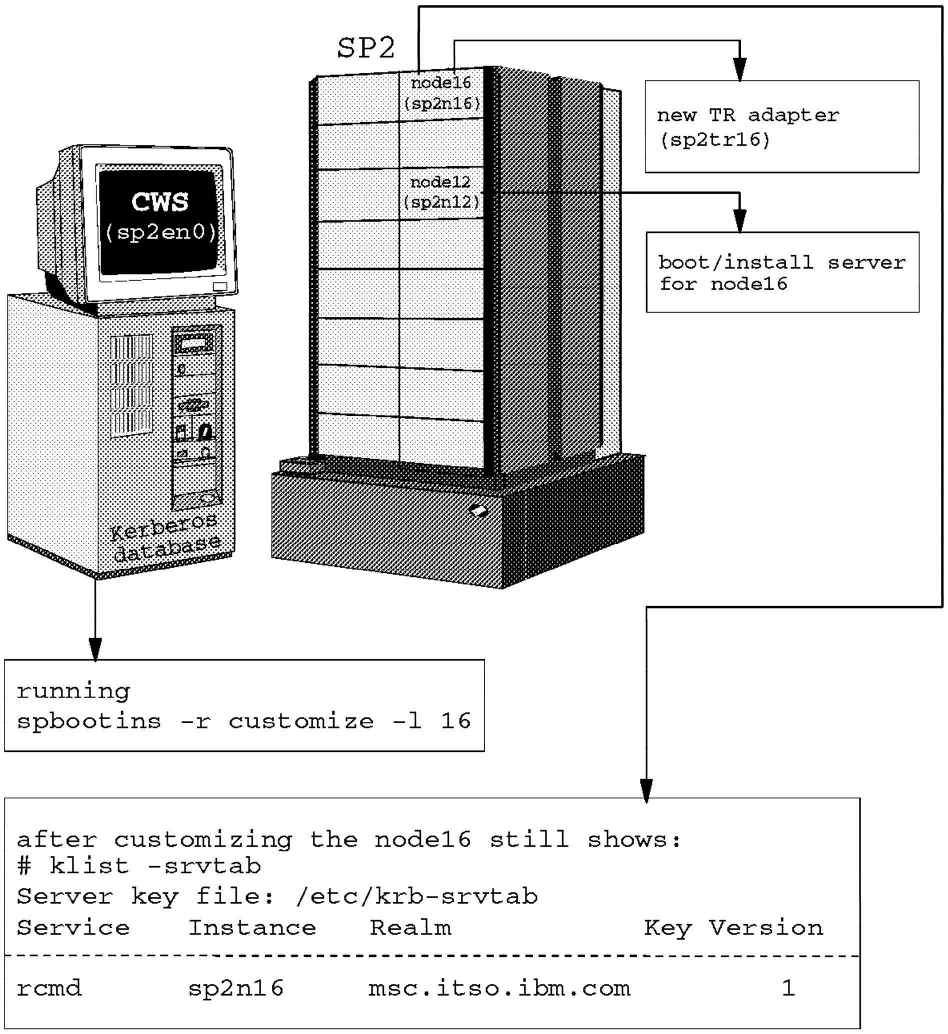
Scenario 1

Symptom: Problems with the /etc/krb-srvtab file.

Environment:

Control Workstation at AIX 4.1.4 PSSP 2.1.
Boot/Install Server other than Control Workstation

Problem Description:



```
running
spbootins -r customize -l 16
```

```
after customizing the node16 still shows:
# klist -srvtab
Server key file: /etc/krb-srvtab
Service    Instance    Realm            Key Version
---------------------------------------------------
rcmd       sp2n16      msc.itso.ibm.com         1
```

Figure 18. Problems with /etc/krb-srvtab File

After adding a network device to an SP node, the setup_server command failed to include a new entry into the new-srvtab file with the new network device.

Actions taken:

1. Added a new Token Ring adapter to the node.
2. Used `smit` to configure additional adapters.
   (`splstdata -a` shows the proper output.)
3. Set the node to “customize” (which runs setup_server).

Results of these actions:

- The new-srvtab file in /tftpboot for the Token Ring has a length of 0 Bytes 0 bytes.
- The new-srvtab file put on the node has only the en0 interface.

Problem Determination:

Let us draw attention to the file /etc/krb-srvtab. What is the content of it? Was it created in a proper way?

During the setup of the Control Workstation the keys for service principals were stored in the authenticated database and in the /etc/krb-srvtab file. Every node and the Control Workstation should have the /etc/krb-srvtab file with the keys for the services provided on that host. The command `klist -srvtab` shows you the service available for that host. If you run this command on the Control Workstation with partitions defined for the SP, the output shows you the services for each partition:

```
#klist -srvtab
Server key file:   /etc/krb-srvtab
Service         Instance        Realm      Key Version
------------------------------------------------------
hardmon         sp2cw0          MSC.ITSO.IBM.COM 1
rcmd            sp2cw0          MSC.ITSO.IBM.COM 1
hardmon         sp2en3          MSC.ITSO.IBM.COM 1
rcmd            sp2en3          MSC.ITSO.IBM.COM 1
rcmd            sp2en2          MSC.ITSO.IBM.COM 1
hardmon         sp2en2          MSC.ITSO.IBM.COM 1
rcmd            sp2en1          MSC.ITSO.IBM.COM 1
hardmon         sp2en1          MSC.ITSO.IBM.COM 1
rcmd            sp2en0          MSC.ITSO.IBM.COM 2
hardmon         sp2en0          MSC.ITSO.IBM.COM 2
```

The output of this command, executed on the node with en0 and tr0 interfaces, shows the following services:

```
#klist -srvtab
Server key file:   /etc/krb-srvtab
Service         Instance        Realm      Key Version
------------------------------------------------------
rcmd            sp2tr16        MSC.ITSO.IBM.COM 1
rcmd            sp2n16         MSC.ITSO.IBM.COM 1
```

In our case, the node has a /etc/krb-srvtab file having only the en0 interface:

```
#klist -srvtab
Server key file:   /etc/krb-srvtab
Service         Instance        Realm      Key Version
------------------------------------------------------
rcmd            sp2n16         MSC.ITSO.IBM.COM 1
```

This problem seems to occur only when using a boot/install server other than the Control Workstation.

So what is the actual state of the /tftpboot/<hostname>-new-srvtab files?

- The new-srvtab file for the Token Ring /tftpboot/sp2tr16-new-srvtab has a length of 0.
- The sp2n16-new-srvtab file does not include the Token Ring interface.

Let us look at the `create_krb_files` wrapper. In the code, beginning with line# 426, we see:

```
# This node is not the control workstation - it is a server node.
# Get the server key file for client node from the control workstation
# Invoke mksrvtab to create or locate a server key file for client.
#
$RSH $control_host -n $MKSRVTAB $ihost $auth_srvr $host_list[$client_node];
  $rc = $?>>8;            # rc of rsh (not the actual cmd run on $control)
  if ($rc != 0) {         # most likely timed out
      &cat_sminstmsg("emsg068", $progname, $control_host, $rc);
      exit(1);
  }

#
# Get the srvtab file for the node
#
# print "$progname: Copying /tftpboot/$ihost-new-srvtab from $control_host \n";
 $RCP $control_host:/tftpboot/$ihost-new-srvtab \
      /tftpboot/$ihost-new-srvtab;
 if (! -e "/tftpboot/$ihost-new-srvtab") {
     &cat_sminstmsg("emsg054",$progname,$ihost);
     print "$CAT_MSG";
     exit(1);
 }
 if ($auth_srvr eq "ssp") {
 $RSH $control_host -n $RM -f /tftpboot/$ihost-new-srvtab ;
    $rc = $?>>8;      # rc of rsh (not the actual cmd run on $control)
    if ($rc != 0) {   # most likely timed out
    &cat_sminstmsg("emsg068", $progname, $control_host, $rc);
    print "$CAT_MSG";
    exit(1);
    }                             # End if
  }                               # End if
}                                 # End else not CW
```

We can see that if the boot/install server is other than the Control Workstation, the node server uses rsh to create the srvtab file on the Control Workstation.

```
$RSH $control_host -n $MKSRVTAB $ihost $auth_srvr $host_list[$client_node];
```

If the boot/install server is not the Control Workstation, the new-srvtab file is removed from the Control Workstation.

```
$RSH $control_host -n $RM -f /tftpboot/$ihost-new-srvtab;
```

The boot/install server uses the rcp command to copy the /tftpboot/<hostname>-new-srvtab file for the client in the local /tftpboot directory.

```
$RCP $control_host:/tftpboot/$ihost-new-srvtab /tftpboot/$ihost-new-srvtab
```

If we now follow the code of the mksrvtab command (located in the /usr/lpp/ssp/install/bin directory) that is used by create_krb_files, we see that this script is run on a Control Workstation by the setup_server script via rsh from other boot servers. It creates a server key file for the client node in /tftpboot, if the Kerberos database is on the Control Workstation.

Our problem is: the srvtab file for the client does not include the entry for the new Token Ring interface.

The only explanation for this phenomenon is that at the time when setup_server was running (invoked by the `spbootins -r` customize command), the new interface did not exist in the Kerberos database. Hence, it failed in creating a new-srvtab file (for that interface), and a zero-length new-srvtab file was created.

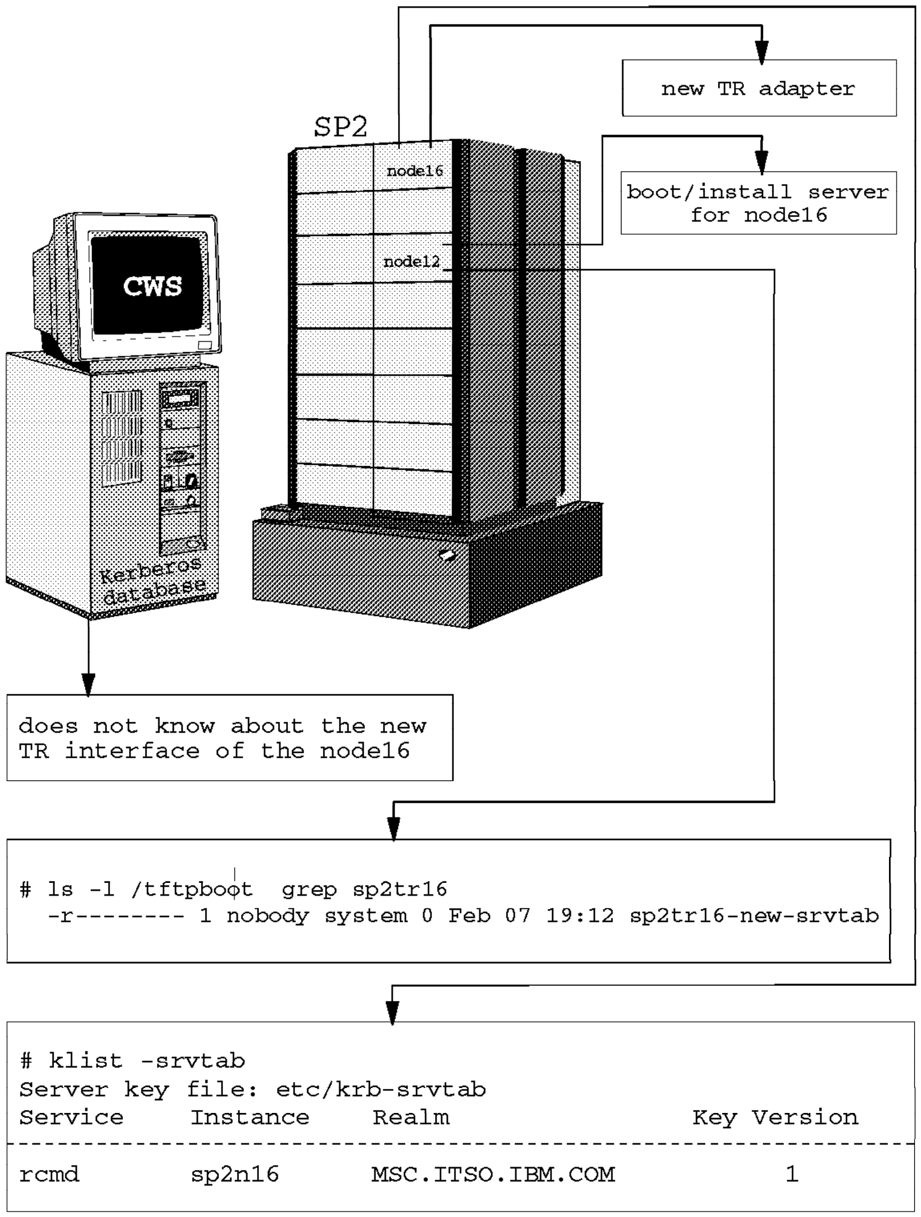
The next action, to combine the new-srvtab files for each interface into a single file to be copied to the boot/install server, failed, too. The new-srvtab file that is put on the boot/install server only has a single interface (the hostname of the node), and this is exactly our problem.

Important

This problem occurs only if you are using a boot/install server other than the Control Workstation.

How do we refresh and update the Kerberos database with the new information? Run setup_server (without setting the node to "customize"); it will properly add the interface to the Kerberos database, and subsequent runs of setup_server will be successful.

Conclusion:


SP2
node16
node12
new TR adapter
boot/install server for node16
CWS
Kerberos database
does not know about the new TR interface of the node16

```
# ls -l /tftpboot  grep sp2tr16
  -r-------- 1 nobody system 0 Feb 07 19:12 sp2tr16-new-srvtab
```

```
# klist -srvtab
Server key file: etc/krb-srvtab
Service     Instance     Realm                  Key Version
-------------------------------------------------------------------
rcmd        sp2n16       MSC.ITSO.IBM.COM             1
```

Figure 19. Result of the Problem Determination: Service key file

Running setup_server while setting the node to "customize" response after adding a new network adapter to the node will not succeed in creating a /etc/krb-srvtab file on the node if you are using a boot/install server other than the Control Workstation.

Solution:

1. Run setup_server without setting the node to "customize."
2. Set the node to "customize" and run setup_server again.

2.2.3.2 Problems Related to SDR Data

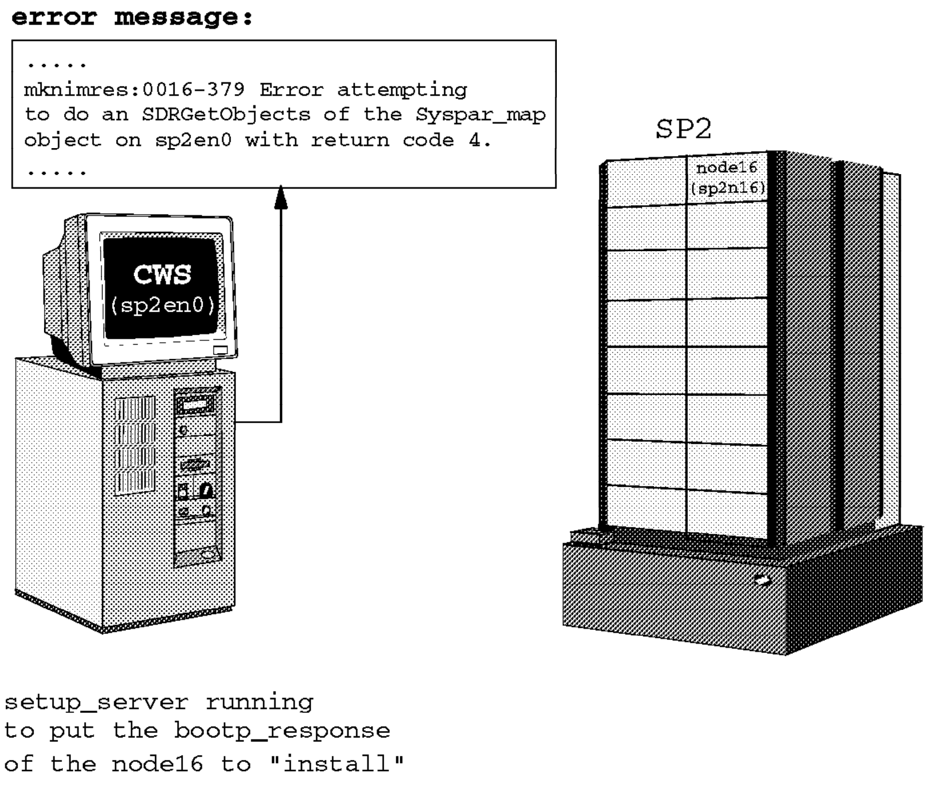
Looking at the code of the setup_server wrappers, we see that in every single case, the step of getting the required data from the SDR is performed. If the data is not correct, or perhaps is non-existent, or still worse, if it is corrupted, then setup_server will not terminate successfully. We used one of these situations for creating our problem determination path for this category of problems.

Scenario 2

Symptom: setup_server fails while setting a node′s bootp_response to “install.”

Environment: Control Workstation at AIX 4.2 and PSSP 2.2.

Problem Description:



```
error message:
.....
mknimres:0016-379 Error attempting
to do an SDRGetObjects of the Syspar_map
object on sp2en0 with return code 4.
.....
```

Figure 20. Problems while Setting a Node′s bootp_response to install

Running setup_server on the Control Workstation to set Node 16 to the “install” response, the mknimres Perl script returns an error to setup_server. The Control Workstation was migrated from AIX 4.1.4 with PSSP 2.1 to AIX 4.2 with PSSP 2.2.

setup_server output:

```
setup_server command results from sp2en0
--------------------------------------------------------
setup_server: Running services_config script to configure SSP services.This
may take a few minutes ...
rc.ntp: NTP already running - not starting ntp
0513-029 The supfilesrv Subsystem is already active.
Multiple instances are not supported.
setup_CWS: Control Workstation setup complete.
mknimmast: Node 0 (sp2en0) already configured as a NIM master.
create_krb_files: tftpaccess.ctl file and client srvtab files created/updated
on server node 0.
mknimres: Copying /usr/lpp/ssp/install/bin/pssp_script to
          /spdata/sys1/install/psspscript.
mknimres: Copying /usr/lpp/ssp/install/config/bosinst_data_migrate.template
to /spdata/sys1/install/pssp/bosinst_data_migrate.
mknimres: 0016-379 Error attempting to do an SDRGetObjects of the Syspar_map
          object on sp2en0 with return code 4.
setup_server: 0016-279 Failure of internally called command:
/usr/lpp/ssp/bin/mknimres; rc= 255.
setup_server: Processing complete (rc= -1).
```

Problem Determination:

The error messages show us that the mknimres wrapper managed to complete the steps successfully until the copying of the bosinst_data_migrate file, inclusively. Reading the data from the SDR caused the problems; the class that could not be read was Syspar_map.

Let us try to find the position of the lines of code that failed. If you look at the mknimres wrapper, beginning with line# 623, you find the command that failed:

```
if ($rc=&runitrc($host_name, "$SDRGO -x -G Syspar_map used==1 syspar_name \
node_number 2> /dev/null")){
   &cat_sminstmsg("emsg379", $progname, $host_name, $rc); # SDRGetObjects
        of the Syspar_map failed
   print (STDERR $CAT_MSG);
   exit -1;
 }
```

So the SDRGetObjects command failed. What is the state of bootp_response of Node 16 (sp2n16)? Is it "install," or something else? The command splstdata -b, set for this partition, shows:

```
# splstdata -b
                  List Node Boot/Install Information

node#        hostname  hdw_enet_addr srvr     response           install_disk
      last_install_image   last_install_time  next_install_image lppsource_name
-------------------------------------------------------------------------------
   11 sp2n11             10005AFA0DE6    0        disk                 hdisk0
          bos.obj.ssp.42 Sun_Jan_26_00:16:44      bos.obj.ssp.42         aix42
   12 sp2n12             10005AFA0F81    0        disk                 hdisk0
          bos.obj.ssp.42 Sun_Jan_26_00:11:05      bos.obj.ssp.42         aix42
   15 sp2n15             10005AFA147C    0        disk                 hdisk0
          bos.obj.ssp.42 Sun_Jan_26_00:09:41      bos.obj.ssp.42         aix42
   16 sp2n16             10005AFA0AB5    0     install                 hdisk0
          bos.obj.ssp.42 Sun_Jan_26_00:10:38      bos.obj.ssp.42         aix42
```

Our problem is:

- The SDRGetObjects command failed reading Syspar_map.
- Node 16 (sp2n16) is set to "install," but no resources have been allocated to this node. (You can check this out by using the lsnim -l <node_name> command.)

If we now:

1. Look at the error message again that was displayed by the setup_server output:

```
...
mknimres: 0016-379 Error attempting to do an SDRGetObjects of the Syspar_map
          object on sp2en0 with return code 4.
...
```

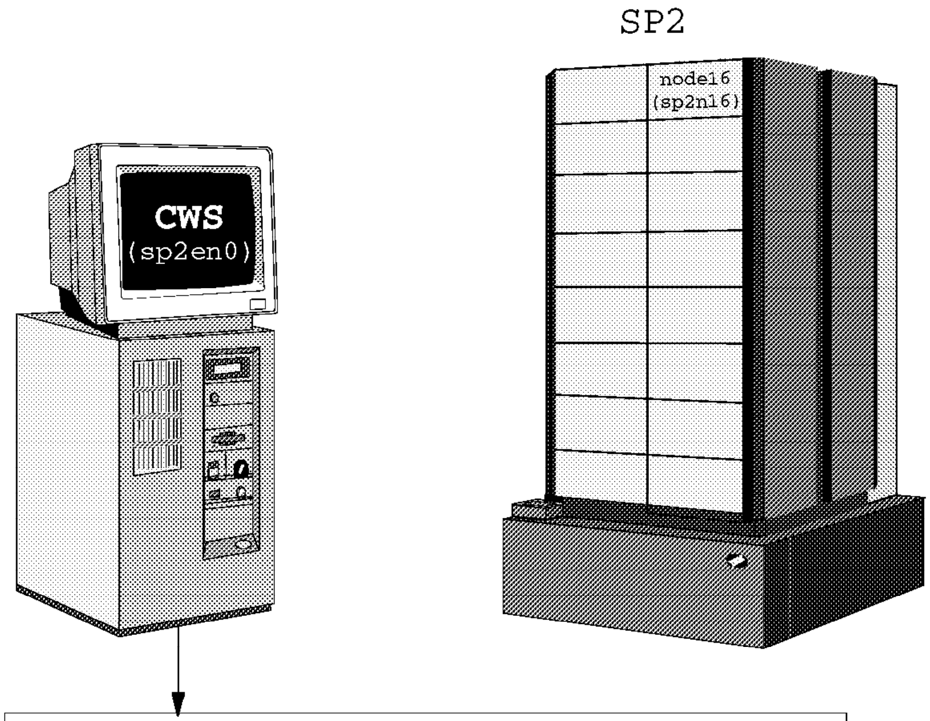
2. Issue the command used in line# 623 of the mknimres script:

```
# SDRGetObjects -x -G Syspar_map used==1 syspar_name node_number
SDRGetObjects: 0025-004 Item specified for query, insertion or deletion
was not found.
```

we see that return code 4 means:
"Item specified for query, insertion or deletion was not found."

It is now obvious that for some reason, maybe while migrating the Control Workstation, the Syspar_map class became corrupt; as a matter of fact, this class does not contain any data.

Conclusion: Figure 21 on page 110 shows the result of our problem determination in this case.



```
# cat /spdata/sys1/sdr/system/classes/Syspar_map
# ls -l
# ls -l /spdata/sys1/sdr/system/classes/Syspar_map
# -rw------- 1 root system 1 Feb 08 11:58 Syspar_map
```

Figure 21. Result of the Problem Determination: Data Corruption

While performing operations on the Control Workstation, the Syspar_map class became corrupt. `mknimres` failed while trying to read nonexistent SDR data.

Solution:

1. Recreate the Syspar_map.

   a. You can do it manually using a text editor. The Syspar_map has the following structure:

```
1=sp2en3 2=192.168.3.40 3=1 4=0 5=1
1=sp2en3 2=192.168.3.40 3=2 4=1 5=1
1=sp2en1 2=192.168.3.38 3=3 4=2 5=1
1=sp2en1 2=192.168.3.38 3=4 4=3 5=1
1=sp2en3 2=192.168.3.40 3=5 4=4 5=1
1=sp2en3 2=192.168.3.40 3=6 4=5 5=1
1=sp2en1 2=192.168.3.38 3=7 4=6 5=1
1=sp2en1 2=192.168.3.38 3=8 4=7 5=1
1=sp2en0 2=192.168.3.37 3=9 4=8 5=1
1=sp2en0 2=192.168.3.37 3=10 4=9 5=1
1=sp2en2 2=192.168.3.39 3=11 4=10 5=1
1=sp2en2 2=192.168.3.39 3=12 4=11 5=1
1=sp2en0 2=192.168.3.37 3=13 4=12 5=1
1=sp2en0 2=192.168.3.37 3=14 4=13 5=1
1=sp2en2 2=192.168.3.39 3=15 4=14 5=1
1=sp2en2 2=192.168.3.39 3=16 4=15 5=1
```

   The field numbers correspond to the following values:

- 1 = syspar_name
- 2 = syspar_addr
- 3 = node_number
- 4 = switch_node_number
- 5 = used

Note

Let us say that the last node in our system was a High Node. The Syspar_map for this configuration would look like the following (in the "5" values, 0 indicates that the slot is not used):

```
1=sp2en3 2=192.168.3.40 3=1 4=0 5=1
1=sp2en3 2=192.168.3.40 3=2 4=1 5=1
1=sp2en1 2=192.168.3.38 3=3 4=2 5=1
1=sp2en1 2=192.168.3.38 3=4 4=3 5=1
1=sp2en3 2=192.168.3.40 3=5 4=4 5=1
1=sp2en3 2=192.168.3.40 3=6 4=5 5=1
1=sp2en1 2=192.168.3.38 3=7 4=6 5=1
1=sp2en1 2=192.168.3.38 3=8 4=7 5=1
1=sp2en0 2=192.168.3.37 3=9 4=8 5=1
1=sp2en0 2=192.168.3.37 3=10 4=9 5=1
1=sp2en2 2=192.168.3.39 3=11 4=10 5=1
1=sp2en2 2=192.168.3.39 3=12 4=11 5=1
1=sp2en0 2=192.168.3.37 3=13 4=12 5=1
1=sp2en0 2=192.168.3.37 3=14 4=13 5=0
1=sp2en2 2=192.168.3.39 3=15 4=14 5=0
1=sp2en2 2=192.168.3.39 3=16 4=15 5=0
```

b. You can use the SDRCreateObjects command, adding lines one after another:

```
SDRCreateObjects Syspar_map syspar_name=sp2en3 \
 syspar_addr=192.168.3.40 node_number=1 switch_node_number=0 used=1
```

```
SDRCreateObjects Syspar_map syspar_name=sp2en3 \
 syspar_addr=192.168.3.40 node_number=2 switch_node_number=1 used=1
```

...and so on...

2. Stop the SDR daemon(s)
3. Start the SDR daemon(s)
4. Set Node 16 to "disk" and "install" again.

2.2.3.3 Problems Related to NIM Operations

Most of the problems that occur while running setup_server are problems related to the NIM operations. There are many reasons for setup_server failures. Some of them have to do with the creation or allocation of NIM resources, for example, missing components of software, or not enough space on the corresponding file system, or perhaps there is a problem with the permissions for a specific file that is taking part in a process, or the resources involved in this process are not available or accessible. Operations with resources like mksysb_#, prompt, noprompt, migrate or psspscript usually fail when the files needed for their creation or allocation do not exist or have permission problems.

The noprompt resource requires /spdata/sys1/install/pssp/bosinst_data. This is a copy of /usr/lpp/ssp/install/config/bosinst_data.template, which is made while running setup_server.

The prompt resource requires /spdata/sys1/install/pssp/bosinst_data_prompt. This is a copy of /usr/lpp/ssp/install/config/bosinst_data_prompt.template, which is made while running setup_server.

The migrate resource requires /spdata/sys1/install/pssp/bosinst_data_migrate. This is a copy of /usr/lpp/ssp/install/config/bosinst_data_migrate.template, which is made while running setup_server.

The psspscript resource requires /spdata/sys1/install/pssp/pssp_script, which is a copy of /usr/lpp/ssp/install/bin/pssp_script.

The resources that are mostly involved in NIM resource operation problems are lppsource and SPOT.

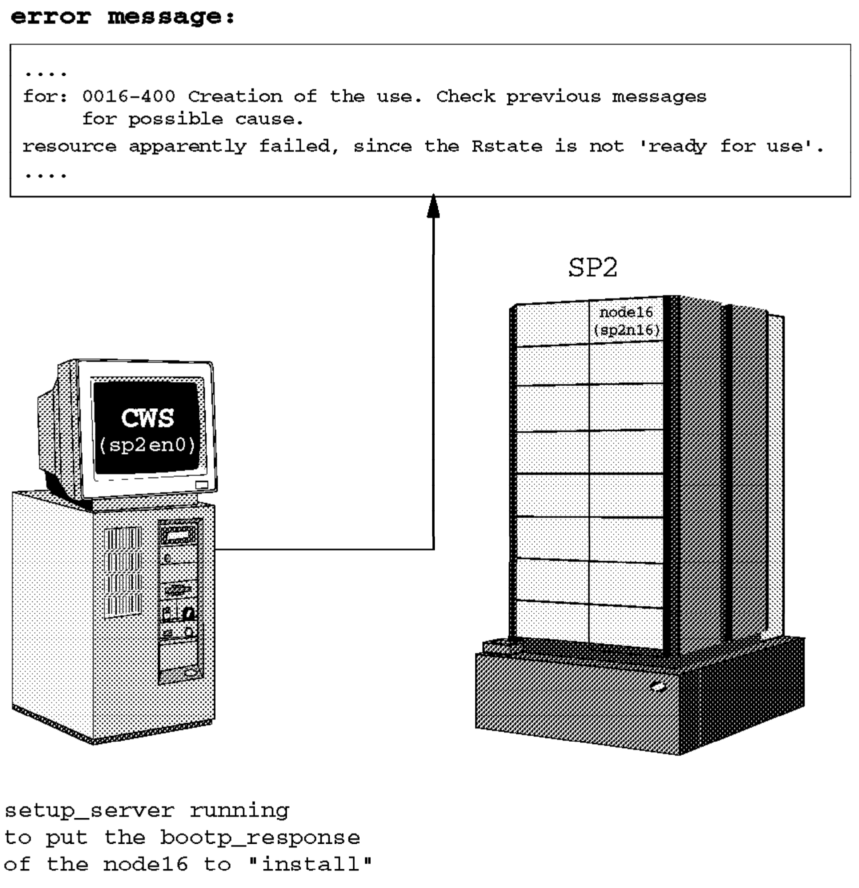
The first step should be to determine in which part of the script the failure occurred, and what error messages are produced by this part of the script.

Scenario 3

Symptom: setup_server fails while creating a resource.

Environment: Control Workstation at AIX 4.1.4 and PSSP 2.2.

Problem Description:



error message:

```
....
for: 0016-400 Creation of the use. Check previous messages
     for possible cause.
resource apparently failed, since the Rstate is not 'ready for use'.
....
```

```
setup_server running
to put the bootp_response
of the node16 to "install"
```

Figure 22. setup_server Fails while Creating a Resource

Running setup_server on the Control Workstation while bootp_response is set to "install," the *mknimres* wrapper returns an error to setup_server.

Setup_server output:

```
...
mknimres: Copying
 /usr/lpp/ssp/install/config/bosinst_data_migrate.template
         to /spdata/sys1/install/pssp/bosinst_data_migrate.
for: 0016-400 Creation of the use.  Check previous messages
     for possible cause.
resource apparently failed, since the Rstate is not ′ready for use′.
Refer to /tmp/spot.out.99999 for more debug information.
setup_server:
0016-279 Failure of internally called command:
/usr/lpp/ssp/bin/mknimres; rc= 255.
setup_server: Processing complete (rc= -1).
```

Problem Determination:

The process of copying the bosinst_data_migrate file was the last successful action before the error messages appeared. So we see that one "resource apparently failed" but which one? We can guess that it is the SPOT resource, because of the recommendation in the next line "Refer to /tmp/spot.out.99999 for more debug information."

To make sure that the problem occurred in creating the SPOT resource, let us look at the mknimres wrapper and try to locate the action that failed. The starting point for this analysis should be to make clear which parts of mknimres are involved by running setup_server while setting the node to the response "install."

Which parts of the wrapper are executed depends on the response to which you want to set the node. Let us look at the table of required resources for each type of response. You can find it in the mknimres script beginning with line #107.

```
# Array to hold table of required resources  26058
$Reqd_Rsrc{install}=″:pssplpp:lppsource:mksysb:spot:″;
$Reqd_Rsrc{customize}=″:pssplpp:″;
$Reqd_Rsrc{disk}=″″;
$Reqd_Rsrc{maintenance}=″:lppsource:spot:″;
$Reqd_Rsrc{diag}=″:spot:″;
$Reqd_Rsrc{migrate}=″:pssplpp:lppsource:spot:″;
```

Now we can see that if the node is going to be set up to *install*, the resources that will be involved while running *mknimres* are pssplpp, lppsource, mksysb, and SPOT.

Let us locate where we are now. The last message of a successful action was about copying the bosinst_data_migrate file. Beginning with line #122, you see the lines:

```
foreach $node_number (@node_list){     # For each supplied node...
   $host_name=$hostname{$node_number}; # Set hostname for loop

   unless(&check_prereqs){
       exit -1;
   }
   #put the output of lsnim -l in the @lsnim array
   unless(&fill_lsnim(@lsnim)){
       exit -1;
   }
   unless(&delete_notready){           # delete resources whose RState
       exit -1;                        # is not equal to 'ready for use'
   }
   unless(&make_script(@lsnim)){
       exit -1;
   }
   unless(&make_bosinst_data(@lsnim)){
       exit -1;
   }
   .......
```

Here we are! The subroutine make_bosinst_data was the last one that was successfully executed. The next steps performed by mknimres for the response "install" are the following subroutines (see the code):

make_custom_bosinst_data()
make_lppsource()
make_mksysb()
make_spot()
make_pssplpp()

Let us start with the analysis of the function make_custom_bosinst_data().

This function is executed only if mknimres runs for a node which has a target disk for the installation set to another disk, rather than to hdisk0. In this case, the bosinst_data file used for the installation of this node is not the default file, but a file created by this function, named clientname.bosinst. This function was not executed in our case, since all the target disks of the nodes were set to hdisk0.

The rest of the listed functions, that is, make_lppsource, make_mksysb, and make_spot, have a similar structure.

First, the data related to the type of the resource is read from the SDR. Then a check is made to see if the given resource is needed for the type of response to which the nodes are set. The nodes with PSSP 1.2 are ignored. Since they have their own install servers, the information about them is not stored in the NIM database.

The next steps that are taken are checks on what resources of the given type are already defined. If the resource that is needed according to the boot response of the node(s) is not defined yet, it is now created. Afterwards, the running state (Rstate) of the resources that are going to be used for the operation, depending on the type of the response to which the node(s) are set, is checked (ready for use, or not ready for use; for lppsource, the value of the simages attribute is also checked).

If the checks complete without problems, the functions terminate properly. If, for example, the Rstate checks do not pass, the function displays an error message and the definition of the resource that had the Rstate set to “not ready for use,” is removed from the NIM database.

Let us have a look at the make_lppsource() function, as an example.
The first step is to read the SDR data about lppsource_name, code_version and boot_response from the Node class for all nodes in all partitions:

```
if ($rc=&runitrc($host_name, ″$SDRGO  -G -x Node lppsource_name boot_serv
er==$node_number code_version bootp_response″)){
   &cat_sminstmsg(″emsg373″, $progname, $host_name); # error getting
                                              #lppsource_name attributes
   print (STDERR $CAT_MSG);
   exit -1;
}
```

If this command fails, the make_lppsource() function fails with the error message emsg373. Now, what is this message, and where does the function take it from? In line #64 of mknimres, we find the name of the file that includes the error messages for this wrapper. In our case, it is the file /usr/lpp/ssp/bin/sminstmsgs.pl. Let us search for the string “emsg373” in this file. We see that the error message would be: `0016-374  An error occurred trying to get all the lppsource_name attributes from the SDR on ....`

We did not get this message, which means that the first step of make_lppsource() was done correctly.

The next step in make_lppsource() deals with the check to see if lppsource is needed for the response of the node. The nodes with PSSP 1.2 are ignored for the reasons mentioned previously.

The next steps taken by make_lppsource() are checks to see what lppsources are defined on the system. If the lppsource object that is needed for the operation on the node(s) (install,maintenance, migrate...) does not exist yet, it is created. Then the checks are done about the Rstate and simages attributes of lppsources that are going to be used by the operations. If the checks complete without problems, the make_lppsource() function finishes.

We did not have any problems with the make_lppsource() function, and the next function, make_mksysb(), terminated without errors as well.

The error message that we got from mknimres referred to the creation of some resource. The next possible resource that could be created would be SPOT.

Let us locate the point in the make_spot() function from which the error messages would come. Following the source code, we see that this function is built in a way similar to make_lppsource(): first read the data from the SDR, then search if there already exist SPOTs corresponding to the existing lppsources, and if not, and they should be used for the operation, create them; and if they exist, then check the Rstate attribute.

Here we are! Now we can check whether the error message about the Rstate of “the resource” really referred to the SPOT. Let us take a closer look at the subroutine that checks the Rstate. It is called ok_or_exit() and passes the resource name as the parameter. Let us look at what happens if Rstate does not have the attribute “ready for use.” The function ok_or_exit() can be found in line# 972 of the mknimres wrapper:

```
sub ok_or_exit{
  local($resource_name)=@_;

  if($rc=&runitrc($host_name,"$LSNIM -Za Rstate $resource_name")){
        &cat_sminstmsg("emsg399", $progname, $host_name, $resource_name,
$rc); # lsnim -Za Rstate failed
  print (STDERR $CAT_MSG);
  exit -1;
}
chop(@Rstate=grep(!/#/,@RUNITRC_OUT));
foreach(@Rstate){
       s/$host_name: //;
       ($junk1, $Rstate)=split(/:/,@Rstate.[0]);
}
if ($Rstate ne "ready for use"){
      &cat_sminstmsg("emsg400", $progname, $resource_name); # creation of
resource apparently failed.
      print (STDERR $CAT_MSG);
      $rc=&runitrc($host_name,"$NIM -o remove $resource_name");
        if (substr($resource_name,0,5) eq "spot_"){
              &cat_sminstmsg("emsg407", $progname, $$);
              print (STDERR $CAT_MSG);
        }
        exit -1:
}
```

If the Rstate of the SPOT is “not ready for use,” the error message ems400 is displayed, then the failed resource is removed, and if this resource happens to be a SPOT, then the error message emsg407 is displayed. Now let us see what the messages emsg400 and emsg407 mean. Searching for these strings in the file /usr/lpp/ssp/bin/sminstmsgs.pl, we find the corresponding text:

```
emsg400 - 0016-400 resource apparently failed,
since the Rstate is not ready for use.

emsg407 - 0016-407 Refer to /tmp/spot.out.<pid>
for more debug information.
```

This is our situation!

Let us go back to the very first error message returned to the setup_server from mknimres:

```
resource apparently failed, since the Rstate is not 'ready for use'.
Refer to /tmp/spot.out.99999 for more debug information.
```

Now we know for sure that the resource that failed in creation was the SPOT.

The facts we have so far are:

- mknimres failed while creating the SPOT.
- There is a logfile /tmp/spot.out.99999.
- The Rstate of the SPOT was “not ready for use.”

Let us have a look at the SPOT logfile. It is created every time the SPOT is created, and it is located in /tmp/spot.out.<process_id>.

The next screen shows only a small part of the file that is important for our analysis, /tmp/spot.out.99999:

```
.....
     disable_err_sig
     nim_log: str=warning: 0042-001 m_instspot: processing error
     encountered on \"master\":
     0042-175 c_instspot: An unexpected result was returned by the
     \"/usr/sbin/installp\" command:
FAILURES
File sets listed in this section failed pre-installation verification
and will not be installed.
...
```

Now we know that the probable cause is the `installp` command and some files related to SPOT cannot be accessed.

The error messages showed us that the SPOT resource was "not ready for use." Since SPOT is created from lppsource, let us have a look at the status of lppsource.

Check to see if lppsource is ready for use. Run `lsnim -l lppsource_aix414` and you should get a screen like the following:

```
#lsnim -l lppsource_aix414
 lppsource_aix414:
     class       = resources
     type        = lpp_source
     server      = master
     location    = /spdata/sys1/install/aix414/lppsource
     alloc_count = 1
     Rstate      = ready for use
     prev_state  = unavailable for use
     simages     = yes
```

Rstate should be `ready for use` and the option *simages* should be yes. `Rstate = not ready for use` means that for some reason the lppsource is not available. Maybe there are missing file sets, or maybe some file sets are corrupt. Have a look at the /usr/lpp/bos.sysmgt/nim/methods/c_sh_lib file and check if your lppsource includes the file sets being stored in the variable `SIMAGES_OPTIONS`. This variable contains the names of all install images that should exist in the corresponding lppsource directory.

```
SIMAGES_OPTIONS="\
        bos \
        bos.adt \
        bos.info.any \
        bos.iconv \
        bos.net \
        bos.diag \
        bos.ifor_ls \
        bos.loc.iso \
        bos.msg.en_US \
        bos.powermgt \
        bos.sysmgt \
        bos.terminfo.all \
        bos.txt \
        devices.all \
        printers.rte \
        xlC.rte \
        X11.apps \
        X11.base \
        X11.compat \
        X11.Dt \
        X11.fnt \
        X11.loc.all \
        X11.motif \
        X11.msg.all \
        X11.vsm"
```

If SIMAGES_OPTIONS = no, it means that the required images for the support images to create the SPOT were not available in the lppsource resource. In this case, look at the /usr/lpp/bos.sysmgt/nim/methods/c_sh_lib file again. The images that are required for the creation of the SPOT resource are listed in the variable REQUIRED_SIMAGES:

```
     REQUIRED_SIMAGES="\
             bos \
             bos.net \
             bos.rte.up \
             bos.rte.mp \
             bos.diag \
             bos.sysmgt \
             bos.terminfo \
             bos.terminfo.all.data \
             devices.base.all \
             devices.buc.all \
             devices.graphics.all \
             devices.mca.all \
             devices.scsi.all \
             devices.sio.all \
             devices.sys.all \
             devices.tty.all"
```

Run `nim -o lslpp lppsource_aix414 |pg` and see if the required images exist in lppsource.

If you find missing file sets in lppsource, proceed with the following steps:

- Put the missing images in the corresponding lppsource directory.
- Run the `nim -Fo check lppsource_aix414` command.
- Run setup_server again.

(Anyway, if we had had problems with the lppsource Rstate and simages attributes, the ok_or_exit() function would have let us know that those attributes did not have the required values. But it is good to know where you can find the list of files that are involved in the creation process of objects such as lppsource or SPOT, and to know that these files can be flexible).

If you are just recreating the SPOT, you do not have to run the whole setup_server script. You can run only the part that makes the SPOT, using the wrapper mknimres: `mknimres -l 0`

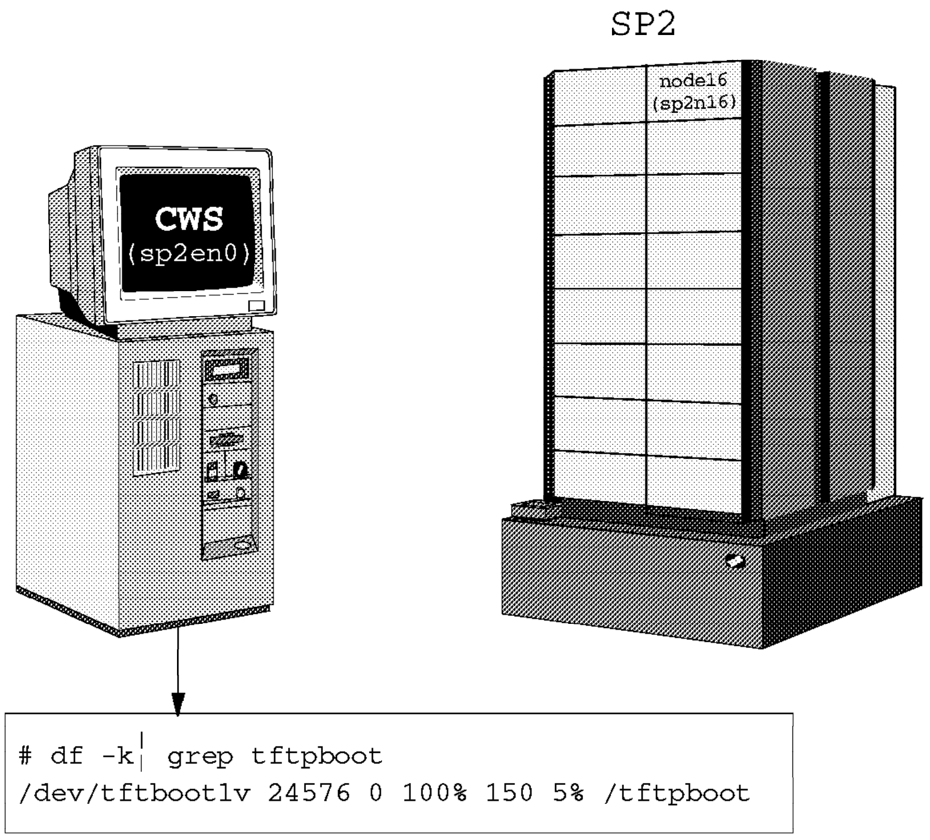
0 means you run it for the Control Workstation; the command `mknimres -l 16` creates all resource objects on the NIM master corresponding to the node number 16.

The checks of lppsource showed that it was in a correct state, that is, there were no file sets missing. So we should look for the reason for the failure of setup_server somewhere else. The /tmp/spot.out.99999 file points to the problems while using the m_instspot routine. What happens during the creation of a SPOT? We know that while a SPOT is being created, the /tftpboot directory is filled with the boot images created by NIM.

Check /tftpboot; is the directory there?

Check if there is enough free space available.

Conclusion:



```
# df -k| grep tftpboot
/dev/tftbootlv 24576 0 100% 150 5% /tftpboot
```

Figure 23. Result of the Problem Determination: Directory is Full

The problem in our case was that /tftpboot was 100% full and the NIM network install images could not be placed in this directory.

Solution:

Increase the size of the /tftpboot file system; if /tftpboot is a directory within the /, increase the size of the root file system:

```
chfs -a size=<number of blocks> /tftpboot
```

Scenario 4

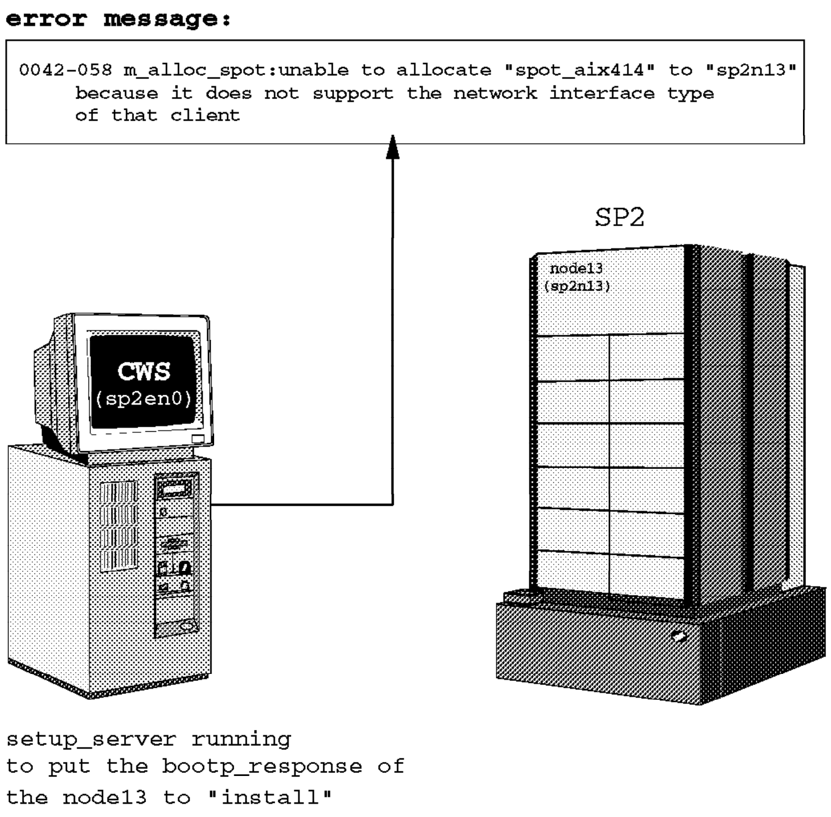
Symptom:

setup_server fails while allocating a resource.

Environment: Control Workstation at AIX 4.1.4 and PSSP 2.2.

Node to be installed: High Node

Problem Description:



```
0042-058 m_alloc_spot:unable to allocate "spot_aix414" to "sp2n13"
     because it does not support the network interface type
     of that client
```

```
setup_server running
to put the bootp_response of
the node13 to "install"
```

Figure 24. Setup_server Fails while Allocating a Resource

The Control Workstation was successfully migrated from PSSP 2.1 to PSSP 2.2.

All nodes are in a default partition, containing a mix of PSSP 2.1 and PSSP 2.2 nodes.

Node 13 (604) was added and was detected by the Control Workstation; appropriate SDR information was created. The procedures to obtain the Ethernet address, define SP Ethernet info, switch info, and so on, worked fine.

Finally, setup_server was run and the `allnimres` wrapper returned an error to setup_server.

Setup_server output:

```
0042-001 nim: processing error encountered on "master":
   0042-001 m_allocate: processing error encountered on "master"
   0042-058 m_alloc_spot: unable to allocate "spot_aix414" to "sp2n13"
        because it does not support the network interface type
        of that client
allnimres: 0016-251: Failure to allocate spot resource spot_aix414
from server node 0
```

Problem Determination:

It is obvious that the problem occurred while running the allnimres wrapper. What does allnimres do? It resets the node, deallocates all resources from this node and, last but not least, allocates to this node all NIM resources that are required according to bootp_response set to this node. The resource that caused problems during allocating was the SPOT resource, and the reason was that SPOT does not support "the network interface type of that client."

What is the state of our High Node? `lsnim -l sp2n13` shows the following:

```
sp2n13 :
   class       = machines
   type        = standalone
   platform    = rs6ksmp
   if1         = spnet_en0 sp2n16 10005AFA0AB5 ent
   cable_type1 = bnc
   prev_state  = ready for a NIM operation
   Mstate      = not running
   lpp_source  = lppsource_aix414
   control     = master
```

Of all the resources that should be allocated to the node having bootp_response set to "install," only lppsource has been allocated to Node 16.

Let us check first if the SPOT resource exists. The command `lsnim -l master` should show us if the master "serves" the SPOT resource:

```
master:
   class       = machines
   type        = master
   Cstate      = ready for a NIM operation
   reserved    = yes
   platform    = rs6k
   serves      = boot
   serves      = nim_script
   comments    = machine which controls the NIM environment
   Mstate      = currently running
   prev_state  = ready for a NIM operation
   if1         = spnet_en0 sp2en0.msc.itso.ibm.com 02608C2D4A7F
   master_port = 1058
   cable_type1 = bnc
   serves      = psspscript
   serves      = prompt
   serves      = noprompt
   serves      = migrate
   serves      = lppsource_aix414
   serves      = mksysb_1
   serves      = spot_aix414
   serves      = mksysb_2
   serves      = mksysb_3
```

SPOT seems to exist, and to be served by master. Let us get more information about the SPOT resource with the command `lsnim -l spot_aix414`:

```
spot_aix414:
   class        = resources
   type         = spot
   version      = 04
   alloc_count  = 0
   server       = master
   location     = /spdata/sys1/install/aix414/spot/spot_aix414/usr
   Rstate       = ready for use
   prev_state   = unavailable for use
   if_supported = rs6k ent
   release      = 01
   if_supported = rs6k fddi
   if_supported = rs6k tok
```

This output gives us more information about our problem. The network interfaces supported by this spot are only Ethernet/Token Ring/FDDI for the rs6k platform. What about our SMP node? If a client wants to use the SPOT resource, it must support the network interface type and the platform of the client′s primary interface. In our case, the network type is `ent` and the platform type is `rs6ksmp`, and our SPOT does not support them. A quick search in /tftpboot confirms that the following boot images are not there:

- spot_aix414.rs6ksmp.ent
- spot_aix414.rs6ksmp.tok

So our problem arises from missing definitions for the supported network type and platform for SMP nodes.

Which software component do we need to support `rs6ksmp tok` and `rs6ksmp ent`?

It is devices.rs6ksmp.base.

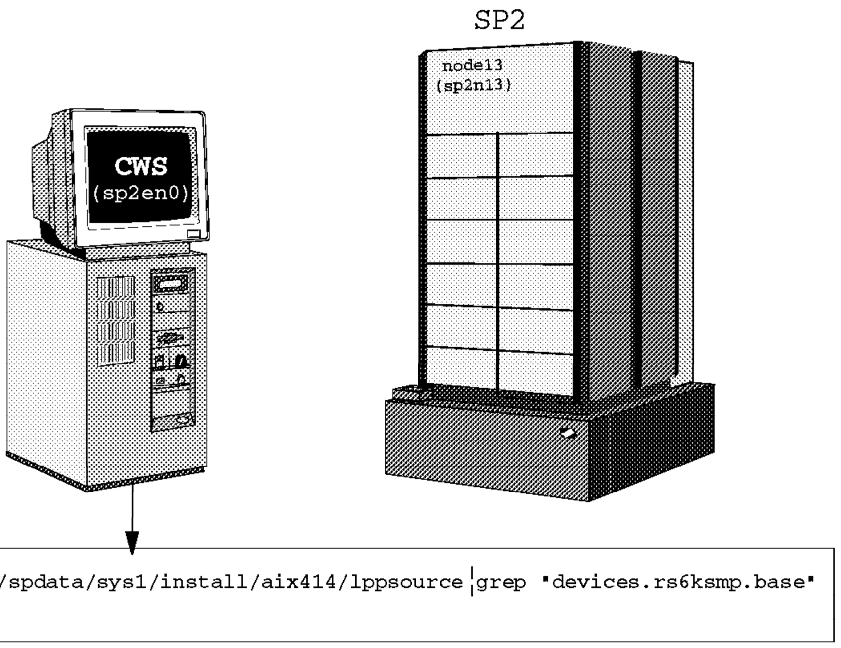
To make sure that our SPOT does not include this file set, enter:

```
nim -o lslpp spot_aix414 | grep "devices.rs6ksmp.base"
```

We did not have any output from this command, which means that spot_aix414 does not have the required file sets. What about lppsource_aix414? Let us search in the lppsource directory for the file set devices.rs6ksmp.base.usr.4.1.4.0.

Conclusion:

Figure 25 shows the result of our problem determination in this case.



```
# ls -l /spdata/sys1/install/aix414/lppsource|grep 'devices.rs6ksmp.base'
#
```

Figure 25. Result of the Problem Determination: Allocating Resources

The file set devices.rs6ksmp.base.usr.4.1.4.0 was missing in the /spdata/sys1/install/aix414/lppsource directory.

Solution:

1. Copy the devices.rs6ksmp.base.usr.4.1.4.0 file set to the lppsource_aix414 directory (if you copy it from another machine via NFS or FTP, do not forget to run `nim -Fo check lppsource_aix414` to refresh the lppsource NIM object).
2. Update your SPOT using:
   `nim -o cust -a lpp_source=aix414 -a fixes=update_all -F spot_aix414`
3. Set Node 16 to bootp_response = install and run setup_server again.

Another way to solve this problem:

1. Copy the missing devices.rs6ksmp.base.usr.4.1.4.0 file set to the /spdata/sys1/install/aix414/lppsource directory.
2. Run `delnimmast -l 0`.
3. Set Node 16 to bootp_response = install. (with the option `-s no`).

4. Run setup_server.

After performing these steps, the problem is solved.

Scenario 5

This scenario is not a typical case study for this chapter but it could be helpful for understanding the NIM problems that have gotten a bad reputation. Why is this case not a typical situation for this part of the redbook?

The NIM information is not set up correctly, but it is not bad enough to have setup_server notice it. So setup_server terminates with RC=0 and lets this problem be "hidden," because it is not "visible" until the node installation is done.

Symptom: The node was hanging with the LED 613 while performing a netboot.

Let us start the *problem determination* at this point: the LED code 613 reports having problems with routing. Let us have a look at the /sbin/rc.boot shell script:

```
.....
        5)      # Network boot
.....

                # set up route tables
                [ -n "${ROUTES}" ] && {
                        for route_args in $ROUTES
                        do
                                OIFS="$IFS"; IFS=':'
                                set -- $route_args
                                IFS="$OIFS"
                                # verify that there are 3 arguments
                                [ $# -ne 3 ] && continue
                           route -v add -net $1 -netmask $2 $3 ||
                                        loopled 0x613
                        done
                }

.....
```

The fact that we see routing problems is strange because we do not need any routing for the controlling Ethernet segment on the SP system; all nodes are in the same segment. Let us check the definition of spnet_en0 in the NIM database:

```
# lsnim -l spnet_en0
spnet_en0:
   class      = networks
   type       = ent
   net_addr   = 192.168.3.0
   snm        = 255.255.255.0
   Nstate     = ready for use
   prev_state = ready for use
   routing1   = spnet_tok0 sp2en0.msc.itso.ibm.com
```

This is the reason for our problem: the routing information for the spnet_en0 object.

The content of the /tftpboot/<hostname>.<domain_name>.info file confirms the routing problem. Look at the last line of the output:

```
#------------------ Network Install Manager ---------------
# warning - this file contains NIM configuration information
#       and should only be updated by NIM
export NIM_NAME=sp2n01
export NIM_HOSTNAME=sp2n01.msc.itso.ibm.com
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=sp2en0.msc.itso.ibm.com
export NIM_MASTER_PORT=1058
export RC_CONFIG=rc.bos_inst
export NIM_BOSINST_ENV="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_bosinst_env"
export NIM_BOSINST_RECOVER="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_bosinst_env
 -a hostname=sp2n01.msc.itso.ibm.com"
export SPOT=sp2en0.msc.itso.ibm.com:/spdata/sys1/install/aix42/spot/spot_aix42/usr
export NIM_BOSINST_DATA=/NIM_BOSINST_DATA
export NIM_CUSTOM="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script
 -a location=sp2en0.msc.itso.ibm.com:/export/nim/scripts/sp2n01.script"
export NIM_BOS_IMAGE=/NIM_BOS_IMAGE
export NIM_BOS_FORMAT=mksysb
export NIM_HOSTS=" 192.168.3.1:sp2n01.msc.itso.ibm.com  192.168.3.37:sp2en0.msc.itso.ibm.com "
export NIM_MOUNTS=" sp2en0.msc.itso.ibm.com:/spdata/sys1/install/aix42/lppsource:
 /SPOT/usr/sys/inst.images:dir  sp2en0.msc.itso.ibm.com:/spdata/sys1/install/pssp/bosinst_data:
 /NIM_BOSINST_DATA:file  sp2en0.msc.itso.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.42:
 /NIM_BOS_IMAGE:file "
export ROUTES=" 9.12.1.0:255.255.255.0:192.168.3.37 "
```

We see that this file contains the confusing routing information, and this file will be copied to /etc as the niminfo file for the node. So, performing the rc.boot, NIM takes the routing information and points to the IP address of the network where the NIM master should be found: 9.12.1.0. This IP address is not our Control Workstation′s address.

Having found the reason for our problem, the *solution* is rather simple:

- Set your node back to bootp_response=disk. This command cancels all resource allocations for this node that should be installed and removes its /tftpboot/<hostname>.<domain_name>.info file.

  Before we perform operations on a NIM spnet_en0 object, we must be sure that it is no longer allocated to any object.

- Remove the routing information from spnet_en0 and check this object again (you may use the `smitty nim.routing` command):

```
# nim -o change -a routing1='' spnet_en0
# lsnim -l spnet_en0
spnet_en0:
   class      = networks
   type       = ent
   net_addr   = 192.168.3.0
   snm        = 255.255.255.0
   Nstate     = ready for use
   prev_state = ready for use
```

- Set the bootp_response of the node to ″install″ and check the /tftpboot/<hostname>.<domain_name>.info file before you start with node conditioning.

```
#------------------ Network Install Manager ---------------
# warning - this file contains NIM configuration information
#       and should only be updated by NIM
export NIM_NAME=sp2n01
export NIM_HOSTNAME=sp2n01.msc.itso.ibm.com
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=sp2en0.msc.itso.ibm.com
export NIM_MASTER_PORT=1058
export RC_CONFIG=rc.bos_inst
export NIM_BOSINST_ENV="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_bosinst_env"
export NIM_BOSINST_RECOVER="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_bosinst_env
 -a hostname=sp2n01.msc.itso.ibm.com"
export SPOT=sp2en0.msc.itso.ibm.com:/spdata/sys1/install/aix42/spot/spot_aix42/usr
export NIM_BOSINST_DATA=/NIM_BOSINST_DATA
export NIM_CUSTOM="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script
 -a location=sp2en0.msc.itso.ibm.com:/export/nim/scripts/sp2n01.script"
export NIM_BOS_IMAGE=/NIM_BOS_IMAGE
export NIM_BOS_FORMAT=mksysb
export NIM_HOSTS=" 192.168.3.1:sp2n01.msc.itso.ibm.com  192.168.3.37:sp2en0.msc.itso.ibm.com "
export NIM_MOUNTS=" sp2en0.msc.itso.ibm.com:/spdata/sys1/install/aix42/lppsource:
 /SPOT/usr/sys/inst.images:dir  sp2en0.msc.itso.ibm.com:/spdata/sys1/install/pssp/bosinst_data:
 /NIM_BOSINST_DATA:file  sp2en0.msc.itso.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.42:
 /NIM_BOS_IMAGE:file "
```

The ROUTES information is not there any more and you can start with the installation of the node.

2.2.3.4 Hints for Solving NIM Operation Problems

1. Analyze the error messages and the log files.
2. Locate the point from where your problems started.
3. Locate the component of NIM that is causing problems.

   You may need to run a wrapper in the debug mode to get the exact point of failure and the environment (parameters, values of variables, and so on) used by the wrapper at that time.

   For more information about setting up a Perl script in the debug mode, refer to Appendix A, "Debugging Perl Scripts" on page 261.
4. If your problems are related to NIM resources, check the Rstate and simages attributes of these resources and of the resources that are involved in the operations you are performing.
5. Check if the resources are locked. If lpp_source has a residual NIM lock from some previous NIM operation, then the `lsnim` command should show a locked attribute for the resource:

   ```
   lppsource_aix415:
       class = resource
       type = lpp_source
       .......
       locked = <proces_group_id>
       .....
   ```

   A possible reason for lpp_source remaining locked is that multiple versions of setup_server are running. Check for this and for the existence of other NIM processes still running. If there are multiple setup_server processes, or other NIM processes, running (other than nimesis), kill them, remove the NIM locks, and start setup_server again. You should be able to remove NIM locks by stopping and restarting the nimesis daemon.
6. Check the permissions set for the components of the resources, and for the directories they are located in.
7. Check the structure of the directories where the resources reside.
8. Make sure that there is enough space in the file systems involved in the operations you are performing.
9. Check if there are file sets missing in lppsource. If yes:
   - Load them into the corresponding lppsource directory.
   - Run the `nim -Fo check lppsource_aix42` command.
   - Run setup_server again (or only the involved wrapper, depending on what you plan to do).
10. If you were pointed to some lppsource images causing problems, and they exist in the right lppsource directory, check if the critical images or files are corrupt or contain another version.
    - Run the `sum` command for the given file.
    - Run the `what` command for the given file.
11. Remember that problems caused by the mismatch between the states of lppsource and SPOT cannot be solved just by running setup_server or the mknimres wrapper again. As you can see in the make_spot() function, if the checks of the Rstate of the SPOTs involved in the performed operations pass

correctly, no action is taken on these SPOTs. This means that the SPOT will not be updated automatically! You have to do it manually. To do this, use the following command:

```
nim -o cust -a lpp_source=<lppsource_name> -a fixes=update_all \
         -F <spot_name>
```

Here is an example:

```
nim -o cust -a lpp_source=aix414 -a fixes=update_all -F spot_aix414
```

12. If you have problems related to the operations on NIM client objects, check the Cstate attribute of the client using the `lsnim -l <client_name>` command.

    The following screen shows the output of this command:

```
# lsnim -l sp2n16
sp2n16:
   class          = machines
   type           = standalone
   Mstate         = currently running
   prev_state     = BOS installation has been enabled
   Cstate         = ready for a NIM operation
   platform       = rs6k
   if1            = spnet_en0 sp2n16 10005AFA0AB5 ent
   cable_type1    = bnc
   cpuid          = 000204085700
   netboot_kernel = up
   Cstate_result  = success
```

    The NIM client may be in a state that conflicts with your intentions for the node.

    You may intend to install a node, but setup_server terminates with an error message saying that the `nim -o bos_inst ...` command failed for this client. While setup_server is running on the NIM master to configure this node and detects, for example, that the node is busy installing, setup_server cannot reconfigure the node. This situation could be caused by the machine being powered off during the netboot install of the node, before successful completion of the installation. This problem can be solved by using the following command:

```
nim -Fo reset <client_name>
```

    The NIM client definition information:

```
bootp_response      Cstate                  allocations
------------------------------------------------------------
install             BOS installation        spot_*
                    has been enabled        lppsource_*
                                            noprompt
                                            psspscript
                                            mksysb_*
------------------------------------------------------------
diag                diagnostic boot has     spot_*
                    been enabled            prompt
------------------------------------------------------------
maintenance         BOS installation        spot_*
                    has been enabled        prompt
------------------------------------------------------------
disk or customize   ready for a NIM
                    operation
```

13. It is possible that you have more than one problem during NIM operations. You probably cannot spend days analyzing all NIM problems and their

possible causes, so it may be better to just “clean up” the NIM and set it again. There are two ways to do this:

A quick way:

1. Run `delnimmast -l 0`.
2. Set a node to bootp_response=install (using the option `-s no`).[1]
3. Run setup_server.
4. Set a node to bootp_response=disk.

A systematic way:

1. Unexport all exported directories except lppsource and pssplpp using the following command repeatedly:

   ```
   rmnfsexp -d <directory name> -B
   ```

2. Reset all NIM clients using the following command repeatedly:

   ```
   nim -Fo reset <client name>
   ```

3. Deallocate all resources allocated to clients, using, for example:

   ```
   nim -o deallocate -a spot=spot_aix414 \
                     -a lpp_source=lppsource_aix414 \
                     -a mksysb=mkysb_2 ........ sp2n16
   ```

   (where sp2n16 is the NIM client object name for Node 16).

4. Remove the resources:

   ```
   nim -o remove spot_aix414
   ```

   ```
   nim -o remove lppsource_aix414
   ```

   ...

   and so on for all resources.

5. Unconfig NIM master and remove the NIM file sets:

   ```
   nim -o unconfig master
   ```

   ```
   installp -u bos.sysmgt.nim.master
   ```

   ```
   installp -u bos.sysmgt.nim.spot
   ```

   ```
   installp -u bos.sysmgt.nim.client
   ```

6. Set a node to boopt_response=install (using the option `-s no`).
7. Run setup_server.
8. Set a node to bootp_response=disk.

These steps may solve all your problems related to the NIM resources and objects.

[1] We use the option `-s no` for two reasons:

- We do not want to run setup_server on every boot/install server of the system.
- Maybe there are some tasks to be done on other components of PSSP software, for example Kerberos, and then setup_server will be started.

2.3 Node Installation/Customization

This is the final phase of the installation and migration process performed on the nodes. This is the stage when the system image, if installing or migrating AIX, is actually sent down the network and installed onto the node or nodes. Following this is, of course, the customization, when the PSSP software is installed and the SP environment set up.

Until now, at each stage, we have been entering, testing and initializing from the command line or through the System Management Interface Tool (SMIT). This has meant that if an error or warning message occurred, we knew which part required attention. At this point in the installation or migration process, however, without knowing what is happening under the covers, the only indication we have of a problem is from the LED codes displayed on the Graphical User Interface (GUI).

What we hope to do in this section is to break down the overall installation process into more manageable parts that can be more easily diagnosed. This will then hopefully give us, as before, an indication of the function or process that has detected the problem. We hope to accomplish this by explaining how the install process fits together and then examining the processes and their related files in more detail. In addition, examples are given to demonstrate how the results of particular checks or tests affect the path to problem resolution.

2.3.1 Basic Steps Performed during Install

The steps will be discussed in more detail later in 2.3.4, "What is Happening during netboot?" on page 136, but the basic process is as follows:

1. The node powers on and performs normal BIST checking. If the node stops at this point, while still in hardware checking, then this is likely a hardware problem. This is not covered in this book.
2. A network boot is started. This is started by the user; however, any RS/6000 system does its own hardware checks when powered on.
3. AIX is installed or migrated.
4. Customization takes place in the form of pssp_script. This is always started after a migration or install. However, if the node is set to "customize" after initial install, then pssp_script is called from the /etc/rc.sp script on startup.
5. The node reboots in normal mode.

2.3.2 General Symptoms of a Problem

As mentioned previously, the only real indication we have that a fault has occurred during the installation, migration or customization of a node is from the LEDs displayed on GUI. A point to bear in mind, though, when you think there is a problem, is that, as mentioned in the *Installation and Migration Guide*, GC23-3898, some codes are displayed for a lengthy period of time. This is accentuated by the number of nodes being installed concurrently.

So how and when can we tell when a hang has occurred?

If you are familiar with the install process on previous versions of PSSP, then it will be fairly obvious when a problem has occurred. If you are not familiar with the install process, then you have to check further on in this chapter as to what is currently being performed and make an informed decision as to how long the LED should be displayed. The only exception to this rule is that some LED codes have a specific meaning that an error has been detected. For example, LED 611 signifies that a non-zero return code was returned when attempting to mount a directory during Initial Program Load (IPL).

The last symptom that may be seen is a blank LED display. This, by itself, provides no information that might help determine the source of the problem. You need to note the LED displayed immediately prior to the blank display to be able to determine at which point corrective procedures should begin.

Some of the LED codes can be grouped into particular areas, which makes identification easier. Here are some of the common groups of LEDs:

LED 231 or 239 Attempting the initial stage of node conditioning.

LED 6xx Error codes displayed from the rc files while executing; some are for information and others for errors only.

LED uxx These LEDs are output from the customization script pssp_script. They are progress indicators, but will remain displayed if the running function fails to complete.

This is only a rough guide to what is happening during install, but it does provide some indication on how we can start to break the overall process down. See 2.3.4, "What is Happening during netboot?" on page 136 for a more complete overview of the install steps and the LEDs displayed.

2.3.3 Basics to Check before Proceeding

Check the output from the splstdata command to make sure the information displayed is consistent and correct. Detecting an input error at this stage saves a great deal of time by avoiding the actions necessary to backtrack down the install path. In this example, we see that Node 7 has been set to install with the AIX code level at 4.2:

```
-> splstdata -b
                 List Node Boot/Install Information

node#         hostname  hdw_enet_addr srvr     response           install_disk
      last_install_image   last_install_time  next_install_image lppsource_name
-------------------------------------------------------------------------------
    3 sp2n03            10005AFA082C    0         disk                  hdisk0
     bos.obj.ssp.414.ptf Wed_Jan_22_14:51:46 bos.obj.ssp.414.ptf         aix414
    4 sp2n04            10005AFA1DB8    0         disk                  hdisk0
     bos.obj.ssp.414.ptf Wed_Jan_22_15:56:33 bos.obj.ssp.414.ptf         aix414
    7 sp2n07            10005AFA13AF    0      install                  hdisk0
          bos.obj.ssp.42 Sat_Jan_25_22:49:56      bos.obj.ssp.42          aix42
    8 sp2n08            10005AFA1B12    0         disk                  hdisk0
          bos.obj.ssp.42 Sat_Jan_25_22:57:12      bos.obj.ssp.42          aix42
```

Figure 26. Checking the Response to nextboot

We need to make sure that all the correct files for Node 7 have been created in the /tftpboot directory:

```
-> ls -l /tftpboot/sp2n07*
-r--------   1 nobody   system       77 Jan 31 10:30 /tftpboot/sp2n07-new-srvtab
lrwxrwxrwx   1 root     system       32 Jan 31 10:31 /tftpboot/sp2n07.msc.itso.
ibm.com -> /tftpboot/spot_aix42.rs6k.up.ent
-rw-r--r--   1 root     system      123 Jan 31 10:31 /tftpboot/sp2n07.msc.itso.
ibm.com.config_info
-rw-r--r--   1 root     system     1279 Jan 31 10:31 /tftpboot/sp2n07.msc.itso.
ibm.com.info
-rw-r--r--   1 root     system      805 Jan 31 10:31 /tftpboot/sp2n07.msc.itso.
ibm.com.install_info
```

Figure 27. Checking the Existence of Files for Installation

Another common source of problems that have been encountered with PSSP 2.2 (that were not previously encountered) are the prerequisite software requirements for the High Availability Infrastructure. The main file set that is required is perfagent.server, because it is a prerequisite for ssp.ha and ssp.pman. Make sure that it is placed in the correct lppsource directory and is at the AIX level for the version you will be running. For example, AIX 4.1 requires perfagent 2.1.4.4, whereas AIX 4.2 requires perfagent 2.2.0.0. In the following we check for existence of this file in the lppsource listed from splstdata for Node 7:

```
-> pwd
/spdata/sys1/install/aix42/lppsource
-> ls -l perfagent*
-rw-r--r--   1 root     sys     3152896 Jan 25 16:11 perfagent.2.2
-rw-r--r--   1 root     sys     3111936 Jan 25 16:59 perfagent.server.2.2.0.6.bff
```

Figure 28. Checking Prerequisite Software in lppsource

The reason for this is that during customization, when pssp_script is run, the lppsource directory is mounted and this file set is explicitly loaded. Failure to find or install this file set will cause the installation procedure to fail, or only partial customization to occur, which will then cause problems later after installation appears to have completed.

2.3.4 What is Happening during netboot?

In an attempt to make diagnosing a problem easier, the first step is to know exactly what is happening under the covers. We offer here a step-by-step list showing the procedures that are executed when installing or migrating.

1. When a network boot is initiated, the nodecond script is executed to perform the network boot. This basically retrieves the default boot device information from the SDR, and matches it with available adapters.
2. A boot protocol (BOOTP) request is then issued to locate the remote boot image. The information for the node is held in /etc/bootptab, which contains the IP addresses and the location of the boot image. The boot image in /tftpboot is simply a link to the correct type of boot image for this node, for example, spot_aix42.rs6k.up.ent. LED 231 is displayed at this point.
3. Once the boot image has been found, it is then retrieved using the Trivial File Transfer protocol (TFTP).
4. Phase 1 of the system boot begins at this point. This is where the software booting starts and the hardware booting ends.
5. A simple shell (ssh) is now run, performing the same job as init, except purely in the RAM file system.
6. Now the rc.boot that was transferred across in the boot image is run. This performs several functions. First it runs ifconfig for the first time and sets up any routing required; LED 606 is displayed. The .info file is transferred and executed, and LED 608 is displayed. Next the SPOT is NFS mounted from the server, LED 610 is displayed, and, once successful, the rc.bos_inst file is executed. This is indicated by LED 612.
7. The primary function of rc.bos_inst in Phase 1 is to poll down the SCSI bus to determine what adapters are present. Those that have entries in the predefined database are configured, for example, the hard disks. Once this part has finished, control is passed back to rc.boot, which also completes Phase 1 of the boot.
8. We are now into Phase 2, and again ssh invokes the rc.boot script. This time rc.boot simply sets the boot device after install and again calls rc.bos_inst, which in turn calls a new function, bi_main.
9. The bi_main script is, as its name suggests, the main part for installing or migrating. Its basic steps are as follows:
   a. The configuration files, as well as bosinst.data and image.data, are restored. LED c40 is displayed at this point.
   b. Extract data files that would normally be on diskette or tape, identified by LED c42.
   c. Update CuAt with the value for the console as this is a no-prompt installation, LED c33.
   d. Check the validity of the bosinst.data file.
   e. Check the validity of the image.data file.
   f. Create the root volume group with the parameters specified in image.data, identifiable by LED c50.
   g. Restore mksysb or migrate. LED c54 is displayed for an extended period of time at this stage.

   h. Change the environment from memory (RAM) to the image just installed, LED c52.

   i. pssp_script, transferred during the installation, is now run locally on the node. Once this has completed, the bi_main script ends Phase 2 by rebooting the node.

10. Ascertain whether an install or a customize is being run. This defines how the client and master are determined.

11. Create the log /var/adm/SPlogs/sysman/$CLIENTN.config.log.$$. All output is now placed here.

12. Transfers and expands <node>.install_info file.
    LED u03 displayed.

13. Transfer the <node>.config_info file.
    LED u57 displayed.

14. Other files are transferred, for example, Kerberos files, tuning.cust, and script.cust.

15. Start source master and add entries into /etc/inittab for the console and Ethernet devices.

16. Determine the processor type.
    LED u51 is displayed if Uniprocessor.
    LED u52 is displayed if Multiprocessor.

17. Install the correct kernel type from lppsource, if not available.

18. Link the kernel, install additional required drivers, set the ipldevice variable (disk), and run the `bosboot` command against ipldevice. LED u55 is displayed if the bosboot fails. Of course, bosboot fails if any of the preceding actions fail.

19. Now load the prerequisite file sets, perfagent.server, bos.rte.tty, and devices.sio.sa.diag.

20. PSSP software can now also be loaded.
    LEDs u80 to u85, and u53 for ssp.ha.

21. Execute tuning.cust and script.cust.
    LEDs u86 and u87.

22. Change boot_response to disk in the SDR.

23. Clean temporary files and exit, LED blanked.

24. Control is now passed back to the bi_main script as previously mentioned, which executes the command `reboot -q`, and normal booting commences.

So now that we understand exactly how the install procedure works, we are able to define the possible areas of failure. The simplest of these is obviously the node conditioning. If the node passes the LED 231 value it has managed to find its boot install server and transfer the boot image. It has then subsequently passed control over to the ssh and the rc scripts.

The second area of failure is, as with any installation, the configuration of devices and installation of the mksysb image. A point to note here is the fact that the scripts involved, such as bi_main, are exactly the same as those used for normal tape or CDROM installation on standalone systems, and are therefore highly unlikely to be prone to undocumented bugs or problems. The problems that *can* occur show up when attempting to set up the NIM and NFS environments; after that, the problems are the same as with any install, for example, attempting to install a mksysb that is larger than the disk space available, or other common issues such as a corrupt mksysb or hardware failure.

The final area, which has exposed a large number of problems, is the execution of the customization script. This is where the software requires a large number of prerequisites to be correct, such as having all the device drivers for any hardware differences on the node from the machine on which the original image was created.

2.3.5, "How to Diagnose a Problem" will examine each area in more detail. In each case we will attempt to follow a logical path of determination, with explanations and examples where appropriate.

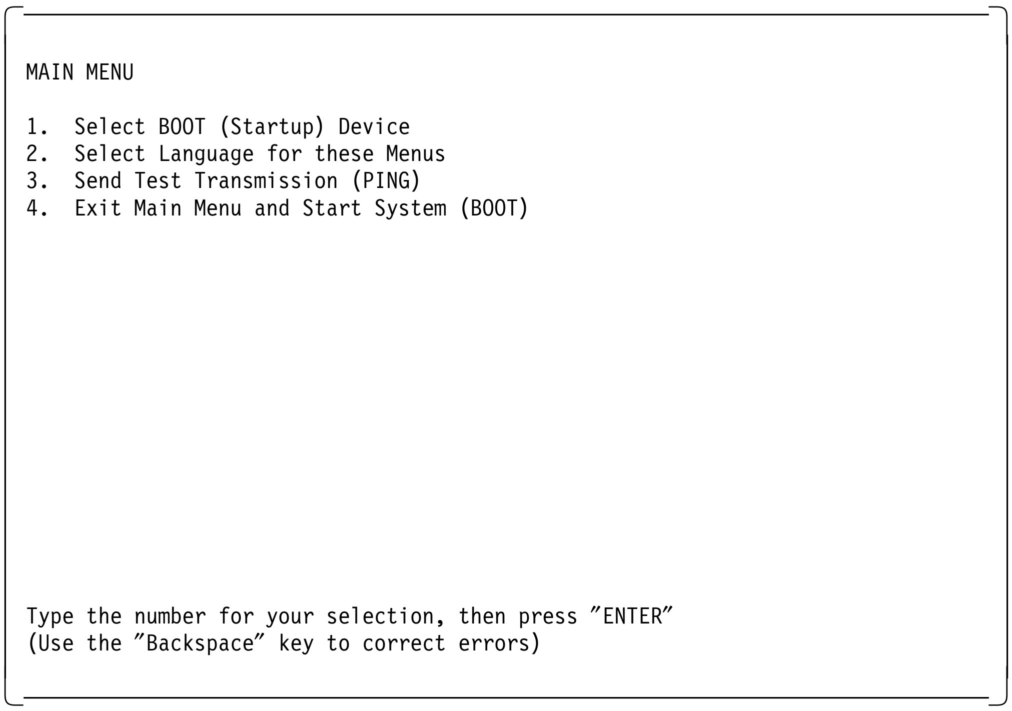
2.3.5 How to Diagnose a Problem

Having outlined the steps taken during installation or migration, we can now start down the road to diagnosing a problem that has arisen. Each step will be looked at individually, with one or two real-life problems and how they were solved. The idea here is to provide a starting point and a direction in which to go to search for causes. Solutions will hopefully be provided for any cases in which the error has already been experienced. However, it is not feasible to cover all possible errors in this book. The idea is, instead, as the title suggests, to offer a guide or technique for finding the cause.

Once a cause has been identified, you are half-way towards fixing the overall problem. The first step is, of course, knowing in which area the failure has occurred, which will hopefully be indicated by a halt at a particular LED. However, in some cases the LED may be blank, and as previously mentioned, we need to know the preceding codes to identify the stage of the install or migration we have reached.

2.3.5.1 Node Conditioning

The first area we will cover is the node conditioning stage of installation. There are not that many things that can go wrong when starting the network boot. The node is first placed in "secure" mode and IPLed, and then, once the LED reaches the 200 value, the key is switched to "service" and "reset." It then cycles through the LEDs again until reaching 260, which is the value at which the startup main menu is being displayed to the console, as shown:



```
MAIN MENU

1.  Select BOOT (Startup) Device
2.  Select Language for these Menus
3.  Send Test Transmission (PING)
4.  Exit Main Menu and Start System (BOOT)










Type the number for your selection, then press "ENTER"
(Use the "Backspace" key to correct errors)
```

Figure 29. Node Conditioning Main Menu

If the node hangs before this point, then there may well be a hardware problem. The best option then would be to power down the node, change the response to boot to “maintenance,” and test the hardware components.

The most typical problem seen at this stage is the LED 231 or LED 231-239 hang, indicating there was a problem in finding the boot install server, or in transferring the image.

So let us go through an example.

We attempted to initiate an install and it halted at the first stage with an LED 231. So where do we start? The first port of call would be to check the nodecond log located in /var/adm/SPlogs/spmon/nc on the Control Workstation. The logs use the naming convention of nc.<Frame_num>.<Slot_num>, and all messages are put here. The messages have the syntax `Nodecond Status: <message>`. If there are any error messages starting with `cannot determine...`, or `unknown...`, then this is likely to be a problem with retrieving information from the SDR. The SDR classes queried are the Node object class and Adapter object class, so check the information contained in these files with `SDRGetObjects`. An alternative method is to debug the script; refer to Appendix A, “Debugging Perl Scripts” on page 261 for information on how to find the exact command being run, whether it is `SDRGetObjects` or one of the hardmon commands (hmcmds).

So let us assume that the errors have been corrected or none existed in the first place. The next course of action is to establish where the error is occurring. Start up the node netbooting as you would normally when trying to install. Once the node has passed the secure phase of the boot, open up a read-only serial connection as shown in the following:

```
-> s1term 1 7
```

where the parameters are s1term <Frame#> <Slot#>. It is worth noting that you need to be in the correct partition for this node. You can now watch the nodecond script work its way through the Menus until it finally reaches the point of booting. The reason that all the IP addresses are cleared is so that the bootp request is issued to all addresses on the local network. At this point, you see one of two things: either the bootp packets are sent without any being received, or the TFTP packets are sent but not received:

```
STARTING SYSTEM (BOOT)



Booting . . .  Please wait.




Ethernet:  Built-In
Hardware address ........................................ 10005AFA17E3

              Packets Sent       Packets Received

BOOTP             00005              00000
```

Figure 30. Example of bootp Failure

In this example, we see that the bootp packets are being sent by the node, but no other system on the network has recognized this node and returned a bootp packet. A similar situation exists if there is a TFTP failure.

Let us first look at a bootp failure.

So what has happened? The adapter on the node has broadcast throughout the network, using its hardware address, for a bootp response and none of the other systems listening for bootp requests have recognized this system. Check the /etc/bootptab file for an entry for the node. We need to make sure that the entry is not commented out, that the hardware address of the adapter is correct, and that there are not multiple or erroneous entries. The entries at the end of the file should look similar to the following:

```
sp2n07.msc.itso.ibm.com:bf=/tftpboot/sp2n07.msc.itso.ibm.com:ip=192.168.3.7:ht=e
thernet:ha=10005AFA13AF:sa=192.168.3.37:sm=255.255.255.0:
```

Figure 31. Output from the bootptab File after Node Is Set to install

We can obtain the hardware address of the adapter by performing manual node conditioning. If this is a High Node, refer to Appendix D, “Manual Node Conditioning on a High Node” on page 281 for further information.

Select the boot device, enter the IP information, and make note of the hardware address of the adapter. Now start a ping test. If this fails, then either the IP information was entered incorrectly, or there is a network problem. If the ping succeeds, then we check again whether the bootptab information is correct. We can also recreate the information by setting the response to boot to “disk,” to clear the entry, and then back to “install” to recreate it.

Note

There may be a file called /etc/bootptab.info, which contains the hardware Ethernet addresses for the nodes. If this file exists, acquiring hardware address will use this file for all the nodes listed there, instead of doing a node conditioning which will bring the nodes down.

However, you must be carefull with this file, since it must be updated everytime an Ethernet adapter get changed or moved.

Now let us look at the TFTP problem. The node has found its boot install server with the bootp request, but has not been able to transfer the image. The failure looks almost identical to Figure 30 on page 140, except that this time the TFTP packets are not received. The first thing to check on the server is whether inetd and tftpd are active, using:

```
lssrc -ls inetd|pg
```

Next we need to check if the entry in /etc/inetd.conf is correct. It should look as follows:

```
tftp    dgram   udp     wait    nobody  /usr/sbin/tftpd tftpd -n
```

If the output matches this, then change user `nobody` to user `root`.

Important

This is purely for testing purposes in this part of the install. Changing this entry creates a security breach. It must be changed back to its original setting once testing is complete.

If the output does not match the sample, change it so that it does. After making changes to /etc/inetd.conf, you must refresh the daemon so that the changes take effect. Once you have refreshed inetd, netboot the node again and open up an s1term to see if any progress was made.

Depending on the results of these actions, we have two courses of action open to us. Either the changing to root user had no effect, in which case permissions are not the problem, or we progressed past the TFTP stage and LED 231, which means that a directory or file on the server has incorrect permissions. In *both cases* inetd.conf should be changed back to its previous state and another refresh of the inetd daemon initiated. If there is a permissions problem, then:

- Check the /tftpboot directory permissions and the files contained in it for the node. These should look like the following:

```
-> ls -ld /tftpboot
drwxrwxr-x   2 root     system      3072 Jan 31 13:39 /tftpboot
-r--------   1 nobody   system        77 Jan 31 11:36 /tftpboot/sp2n07.msc.itso.
ibm.com -> /tftpboot/spot_aix42.rs6k.up.ent
lrwxrwxrwx   1 root     system        32 Jan 31 13:38 /tftpboot/sp2n07.msc.itso.
ibm.com.config_info
-rw-r--r--   1 root     system       123 Jan 31 13:38 /tftpboot/sp2n07.msc.itso.
ibm.com.info
-rw-r--r--   1 root     system      1279 Jan 31 13:38 /tftpboot/sp2n07.msc.itso.
ibm.com.install_info
-rw-r--r--   1 root     system       805
```

Figure 32. Checking Permissions of Files in /tftpboot

- Check the /tftpboot directory itself:

  ```
  -> ls -ld /tftpboot
  drwxrwxr-x   2 root     system      3072 Feb 07 19:16 /tftpboot
  ```

- Check if the file system mount point is correct; /tftpboot resides under the root file system:

  ```
  -> ls -ld /
  drwxr-xr-x  26 bin      bin         1536 Jan 31 14:29 /
  ```

- Check for the existence of user nobody and the user's attributes in the /etc/passwd file:

```
root:!:0:0::/:/bin/ksh
daemon:!:1:1::/etc:
sys:!:3:3::/usr/sys:
adm:!:4:4::/var/adm:
uucp:!:5:5::/usr/lib/uucp:
guest:!:100:100::/home/guest:
nobody:!:4294967294:4294967294::/:
lpd:!:104:9::/:
```

Figure 33. Output from the /etc/passwd File

If the change to root had no effect, then the problem may be more difficult to find. We can check items of the following type:

- Check if access is still granted to the /tftpboot directory:

```
-> pg /etc/tftpaccess.ctl
# PSSP and NIM access for network boot
allow:/tftpboot
allow:/usr/lpp/ssp
allow:/etc/SDR_dest_info
allow:/etc/krb.conf
allow:/etc/krb.realms
(EOF):
```

Figure 34. Checking if TFTP Access Is Granted

- Check the size and permissions of the command, daemon and link. These may differ slightly in different system versions but the permissions, owner and group should all remain the same.

```
-> ls -l /usr/bin/tftp
-r-sr-xr-x   3 root    system   29084 Apr 12 1996  /usr/bin/tftp
-> ls -l /usr/sbin/tftpd
-r-xr-xr-x   1 root    system   21420 Apr 12 1996  /usr/sbin/tftpd
-> ls -l /etc/tftpd
lrwxrwxrwx   1 root    system      15 Jan 24 18:37 /etc/tftpd -> /usr/sbin/tftpd
```

Figure 35. Verifying Permissions and Links for TFTP

- Check the level of bos.net.tcp.client; enhancements may be available to the base level file set:

```
-> lslpp -l bos.net.tcp.client
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.net.tcp.client       4.1.4.16  COMMITTED  TCP/IP Client Support

Path: /etc/objrepos
  bos.net.tcp.client       4.1.4.16  COMMITTED  TCP/IP Client Support
```

Figure 36. Listing the Level of the TCP Client

Note

It is advisable to use the later enhancements of a file set, as these contain fixes for the reported problems at that time. Any problem that you may run into may already be reported and corrected in an enhancement to the base file set. These fixes can be obtained through local support centers or from anonymous FTP servers using the FixDist package to download. See also Appendix B, “How to Get FixDist” on page 275.

2.3.5.2 Installing AIX over the Network

The next possible failure could be the actual install of AIX onto the node. This is not common, and the main point here will be to ascertain whether the problem is hardware or software related. If the problem looks hardware related, an engineer should be called.

How can we identify what is going wrong? The LED values seen from now until pssp_script runs indicate that the rc scripts are running in memory. The potential areas of failure are the initial setup of the network card and the transfer of the files required to start the install. Once the restoration of the data files and the mksysb image begin, identified by LED cxx codes, then problem determination procedures for normal install failures of standalone systems should be followed. This is because, as mentioned, the same rc scripts that are used to install standalone systems are used here, the only difference being that the environment sources and variables are set up by NIM.

The problem we have here is that the scripts are being run locally on the node in memory, meaning that we have not configured any disks. This, of course, means that no disk space is available to output errors to files. Given this situation, the only indication of exactly what is going on is from the LED codes displayed. To get further information on the exact parameters being passed to the commands, we can set NIM into debug mode (see Appendix C, “Setting NIM to Debug Mode” on page 277 on how to do this).

Putting NIM into debug mode is the main tool we have for finding out exactly what is being attempted. The output is displayed line by line with the command and variables or parameters being used, as you will see in the examples later in this section. The NIM debug tool is used in most cases in the following steps, so it may be a good idea to familiarize yourself with the steps involved and the type of output seen.

Let us now look at each step performed at the beginning of the rc.boot script in more detail.

Before any transfer can occur, the network adapter must be set up. This is done by reading the bootinfo information passed during the previous transfer stages, contained in the bootpd daemon reply packet. Next, `ifconfig` is called to add the adapter and routes according to this information. The LED codes 606 or 607 are displayed if a failure occurs. Here is an example from NIM in debug mode showing how the information is read from the bootinfo information:

```
+ /usr/lib/methods/showled 0x600
 showled LED{600}
+ 1> /usr/lib/libs.a
+ bootinfo -c
+ set -- 192.168.4.1 192.168.4.137 192.168.4.138 68 22 0 /tftpboot/sp21n01.msc.i
tso.ibm.com 99.130.83.99.1.4.255.255.255.0.255.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0. 7
+ CLIENT_IPADDR=192.168.4.1
+ BOOT_SERV_IP=192.168.4.137
+ BOOT_GATE_IP=192.168.4.138
+ E802=0
+ BOOTFILE=/tftpboot/sp21n01.msc.itso.ibm.com
+ VEND=99.130.83.99.1.4.255.255.255.0.255.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
+ [ -n 255.255.255.0 ]
+ SUBMASK=netmask 255.255.255.0
+ [ 192.168.4.138 = 0 -o 192.168.4.138 = 192.168.4.137 ]
```

Figure 37. NIM Debug Output Showing Variables Being Set from bootinfo -c

If the information returned by the `bootinfo` command for the IP addresses is correct, then there may well be a hardware error. We need to verify that the Ethernet adapter is working correctly either by running diagnostics or contacting a hardware engineer.

Important

A point to note is that NIM installation does not support the 2992 and 2993 or the 100Mb Ethernet install adapters for installing. Trying to install with one of these adapters may well cause an LED 606 hang.

After defining and configuring the network adapter, the rc.boot script now uses TFTP to transfer across /tftpboot/<node>.info, located on the Control Workstation. LED 608 is displayed until the file transfer has completed.

Let us assume that the install has halted and an LED 608 has occurred. We know the info file is being transferred, and we also know that TFTP is working because just prior to this step the bootimage was transferred. However, double-checking would be a good idea, and the first stage of the checking process is to put NIM into debug mode to see the exact TFTP command being run:

```
+ /usr/lib/methods/showled 0x608
 showled LED{608}
+ tftp -g /SPOT/niminfo 192.168.4.137 /tftpboot/sp21n01.msc.itso.ibm.com.info im
age
```

Figure 38. NIM Debug Output Showing TFTP in Progress

We see that we are trying to transfer the file sp21n01.msc.itso.ibm.com.info from the /tftpboot directory located on the server of IP address 192.168.4.137 to /SPOT/niminfo. Provided the IP address is correct, we then need to check if the correct file exists with the correct permissions (see Figure 32 on page 142 for the correct permissions).

The transferred file is then executed to expand all the NIM variables required for installation. If this fails, LED 609 is displayed. Check if the file /tftpboot/<node_name>.info looks something like the following:

```
#------------------ Network Install Manager ---------------
# warning - this file contains NIM configuration information
#       and should only be updated by NIM
export NIM_NAME=sp2n07
export NIM_HOSTNAME=sp2n07.msc.itso.ibm.com
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=sp2en0.msc.itso.ibm.com
export NIM_MASTER_PORT=1058
export RC_CONFIG=rc.bos_inst
export NIM_BOSINST_ENV="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_bosinst_env"
export NIM_BOSINST_RECOVER="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_bosinst_en
v -a hostname=sp2n07.msc.itso.ibm.com"
export SPOT=sp2en0.msc.itso.ibm.com:/spdata/sys1/install/aix42/spot/spot_aix42/u
sr
export NIM_BOSINST_DATA=/NIM_BOSINST_DATA
export NIM_CUSTOM=" /../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script -a location
=sp2en0.msc.itso.ibm.com:/export/nim/scripts/sp2n07.script"
export NIM_BOS_IMAGE=/NIM_BOS_IMAGE
export NIM_BOS_FORMAT=mksysb
export NIM_HOSTS=" 192.168.3.7:sp2n07.msc.itso.ibm.com  192.168.3.37:sp2en0.msc.
itso.ibm.com"
export NIM_MOUNTS=" sp2en0.msc.itso.ibm.com:/spdata/sys1/install/aix42/lppsource
:/SPOT/usr/sys/inst.images:dir  sp2en0.msc.itso.ibm.com:/spdata/sys1/install/pss
p/bosinst_data:/NIM_BOSINST_DATA:file  sp2en0.msc.itso.ibm.com:/spdata/sys1/inst
all/images/bos.obj.ssp.42:/NIM_BOS_IMAGE:file"
```

Figure 39. Checking the Contents of the /tftpboot/<node>.info File

After successful transfer and execution of the niminfo file, we then attempt to NFS-mount the SPOT, along with the other mounts specified in the info file previously transferred. LED 610 is displayed while this happens, or LED 611 if an error is detected. If the install stops on LED 610, then some sort of hang has occurred because a return code has not been processed from the attempted mount. In such cases the NFS daemon may have a problem. First check the error log on the Control Workstation. There may be error entries for nfsd; the following is one from a typical NFS hang with LED 610:

```
------------------------------------------------------
LABEL:          SRC
IDENTIFIER:     E18E984F
Date/Time:       Fri Jan 17 11:10:34
Sequence Number: 489
Machine Id:      00009736700
Node Id:         sp21en0
Class:           S
Type:            PERM
Resource Name:   SRC

Description
SOFTWARE PROGRAM ERROR

Probable Causes
APPLICATION PROGRAM

Failure Causes
SOFTWARE PROGRAM

        Recommended Actions
        PERFORM PROBLEM RECOVERY PROCEDURES

Detail Date
SYMPTOM CODE
         256
SOFTWARE ERROR CODE
       -9035
ERROR CODE
           0
DETECTING MODULE
'srchevn.c'2line:'166'
FAILING MODULE
nfsd
------------------------------------------------------
```

Figure 40. Sample errlog Output for an NFS Hang

In this case there was obviously a problem with the CWS nfsd. On checking the status with `lssrc -g nfs`, we saw the following output:

```
-> lssrc -g nfs
Subsystem        Group            PID     Status
 biod            nfs              5136    active
 nfsd            nfs              7714    inoperative
 rpc.mountd      nfs              9780    active
 rpc.statd       nfs              10040   active
 rpc.lockd       nfs              10300   active
```

Figure 41. Listing the Status of the nfsd Daemon

To correct the problem, we tried to refresh NFS by entering the following commands:

```
stopsrc -g nfs
stopsrc -g portmap
startsrc -g portmap
startsrc -g nfs
```

However, the nfsd daemon still listed as inoperative, so we had to reboot the Control Workstation to correct the inoperative status. Once nfsd was active again, we were able to progress past LED 610.

The more common scenario at this stage is that an error is returned and LED 611 is displayed.

Important

A problem with NFS was introduced with bos.net.tcp.client 4.1.4.13, which caused an LED 611 error. If you are working with AIX 4.1, check that the file set level installed on the CWS and in the SPOT is not at this level. To check the spot, run the command
`nim -o lslpp -a lslpp_flags="-al bos.net.tcp.client" <SPOT>`
If either the Control Workstation or the SPOT have this level of bos.net.tcp.client, you should either obtain a later level or, if time is a factor for the install, return to the base level file set by deinstalling the updates.

To update the SPOT, use the following steps:

1. Issue `smitty nim_res_op`
2. Select the appropriate SPOT
3. Select update_all
4. Enter the source directory where the update is located

What we are actually doing at this stage is mounting all the files and images required to perform the installation or migration and customization. Again, as previously stated, the ideal method to find the failing mount is to set up NIM in debug mode and capture the output. However, since this takes some time, here are the related areas that can be examined prior to setting NIM to debug:

- Use `lsnim -Fl <client>` to check if the NIM client machine has the exported directories listed. This should list the client attributes. We need to check for the *exported* stanza:

```
-> lsnim -Fl sp21n01 | grep exported
   exported      = /spdata/sys1/install/aix414/spot/spot_aix414/usr
   exported      = /spdata/sys1/install/aix414/lppsource
   exported      = /spdata/sys1/install/pssp/bosinst_data
   exported      = /spdata/sys1/install/pssp/pssp_script
   exported      = /spdata/sys1/install/images/mksysb_test
   exported      = /export/nim/scripts/sp21n01.script
```

- Correlate this exported output with that from the `exportfs` command:

```
-> exportfs | grep sp21n01
/spdata/sys1/install/pssplpp                 -ro,root=sp21n01.msc.itso.ibm.c
om:sp21n05.msc.itso.ibm.com:sp21n06.msc.itso.ibm.com:sp21n07.msc.itso.ibm.com:sp
21n08.msc.itso.ibm.com
/spdata/sys1/install/aix414/spot/spot_aix414/usr -ro,root=sp21n01.msc.itso.ibm.c
om,access=sp21n01.msc.itso.ibm.com
/spdata/sys1/install/aix414/lppsource        -ro,root=sp21n01.msc.itso.ibm.c
om,access=sp21n01.msc.itso.ibm.com
/spdata/sys1/install/pssp/bosinst_data       -ro,root=sp21n01.msc.itso.ibm.c
om,access=sp21n01.msc.itso.ibm.com
/spdata/sys1/install/pssp/pssp_script        -ro,root=sp21n01.msc.itso.ibm.c
om,access=sp21n01.msc.itso.ibm.com
/spdata/sys1/install/images                  -root=risc38:sp21n01.msc.itso.i
bm.com,access=risc38:sp21n01.msc.itso.ibm.com
/export/nim/scripts/sp21n01.script           -ro,root=sp21n01.msc.itso.ibm.c
om,access=sp21n01.msc.itso.ibm.com
```

Important

There is a documented APAR, IX64998, that explains another known cause of an LED 611 problem. The error is seen when an AIX 4.2 mksysb is allocated to multiple clients. The solution is to make sure that the root option is added in /etc/exports and /etc/xtab.

- Verify that the directory <spot_location>/sys/inst.images is *not* a linked directory:

```
-> ls -l /spdata/sys1/install/aix414/spot/spot_aix414/usr/sys/
total 16
drwxr-xr-x   4 bin      bin          512 Jan 24 14:45 inst.data
drwxr-xr-x   2 bin      bin          512 Jan 24 14:45 inst.images
```

- Check if the image file is linked to the correct boot image file:

```
-> ls -l /tftpboot/sp2n07.msc.itso.ibm.com
lrwxrwxrwx   1 root     system        32 Feb 06 17:44 /tftpboot/sp2n07.msc.itso.
ibm.com -> /tftpboot/spot_aix42.rs6k.up.ent
```

- Clean up the NIM setup and exported directory, if they are in an undetermined state, with the following steps:

  1. Remove entries from /etc/exports with:

     ```
     /export/nim/scripts/*
     /spdata/*
     ```

  2. Clear up NFS-related files in /etc:

     ```
     rm /etc/state
     rm /etc/sm/* /etc/sm.bak/*
     ```

  3. Unconfigure and reconfigure NIM:

     ```
     nim -o unconfig master
     installp -u bos.sysmgt.nim.master
     ```

  4. Set the node or nodes back to “install” and run setup_server. This will also reinstall NIM:

     ```
     spbootins -r install -l <node#>
     ```

  5. Refresh the newly created exports list:

     ```
     exportfs -ua
     exportfs -a
     ```

6. Refresh NFS:

```
stopsrc -g nfs
stopsrc -g portmap
startsrc -g portmap
startsrc -g nfs
```

If, after following these check points and actions, no progress has been made, then the only option left is to set NIM into debug and track down the mount that is failing.

2.3.5.3 The bi_main Script

After all the mounts have been completed successfully, the bi_main script is invoked and the installation of mksysb should commence. Problems at this stage, after the NIM directories have been mounted and before pssp_script is called, are the same as when installing standalone systems. Therefore, problem determination procedures are largely the same.

The starting point for determining a problem with an LED hang, usually cxx or 623, would be to open a console with `s1term`, as mentioned previously. This shows, through normal system messages, the point to which the install has progressed and any errors that normally are echoed to the standard output.

Although we will not look at determination procedures in detail, here is an example to demonstrate how to obtain the system information to proceed.

So what steps can we take to start to determine where the install problem has occurred? We can see the approximate point with the `s1term` output, but we require more detail.

Once the bi_main script is called to install or migrate the AIX operating system, it logs output locally to the node in a file called /var/adm/ras/bi.log. This is where we should look if we encounter an LED 623 or LED cxx halt.

Note

LED 623 is a NIM fatal error indicator, telling us that the BOS installation has encountered a fatal error. By itself it gives us no indication as to what was happening. We need to find the LED code that immediately precedes LED 623.

The first thing we need to do is to set the response to boot for the node to “maintenance” with `spbootins`, and then perform a network boot.

Once the node has booted we are presented with the maintenance menu and need to access the root volume group, as follows:

1. Start maintenance mode for system recovery.
2. Access a root volume group.
3. Select the correct disk that rootvg was installed on.
4. Access this volume group and start a shell.

Let us take a look at an example where an install of a node has stopped with LED 623. As previously mentioned, we want to know the preceding LED code. On closer examination, the code immediately prior to 623 is LED c50. We now take the first steps, that is, setting the node to maintenance, netbooting, and importing rootvg. After accessing the root volume group, we can examine the log file for errors, as in the following example:

```
Preparing target disks.
 0516-064 lcreatevg: Unable to write the volume group descriptor area
         or status area. Check the state of the physical volume.
 Run diagnostics if necessary.
```

Figure 42. Error Listed in /var/adm/ras/bi.log after LED c50 and 623 Halt

We see that the recommendation is to test with diagnostics. We can first provide a further test by trying to manually execute the command running at the time of failure. In this case, `lcreatevg` is a low level Logical Volume Manager (LVM) command called by mkvg.

```
-> mkvg -y rootvg hdisk0
0516-059 lcreatevg: Unable to read physical volume ID or the physical
        volume may not have an ID.
0516-862 mkvg: Unable to create volume group.
```

We see that there are definitely some problems with the disk. The next stage would be to either run standalone diagnostics against the disk and associated devices, for example the SCSI adapters, or to contact a hardware engineer.

2.3.5.4 Node Customization

Following the successful installation of AIX, we proceed to customization in the form of pssp_script. Many of the problems encountered during installation or migration arise during this step, because any discrepancies in the System Data Repository (SDR) or Network Installation Manager (NIM) setup that have passed previous testing come to light here.

At this stage, as with the setting up of the NIM environment, each LED is quite specific to a particular task. Another advantage at this stage is that AIX has now been loaded onto the node. Since pssp_script is transferred across during installation and then run locally on the node, it is able to write its output to a file, namely the file /var/adm/SPlogs/sysman/<node>.config.log.<pid>, which is an obvious place to start if customization or installation problems occur.

Another advantage of having AIX on the node is that when customization fails, then, where cases permit, we can simply manually transfer the script over to the node and run it. This may not always be possible, as we will see in the LED u03 example If we can correct the problem without reinstalling AIX this will save us from having to pass through the entire installation procedure again.

Within the customization script there are three main stages where failures or problems can occur. These are:

- Transfer of the required files:

  <node>.install_info

  <node>.config_info

  /etc/SDR_dest_info

The Kerberos files /etc/krb.conf, /etc/krb.realms, and /etc/krb-srvtab

- Processor-specific customization

  By this we mean that a mksysb that was originally taken on a uniprocessor (UP) system can be installed on a multiprocessor system (MP), but the correct kernel must be loaded and linked.

- Installation of the correct level of PSSP from the boot install server

Let us look at the first stage of customization.

This is where the customization script has halted with an LED code informing us that it was attempting to retrieve one of the files required from the boot install server with TFTP. The LED codes are:

LED u03	Transferring the <node>.install_info file.
LED u04	Expanding install_info variables and transferring the template files required by the config_info file.
LED u57	Transferring the <node>.config_info file.
LED u61	Transferring the /etc/SDR_dest_info file.
LED u67	Transferring the /etc/krb.conf file.
LED u68	Transferring the /etc/krb.realms file.
LED u69	Transferring /tftpboot/<node>-new-srvtab to /tmp/krb-srvtab, changing permissions and moving to the /etc directory.

How can we find out what has gone wrong if we have a halt with one of these LEDs?

The first step is to check the output file created by pssp_script on the node where it executes. The file is contained in the directory /var/adm/SPlogs/sysman and has the naming convention of <node_name>.config.log.<PID>. To make sure that pssp_script echoes all the actions and parameters being used, add the line `set -x` to the script just after the "Main" stanza begins. This puts the output into verbose mode. The full pathname on the Control Workstation is /spdata/sys1/install/pssp/pssp_script. Here is an example of an edited pssp_script file:

```
fi

}

# main Main MAIN
set -x

# Move the NIM firstboot file
/bin/mv /etc/firstboot /etc/fb.NIM.ignore.$$ 2>/dev/null
```

Figure 43. The pssp_script after Setting to verbose Mode

Restart the customization to log the new verbose information in the log file, if this is an install or migration that will have to be restarted to call pssp_script, since we do not know at this point what caused the halt. Because AIX has now been loaded onto the node, we are able to boot the node into “maintenance” and look at the output file.

After examining the file we have a clearer view of the values of the variables or parameters passed to the command being executed. This will become clearer with the following example. We can then use this information to examine the files, daemons and other components that have an effect on this procedure.

In this example, the node has halted at the first point in the customization phase with an LED u03 (during an initial install). When monitoring the progress of the install with an open console terminal, the screen simply shows:

```
Filesets processed:  4 of 7  (Total time:  12 secs).

installp:  APPLYING software for:
        devices.mca.8efc.rte 4.1.4.0
        devices.mca.8efc.diag 4.1.4.0
        devices.mca.8efc.com 4.1.4.0


  . . . . << Copyright notice for devices.mca.8efc >> . . . . . . .
 Licensed Materials - Property of IBM

 576539300
   (C) Copyright International Business Machines Corp. 1985, 1995.

 All rights reserved.
 US Government Users Restricted Rights - Use, duplication or disclosure
 restricted by GSA ADP Schedule Contract with IBM Corp.

  . . . . << End of copyright notice for devices.mca.8efc >>. . . .

Finished processing all filesets.  (Total time:  16 secs).

         96              27      Network Install Manager customization.
```

Figure 44. Ouput to Console with LED u03 on 3-digit Display

The first number is the percentage of installation completed; the second number is the time elapsed in minutes, which increases continuously.

Now we powered off the node, set the response to boot to be “maintenance” with the command `spbootins -r maintenance -l 6`, and then netbooted the node again.

When the node reaches the maintenance screen, a console is opened automatically for user input. We chose the following options to gain access to the root volume group so that we could view the output log:

1. Start maintenance mode for system recovery.
2. Access a root volume group.
3. Select the disk that rootvg was installed on.
4. Access this volume group and start a shell.

Having successfully imported the root volume group, we were then able to look at the configuration log file located in the /var/adm/SPlogs/sysman directory. This contained the following output:

```
+ exec
+ 1>& 3
+ date
+ echo Customization for risc38.msc.itso.ibm.com starting at Thu Jan  1 00:20:08

 UTC 1970
Customization for risc38.msc.itso.ibm.com starting at Thu Jan  1 00:20:08 UTC 19
70onfig.log.1620 (0%)
+ echo

+ /etc/methods/showled 0xc03
+ tftpfile /tmp/risc38.msc.itso.ibm.com.install_info 192.168.3.37 /tftpboot/risc
38.msc.itso.ibm.com.install_info
Getting /tftpboot/risc38.msc.itso.ibm.com.install_info from 192.168.3.37
tftp: sendto: No route to host
rm: /tmp/risc38.msc.itso.ibm.com.install_info: No such file or directory
Getting /tftpboot/risc38.msc.itso.ibm.com.install_info from 192.168.3.37
tftp: sendto: No route to host
rm: /tmp/risc38.msc.itso.ibm.com.install_info: No such file or directory
.
.
```

Figure 45. Output from the Configuration Log on the Node

We see from this that the node was attempting to transfer the install_info file for the machine risc38 from a server of IP address 192.168.3.37. The machine risc38 was the system the mksysb image was originally created on and server address 192.168.3.37 was not the address of the Control Workstation. So from where was the false information being generated?

We ran through the /spdata/sys1/install/pssp/pssp_script script to find the point of failure. This showed that TFTP was in progress, with the command:

```
tftpfile /tmp/$INSTFILE $NIM_MASTER $INSTALLFILE
```

In this command, `$INSTFILE` is the target file on the client. The `$NIM_MASTER` attribute is set at the beginning of pssp_script, depending on the mode the script is being called with, that is, either “install” or “customize.” In the following we see the excerpt from pssp_script that determines how the variables are defined:

```
if [[ $NIM_CUSTOM != ″″ ]]
then
        mode=″install″
else
        mode=″customize″
fi

if [[ $mode = ″install″ ]]
then
#Setup IP address of the NIM Master
if [[ -f /etc/niminfo ]]
        /bin/cp /etc/niminfo /tmp/niminfo 2>/dev/null
        chmod +x /tmp/niminfo
        . /tmp/niminfo 2>/dev/null
fi

# Does this system have a valid NIM client /etc/niminfo file
if [ ″$NIM_CONFIGURATION″ = ″standalone″ ]
then
        # Yes - use the /etc/niminfo file
        NIM_MASTER=/bin/echo $NIM_HOSTS | cut -f2 -d ″ ″ | cut -f1 -d:
        NIM_MASTER_HN=/bin/echo $NIM_HOSTS | cut -f2 -d ″ ″ | cut -f2 -d:
        NIM_MASTER_SHN=/bin/echo $NIM_MASTER_HN | /bin/cut -d″.″ -f1
        /bin/rm /tmp/niminfo
        CLIENTN=$NIM_HOSTNAME
else
        # No - must be a NIM master - use /etc/ssp/server_name
        # to determine NIM master address
        NIM_MASTER=/bin/cat /etc/ssp/server_name | cut -f1 -d ″ ″
        NIM_MASTER_HN=/bin/cat /etc/ssp/server_name | cut -f2 -d ″ ″
        NIM_MASTER_SHN=/bin/cat /etc/ssp/server_name | cut -f3 -d ″ ″
        NIM_NAME=/bin/hostname -s
        CLIENTN=/bin/cat /etc/ssp/reliable_hostname | cut -f2 -d ″ ″
fi
```

Figure 46. Excerpt from pssp_script Where Variables Are Set in Install

We see that the variables on install should be taken from the variables contained in the /etc/niminfo file or from the /etc/ssp directory.

The $INSTALLFILE variable is also determined in a similar way:

```
INSTALLFILE=/tftpboot/$CLIENTN.install_info
```

Once again, the $CLIENTN variable is also set at the beginning of pssp_script either by /etc/niminfo or from the /etc/ssp/reliable_hostname file.

We checked for the existence of the /etc/ssp directory. As expected, it didn′t exist, basically because it is created later on during pssp_script. The next step then was to look at the /etc/niminfo file:

```
#------------------ Network Install Manager ---------------
export NIM_NAME=risc38
export NIM_HOSTNAME=risc38.msc.itso.ibm.com
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=sp2en0.msc.itso.ibm.com
export NIM_MASTER_PORT=1058
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/bos
export NIM_BOS_FORMAT=rte
export NIM_HOSTS=" 9.12.1.38:risc38.msc.itso.ibm.com  192.168.3.37:sp2en0.msc.it
so.ibm.com "
export NIM_MOUNTS=""
export ROUTES=" default:255.255.255.0:9.12.1.30 "
```

Figure 47. Output of Incorrect /etc/niminfo File

The information here is incorrect. It came from the system on which mksysb was created. Basically, what happened was this: the install progressed to this point because NIM was using the information contained in the SPOT. This was transferred early on in Phase 1 of rc.boot. By placing NIM into debug mode, we see this event:

```
+ /usr/lib/methods/showled 0x608
 showled LED{608}
+ tftp -g /SPOT/niminfo 192.168.4.137 /tftpboot/sp21n01.msc.itso.ibm.com.info im
age
Received 1292 Bytes in 0.1 Seconds
+ [ -s /SPOT/niminfo ]
+ . /SPOT/niminfo
+ export NIM_NAME=sp21n01
+ export NIM_HOSTNAME=sp21n01.msc.itso.ibm.com
+ export NIM_CONFIGURATION=standalone
+ export NIM_MASTER_HOSTNAME=sp21en0.msc.itso.ibm.com
+ export NIM_MASTER_PORT=1058
+ export RC_CONFIG=rc.bos_inst
+ export NIM_BOSINST_ENV=/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_bosinst_env
+ export NIM_BOSINST_RECOVER=/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_bosinst_e
nv -a hostname=sp21n01.msc.itso.ibm.com
+ export SPOT=sp21en0.msc.itso.ibm.com:/spdata/sys1/install/aix414/spot/spot_aix
414/usr
+ export NIM_BOSINST_DATA=/NIM_BOSINST_DATA
+ export NIM_CUSTOM=/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script -a location
=sp21en0.msc.itso.ibm.com:/export/nim/scripts/sp21n01.script
+ export NIM_BOS_IMAGE=/NIM_BOS_IMAGE
+ export NIM_BOS_FORMAT=mksysb
+ export NIM_HOSTS= 192.168.4.1:sp21n01.msc.itso.ibm.com  192.168.4.137:sp21en0.
msc.itso.ibm.com
+ export NIM_MOUNTS= sp21en0.msc.itso.ibm.com:/spdata/sys1/install/aix414/lppsou
rce:/SPOT/usr/sys/inst.images:dir  sp21en0.msc.itso.ibm.com:/spdata/sys1/install
/pssp/bosinst_data:/NIM_BOSINST_DATA:file  sp21en0.msc.itso.ibm.com:/spdata/sys1
/install/images/mksysb_test:/NIM_BOS_IMAGE:file
```

Figure 48. NIM Debug Output Showing the niminfo File

What happened was that the RAM environment was changed to disk, therefore using the correct niminfo file. However, the NIM install process detected that the NIM master file set was installed and moved the /etc/niminfo.prev file to /etc/niminfo. This is actually documented in APAR IX53428.

We checked the node; the niminfo.prev file existed. We also checked the mksysb image by running `restore -Tvf` against it, and the /etc/niminfo.prev file existed.

After using another “clean” mksysb, the install passed LED u03. This is, as mentioned, a situation where transferring pssp_script was not an option after identifying the problem because the cause was built into the install.

Note

A common source of LED u03 hangs are short and long naming conventions for the nodes. The reason for this is that when the transfer of <node>.install_info is called, <node> is determined by referencing the reliable hostname in the SDR, not the initial hostname. Therefore, a quick fix for the problem may be to change the file /tftpboot/<node>.install_info so that the <node> name matches the SDR reliable hostname definition.

Now let us look at the processor-specific customization in more detail.

A check is made for the processor type with the `bootinfo` command. If the type is Uniprocessor, the LED displayed is u51. Otherwise, it must be a Multiprocessor and LED u52 is displayed. If the processor type is MP, then some additional actions are taken to set up the node that would be required on a standalone SMP system.

Next, the script checks whether the currently loaded mksysb image contains the correct base operating system runtime kernel. If not, /spdata/sys1/install/<lppsource_name>/lppsource is mounted from the Control Workstation and an attempt is made to install. This newly installed kernel is then linked to /usr/lib/boot/unix and a `bosboot` command executed.

If an error occurs with `bosboot`, this is detected and LED u55 is displayed. It is worth noting here that `bosboot` fails if any of the actions immediately prior to its execution fail.

If `bosboot` completes correctly, then the lppsource directory is mounted and `installp` is invoked to install the latest versions of perfagent.server, bos.rte.tty, and devices.sio.sa.diag.

Note

bos.rte.tty and devices.sio.sa.diag are included in the SPOT used for the install of the system by default. You can check this with the command `nim -o lslpp -a lslpp_flags=″-al <fileset>″ spot_aix414`. The `installp` command called at this time will only install a later version of these file sets if one exists in the lppsource directory.

If a u55 LED is seen, or the customization hangs on u51 or u52, then the determination procedures are very much the same as for LED u03. We need to boot the node into “maintenance” and examine the log file in /var/adm/SPlogs/sysman to determine what caused the failure.

The final phase of customization takes place at this point.

This is where the relevant PSSP code specified for the node is actually loaded. Provided all the prerequisite software is installed, there should be problems here. If the prerequisite software is not installed and cannot be found in the mounted directory, then this should not be a fatal error. Instead, `installp` outputs an error to the /var/adm/SPlogs/sysman/<node>.config.log.<pid> log file on the node and continues to the next part of pssp_script. The only way then to determine if there was a problem installing is by either reviewing the logs or checking the PSSP software installed. Here is the sequence of events for the installation:

1. LED u80 is displayed.
   /spdata/sys1/install/pssplpp/<code_version> is mounted locally over the mount point /mnt, and ssp.clients is installed.
2. LED u81 is displayed.
   The `installp` command loads ssp.basic and calls the script /usr/lpp/ssp/install/bin/post_process to add the required daemons to /etc/inittab and start them.
3. LED u53 is displayed.
   The `installp` command loads ssp.ha.
4. LED u82 is displayed.
   The `installp` command loads ssp.sysctl and calls the script /usr/lpp/ssp/install/bin/post_sysctl to update /etc/services and SRC with entries for the sysctld daemon.
5. LED u83 is displayed.
   The `installp` command loads ssp.sysman and ssp.pman.
6. LED u84 is displayed.
   The `installp` command loads ssp.css and configures the switch into the ODM with cfgmgr and rc.switch.
7. LED u85 is displayed.
   `installp` loads ssp.jm.
8. Run script.cust if it exists, get the system partition name from the SDR, and change the response to boot to “disk.” Finally, clean up temporary files and exit from pssp_script.

Let us take a real-life example of a post-installation problem identification and correction.

In this phase of customization, as already mentioned, if an error occurs with `installp`, this is non fatal, meaning that the install will not halt at this point. So how can we tell whether the packages were all installed correctly?

Basically, on every node that was installed with PSSP 2.2, a total of nine packages should be installed, or eight if the system does not have a switch. The following list shows the packages that should be installed:

ssp.basic	SP System Support Package
ssp.clients	SP Authenticated Client Commands
ssp.css	SP Communication Subsystem Package (if this system has a switch)
ssp.ha	SP High Availability Services
ssp.jm	SP Job Manager Package (if this system has a switch)
ssp.perlpkg	SP PERL Distribution Package
ssp.pman	SP Problem Management
ssp.sysctl	SP Sysctl Package
ssp.sysman	Optional System Management programs

The simplest way of determining whether or not the PSSP installation part of customization succeeded is to list the PSSP file sets with lslpp. Below is the output from a node that was previously running with PSSP 2.1 and has been migrated:

```
-> lslpp -l ssp*
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  ssp.authent                2.1.0.2  APPLIED    SP Authentication Server
  ssp.basic                  2.2.0.0  COMMITTED  SP System Support Package
  ssp.clients                2.2.0.0  COMMITTED  SP Authenticated Client
                                                 Commands
  ssp.css                    2.2.0.0  COMMITTED  SP Communication Subsystem
                                                 Package
  ssp.docs                   2.1.0.7  APPLIED    SP man and info files
  ssp.gui                    2.1.0.6  APPLIED    SP System Monitor Graphical
                                                 User Interface
  ssp.jm                     2.2.0.0  COMMITTED  SP Job Manager Package
  ssp.perlpkg                2.2.0.0  COMMITTED  SP PERL Distribution Package
  ssp.public                 2.1.0.2  APPLIED    Public Code Compressed
                                                 Tarfiles
  ssp.sysctl                 2.2.0.0  COMMITTED  SP Sysctl Package
  ssp.sysman                 2.2.0.0  COMMITTED  Optional System Management
                                                 programs
  ssp.top                    2.1.0.2  APPLIED    SP Communication Subsystem
                                                 Topology Package

Path: /etc/objrepos
  ssp.authent                2.1.0.0  COMMITTED  SP Authentication Server
  ssp.basic                  2.2.0.0  COMMITTED  SP System Support Package
  ssp.clients                2.2.0.0  COMMITTED  SP Authenticated Client
                                                 Commands
  ssp.css                    2.2.0.0  COMMITTED  SP Communication Subsystem
                                                 Package
  ssp.jm                     2.2.0.0  COMMITTED  SP Job Manager Package
  ssp.sysctl                 2.2.0.0  COMMITTED  SP Sysctl Package
  ssp.sysman                 2.2.0.0  COMMITTED  Optional System Management
                                                 programs
  ssp.top                    2.1.0.0  COMMITTED  SP Communication Subsystem
                                                 Topology Package
```

Figure 49. A Full Software Listing for PSSP File Sets Installed

There are two points to note in the previous example. The first is that we still have PSSP 2.1 software packages listed as being committed. This is not a problem. It simply indicates that prior to migration the node had the full PSSP 2.1 package installed. The customization phase of PSSP 2.2 simply installs the required packages on top of any previous versions. It is up to you whether to deinstall the previous versions, but there will be no conflicts provided the required PSSP 2.2 file sets are loaded.

The second point to note is that we are missing two of the packages, ssp.ha and ssp.pman. This indicates that there was indeed a problem during the customization phase. Because of these missing file sets, we will see symptoms such as Host Responds displaying in red.

How can we find out what went wrong and go about correcting the problem?

The obvious first step is to once again review the log created by pssp_script in /var/adm/SPlogs/sysman. There we see the following entry for ssp.pman:

```
+ /usr/sbin/installp -a -X -d/mnt/pssp.installp ssp.pman
+-----------------------------------------------------------------------------+

                    Pre-installation Verification...
+-----------------------------------------------------------------------------+

installp:  Pre-installation verification may take several minutes.
Please wait...
Verifying selections...done
Verifying requisites...done
Results...

FAILURES
--------
  Filesets listed in this section failed pre-installation verification
  and will not be installed.

  Requisite Failures
  ------------------
  SELECTED FILESETS:  The following is a list of filesets that you asked to
  install.  They cannot be installed until all of their requisite filesets
  are also installed.  See subsequent lists for details of requisites.

    ssp.pman 2.2.0.0                         # SP Problem Management

  GROUP REQUISITES:  The dependencies of one or more of the selected filesets
  listed above are defined by a group requisite.  A group requisite must pass
  a specified number of requisite tests.  The following describe group
  requisite failures for filesets that you selected.  (See the "Requisite
  Failure Key" below for details of group member failures.)



  Requisite Failure Key:

  AVAILABLE REQUISITES:  The following filesets are requisites of one or
  more of the selected filesets listed above.  They are available on
  the installation media.  To install these requisites with the selected
  filesets, specify the option to automatically install requisite
  software (-g flag).

    ssp.ha 2.2.0.0                           # SP High Availability Services

  << End of Failure Section >>

FILESET STATISTICS
------------------
    1  Selected to be installed, of which:
        1  FAILED pre-installation verification
  ----
    0  Total to be installed


Pre-installation Failure/Warning Summary
----------------------------------------
Name                      Level           Pre-installation Failure/Warning
-------------------------------------------------------------------------------


ssp.pman                  2.2.0.0         Requisite failure
```

Figure 50. Typical Requisite Failure Output from installp

As we can see, ssp.pman requires that ssp.ha be installed on the system. However, ssp.ha also failed to be installed with a similar installp failure message. We know that the requisite for ssp.ha is perfagent.server. We also know that this should have been loaded during customization. This is another error from the same log file:

```
Finished processing all filesets.  (Total time:  10 secs).

+-----------------------------------------------------------------------------+

                         Post-installation Processing...
+-----------------------------------------------------------------------------+

+-----------------------------------------------------------------------------+

                                    Summaries:
+-----------------------------------------------------------------------------+

Pre-installation Failure/Warning Summary
----------------------------------------
Name                      Level           Pre-installation Failure/Warning
-------------------------------------------------------------------------------


perfagent.server                          Not found on the installation media
bos.rte.tty               4.1.4.0         Already installed
devices.sio.sa.diag       4.1.4.0         Already installed
devices.sio.sa.diag       4.1.4.1         Already installed
```

Figure 51. Failure to Install perfagent.server

Again the message is obvious: perfagent could not be found. This file set should be located in the same directory as the MP and UP and device software on the Control Workstation, that is, /spdata/sys1/install/aix414/lppsource. When checked, we see that the file set is missing, so we forgot to follow the basic checks before installing.

How do we go about correcting this?

This is a situation where we do not need to go through the entire installation or migration process again. The solution is to copy the image into the lppsource directory, then update the table of contents by running inutoc on the Control Workstation:

```
-> cd /spdata/sys1/install/aix414/lppsource
-> inutoc .
```

Now set the bootp_response to "customize" with spbootins and reboot the node. This is a normal boot, *not* a netboot. The customization script is called from /etc/rc.sp, not from NIM as in installation or migration. After the node has finished booting, check the lslpp output once again to confirm that the missing file sets have been loaded.

Chapter 3. Service Management Problems

The code of Kerberos, AMD or File Collections software did not change with PSSP 2.2, so the common problems of this software are the same as the ones we had using PSSP 2.1. Information about the Kerberos components and a description of some "radical" steps you can take to recreate the Kerberos database are found in *RS/6000 SP Problem Determination Guide*, SG24-4778.

But there are still some situations that have not been discussed yet, such as SP installations having a flat network. If you want to change to a domain name server, do you install the Control Workstation from scratch? And what about the nodes? Or what should you do if you do not want to use AMD any more? How can you synchronize the nodes running different time zones?

This chapter provides detailed solutions to these kinds of service management problems.

3.1 Kerberos Problems

Most known problems using Kerberos are related to the inability to perform an `rsh` to the server. There are many reasons for this problem.

Perhaps the server key has been updated, but not the existing tickets.

Perhaps there is a time difference between the systems (Kerberos can handle only a five-minute difference). If you changed the date, NTP will take a while to catch up. In the meantime, kerberized commands will fail.

Perhaps /tmp, where the ticket cache file resides, or /var, where the server keeps the authentication database, are 100% full. Apart from these possibilities, there are still many more reasons for Kerberos problems that we have not yet discussed.

Before you start Kerberos problem determination, follow this checklist:

1. Check the hostname resolution.
2. Check your ticket, using the `klist` command. You may have to destroy it and reissue it again (`kdestroy/kinit`).
3. Check the /.klogin file.
4. Check the PATH variable.
5. Check your file systems by using the `df -k` command.
6. Check the date on the server and client.
7. Check if the Kerberos daemons are running on the Control Workstation (`ps -ef|grep kerb`).
8. Check /etc/krb.realms on the client.
9. Check if you have to recreate /etc/krb-srvtab on the node.
10. Check /etc/krb-srvtab on the authentication server.

3.1.1 DNS or Flat Network?

From your own experience you know how important it is to set the proper hostnames and IP addresses for your Control Workstation and nodes, in the very first phase of the SP installation process.

Normally, when you install the system, you already know if you are going to use a domain name server or flat network. Usually, if there already is a domain name server, the SP system is supposed to use it. But what if you want your SP system with a flat network to use the domain name server after it has been configured and customized? Or, even better, how can you put a domain name server on an SP node?

There are two parts involved in solving this problem:

- SDR
- Kerberos

First you can create a path for changing these components, while changing from a flat network to using a domain name server.

Do the following steps on the nodes:

1. Change the hostname for the controlling Ethernet interface on the nodes with the following command:
```
mktcpip -h <new_node_hostname> -a <IP_address_of_the_node> \
 -m <subset mask> -i en? -n <name_server> -d <domain> \
 -g <gateway_IP_address> -t N/A
```
```
/usr/mktcpip -h'sp2n01 -a 192.168.3.1 -m'255.255.255.0' -i'en0' \       \
 -n'9.12.1.30' -d'msc.itso.ibm.com' -g'192.168.3.37' -t'bnc'
```
2. Take care of the NIM files.
   - The /etc/niminfo file should be updated and look similar to the following:

```
# warning - this file contains NIM configuration information
#       and should only be updated by NIM
export NIM_NAME=sp2n01
export NIM_HOSTNAME=sp2n01.msc.itso.ibm.com
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=sp2en0.msc.itso.ibm.com
export NIM_MASTER_PORT=1058
export RC_CONFIG=rc.bos_inst
export NIM_BOSINST_ENV="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_bosinst_env"
export NIM_BOSINST_RECOVER="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_bosinst_env
-a hostname=sp2n01.msc.itso.ibm.com"
export NIM_BOSINST_DATA=/NIM_BOSINST_DATA
export NIM_CUSTOM="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script
-a location=sp2en0.msc.itso.ibm.com:/export/nim/scripts/sp2n01.script"
export NIM_BOS_IMAGE=/NIM_BOS_IMAGE
export NIM_BOS_FORMAT=mksysb
export NIM_HOSTS=" 192.168.3.1:sp2n01.msc.itso.ibm.com 192.168.3.37:sp2en0.msc.itso.ibm.com "
export NIM_MOUNTS=" sp2en0.msc.itso.ibm.com:/spdata/sys1/install/aix42/lppsource:
/SPOT/usr/sys/inst.images:dir  sp2en0.msc.itso.ibm.com:/spdata/sys1/install/
images/bos.obj.ssp.42:/NIM_BOS_IMAGE:file "
```

You can update this file manually or by using NIM commands. If you are going to use the NIM commands, you have to do some workaround. We explain the necessary workaround after we describe the code of the pssp_script.

The /etc/niminfo file is created during node installation. It is a copy of the /tftpboot/<hostname>.info file that was created by setup_server after you changed the bootp_response of the node to “install.”

The pssp_script checks the content of the NIM_CUSTOM variable and, if the variable is set to “install,” the existence of /etc/niminfo is also checked. Then it is determined if the machine is standalone or not. If it is standalone, the contents of the /etc/niminfo file are used, as follows:

```
.....
if [[ $NIM_CUSTOM != "" ]]
then
        mode="install"
else
        mode="customize"
fi

if [[ $mode = "install" ]]
then
#Setup IP address of the NIM Master
if [[ -f /etc/niminfo ]]
then
        /bin/cp /etc/niminfo /tmp/niminfo 2>/dev/null
        chmod +x /tmp/niminfo
        . /tmp/niminfo 2>/dev/null
fi
# Does this system have a valid NIM client /etc/niminfo file
if [ "$NIM_CONFIGURATION" = "standalone" ]
then
        # Yes - use the /etc/niminfo file
        NIM_MASTER=/bin/echo $NIM_HOSTS | cut -f2 -d " " | cut -f1 -d:
        NIM_MASTER_HN=/bin/echo $NIM_HOSTS | cut -f2 -d " " | cut -f2 -d:
        NIM_MASTER_SHN=/bin/cat /etc/ssp/server_name | cut -f3 -d " "
        NIM_NAME=/bin/hostname -s
        CLIENTN=/bin/cat /etc/ssp/reliable_hostname | cut -f2 -d " "
else
        # No - must be a NIM master - use /etc/ssp/server_name
        # to determine NIM master address
        NIM_MASTER=/bin/cat /etc/ssp/server_name | cut -f1 -d " "
        NIM_MASTER_HN=/bin/cat /etc/ssp/server_name | cut -f2 -d " "
        NIM_MASTER_SHN=/bin/cat /etc/ssp/server_name | cut -f3 -d " "
        NIM_NAME=/bin/hostname -s
        CLIENTN=/bin/cat /etc/ssp/reliable_hostname | cut -f2 -d " "
fi
else
        host_number=/usr/lpp/ssp/install/bin/node_number
        if [[ $host_number -eq 0 ]]
        then
                echo "This is the control workstation"
                exit 1
        fi
.....
```

So how can you let NIM update the /etc/niminfo file without doing it manually and without installing the node from scratch?

You can use a workaround. Set the node to bootp_response=install and let setup_server create the configuration files. As soon as the /tftpboot/<hostname>.<domain_name>.info is created, you can copy it to the node as /etc/niminfo. Then set the bootp_response of the node to “disk” again. This step should be done *after* the SDR on the Control Workstation has been changed and *before* setting the bootp_response of the node(s) to “customize.”

- In case the node is a boot/install server, delete the whole NIM database from the machine by using the `delnimmast` wrapper:

```
# delnimmast -l <node_number>
```

3. Take care of the PSSP files.

   Make sure that the following files have the right information about the hostnames:

   - /etc/ssp/server_hostname

   ```
   192.168.3.37 sp2en0.msc.itso.ibm.com sp2en0
   ```

   - /etc/ssp/server_name

   ```
   192.168.3.40 sp2en3.msc.itso.ibm.com sp2en3
   ```

   - /etc/ssp/reliable_hostname

   ```
   192.168.3.1 sp2n01.msc.itso.ibm.com sp2n01
   ```

   - /etc/krb.conf, /etc/krb.realms, and /.klogin are recreated during node customization, so you do not have to change them manually.

4. Delete any Kerberos tickets by using the following command: `/usr/lpp/ssp/kerberos/bin/kdestroy`

Do the following steps on the Control Workstation:

1. Make a tar archive of your current /spdata file system.
2. Archive the SDR by using the `/usr/lpp/ssp/bin/SDRArchive` command.
3. Specify the new hostname for the Control Workstation:

```
/usr/sbin/mktcpip -h'sp2en0' -a'192.168.3.37' -m'255.255.255.0' -i 'en0' \
 -n'9.12.1.30' -d'msc.itso.ibm.com' -g'9.12.1.30' -t'bnc'
```

4. Change the /etc/krb.conf file. It should look similar to the following:

```
MSC.ITSO.IBM.COM
MSC.ITSO.IBM.COM sp2en0.msc.itso.ibm.com admin server
```

5. Change the /etc/krb.realms file:

```
sp2n01 MSC.ITSO.IBM.COM
sp2n02 MSC.ITSO.IBM.COM
sp2n03 MSC.ITSO.IBM.COM
sp2n04 MSC.ITSO.IBM.COM
sp2n05 MSC.ITSO.IBM.COM
sp2n06 MSC.ITSO.IBM.COM
sp2n07 MSC.ITSO.IBM.COM
sp2n08 MSC.ITSO.IBM.COM
sp2n09 MSC.ITSO.IBM.COM
sp2n10 MSC.ITSO.IBM.COM
sp2n11 MSC.ITSO.IBM.COM
sp2n12 MSC.ITSO.IBM.COM
sp2n13 MSC.ITSO.IBM.COM
sp2n14 MSC.ITSO.IBM.COM
sp2n15 MSC.ITSO.IBM.COM
sp2n16 MSC.ITSO.IBM.COM
```

6. Change the /.klogin file:

```
root.admin@MSC.ITSO.IBM.COM
rcmd.sp2n01@MSC.ITSO.IBM.COM
rcmd.sp2n02@MSC.ITSO.IBM.COM
rcmd.sp2n03@MSC.ITSO.IBM.COM
rcmd.sp2n04@MSC.ITSO.IBM.COM
rcmd.sp2n05@MSC.ITSO.IBM.COM
rcmd.sp2n06@MSC.ITSO.IBM.COM
rcmd.sp2n07@MSC.ITSO.IBM.COM
rcmd.sp2n08@MSC.ITSO.IBM.COM
rcmd.sp2n09@MSC.ITSO.IBM.COM
rcmd.sp2n10@MSC.ITSO.IBM.COM
rcmd.sp2n11@MSC.ITSO.IBM.COM
rcmd.sp2n12@MSC.ITSO.IBM.COM
rcmd.sp2n13@MSC.ITSO.IBM.COM
rcmd.sp2n14@MSC.ITSO.IBM.COM
rcmd.sp2n15@MSC.ITSO.IBM.COM
rcmd.sp2n16@MSC.ITSO.IBM.COM
```

7. Recreate /etc/krb-srvtab on the Control Workstation:

   In our configuration, we have one Token Ring interface and four partitions defined on the Control Workstation: sp2en0, sp2en1, sp2en2, and sp2en3, as follows:

```
# cd /tmp
# /usr/kerberos/etc/ext_srvtab -n sp2cw0 sp2en0 sp2en1 sp2en2 sp2en3
# ls -la
total 72
drwxr-xr-x   2 root     system       512 Feb 27 14:59 .
drwxrwxrwt   8 bin      bin         6656 Feb 27 15:04 ..
-rw-------   1 root     system        79 Feb 27 14:42 sp2cw0-new-srvtab
-rw-------   1 root     system        79 Feb 27 14:42 sp2en0-new-srvtab
-rw-------   1 root     system        79 Feb 27 14:49 sp2en1-new-srvtab
-rw-------   1 root     system        79 Feb 27 14:49 sp2en2-new-srvtab
-rw-------   1 root     system        79 Feb 27 14:49 sp2en3-new-srvtab
# cat sp2cw0-new-srvtab sp2en0-new-srvtab sp2en1-new-srvtab \
      sp2en2-new-srvtab sp2en3-new-srvtab > /etc/krb-srvtab
# chmod 400 /etc/krb-srvtab
```

8. Delete the NIM database by using the `delnimmast` wrapper:

```
# delnimmast -l 0
```

9. Change the reliable hostnames in the node class (for every partition):

```
# export SP_NAME=sp2en3
# SDRChangeAttrValues Node node_number==1 reliable_hostname=sp2n01.msc.itso.ibm.com
# SDRChangeAttrValues Node node_number==2 reliable_hostname=sp2n02.msc.itso.ibm.com
# SDRChangeAttrValues Node node_number==5 reliable_hostname=sp2n05.msc.itso.ibm.com
# SDRChangeAttrValues Node node_number==6 reliable_hostname=sp2n06.msc.itso.ibm.com
# SDRGetObjects Node node_number initial_hostname reliable_hostname
node_number  initial_hostname reliable_hostname
          1 sp2n01       sp2n01.msc.itso.ibm.com
          2 sp2n02       sp2n02.msc.itso.ibm.com
          5 sp2n05       sp2n05.msc.itso.ibm.com
          6 sp2n06       sp2n06.msc.itso.ibm.com
```

10. Change the hostnames used in the Switch_partition class to include the full domain name (for each partition):

```
# SDRChangeAttrValues Switch_partition switch_partition_number==1 \
 primary_name=sp2n01.msc.itso.ibm.com oncoming_primary_name=sp2n01.msc.itso.ibm.com \
 primary_backup_name=sp2n06.msc.itso.ibm.com oncoming_primary_backup_name=sp2n06.msc.itso.ibm.com

# SDRGetObjects Switch_partition switch_partition_number primary_name \
   oncoming_primary_name primary_backup_name oncoming_primary_backup_name
switch_partition_number primary_name oncoming_primary_name primary_backup_name oncoming_primary_backup_name
  1  sp2n01.msc.itso.ibm.com sp2n01.msc.itso.ibm.com sp2n06.msc.itso.ibm.com sp2n06.msc.itso.ibm.com
```

11. Set the bootp_response of all nodes to “install.” This command lets the setup_server create all NIM resources and configuration files, as follows:

```
# spbootins 1 1 16 -r install
```

12. Set the bootp_response of all nodes to “customize.” This command invokes the setup_server routine also, but with the bootp_response set up to “customize,” the `allnimres` wrapper will not do any allocation.

```
# spbootins 1 1 16 -r customize -s no
# setup_server
```

Verify that the *.config_info and *.install_info files in /tftpboot directory are using the reliable hostnames as part of the file names, as follows:

```
# cd /tftpboot
# ls *info
sp2n01.msc.itso.ibm.com.config_info   sp2n08.msc.itso.ibm.com.config_info
sp2n01.msc.itso.ibm.com.install_info  sp2n08.msc.itso.ibm.com.install_info
sp2n02.msc.itso.ibm.com.config_info   sp2n11.msc.itso.ibm.com.config_info
sp2n02.msc.itso.ibm.com.install_info  sp2n11.msc.itso.ibm.com.install_info
sp2n03.msc.itso.ibm.com.config_info   sp2n12.msc.itso.ibm.com.config_info
sp2n03.msc.itso.ibm.com.install_info  sp2n12.msc.itso.ibm.com.install_info
sp2n04.msc.itso.ibm.com.config_info   sp2n13.msc.itso.ibm.com.config_info
sp2n04.msc.itso.ibm.com.install_info  sp2n13.msc.itso.ibm.com.install_info
sp2n05.msc.itso.ibm.com.config_info   sp2n14.msc.itso.ibm.com.config_info
sp2n05.msc.itso.ibm.com.install_info  sp2n14.msc.itso.ibm.com.install_info
sp2n06.msc.itso.ibm.com.config_info   sp2n15.msc.itso.ibm.com.config_info
sp2n06.msc.itso.ibm.com.install_info  sp2n15.msc.itso.ibm.com.install_info
sp2n07.msc.itso.ibm.com.config_info   sp2n16.msc.itso.ibm.com.config_info
sp2n07.msc.itso.ibm.com.install_info  sp2n16.msc.itso.ibm.com.install_info
```

13. Make sure that the lines in /tftpboot/script.cust for copying /etc/resolv.conf from the Control Workstation to the node(s) and getting the tickets are uncommented, as follows:

```
.....
# Define the file that will store the ticket
export KRBTKFILE=/tmp/tkt_rcmd$$

# Get the kerberos ticket
/usr/lpp/ssp/rcmd/bin/rcmdtgt
#
# If you are using /etc/resolv.conf to resolve hostnames and addresses
# then use this to copy the /etc/resolv.conf file from the boot/install server
#
# Perform kerberos authenticated rcp
/usr/lpp/ssp/rcmd/bin/rcp $SERVER:/etc/resolv.conf /etc/resolv.conf
.....
# Remove the ticket file
/bin/rm $KRBTKFILE
.....
```

The /etc/resolv.conf file should look as follows:

```
search  msc.itso.ibm.com itso.ibm.com
domain  msc.itso.ibm.com
nameserver      9.12.1.30
```

14. Refresh the Kerberos daemons:

```
# ps -ef|grep kerb
    root  4056 44502   2 10:07:33  pts/6  0:00 grep kerb
    root  5132     1   0   Feb 25      -  0:00 /usr/lpp/ssp/kerberos/etc/kadmind -n
    root 16722     1   0   Feb 25      -  0:00 /usr/lpp/ssp/kerberos/etc/kerberos
# kill 5132 16722
```

Now you can start rebooting the nodes. Customize the boot/install servers *first*, then the remaining nodes. Each boot/install server must recreate the NIM database.

After performing this procedure, you should verify that it was successful on both the Control Workstation and on the nodes by doing the following:

1. Check the hostname resolution.
2. Recheck the following files on the nodes:

   ```
   # cat /etc/ssp/reliable_hostname
   192.168.3.1 sp2n01.msc.itso.ibm.com sp2n01
   # cat /etc/ssp/server_name
   192.168.3.40 sp2en3.msc.itso.ibm.com sp2en3
   # cat /etc/ssp/server_hostname
   192.168.3.37 sp2en0.msc.itso.ibm.com sp2en0
   ```

3. Check the following files on the nodes and Control Workstation:
   - /etc/krb.conf
   - /etc/krb.realms
   - /etc/krv-srvtab
   - /.klogin

Note: Do not forget to update your domain name server with the entries for your SP system.

3.2 File Collection Problems

File collections must be transferred to the nodes and kept up-to-date. The supper Perl script (/var/sysman/supper) uses the Software Update Protocol (SUP) to manage the SP file collections and transfer them across the system. Table 4 shows the summary of the default file collections on the SP system. Examine the notation of these groups closely, especially the definiton of *available* and *resident* file collections.

Table 4. File Collections on an SP System

Control Workstation		Boot/Install Servers		SP Nodes	
resident	available	resident	available	resident	available
sup.admin	node.root	sup.admin		sup.admin	
user.admin		user.admin		user.admin	
power_system		node.root		node.root	
		power_system			

3.2.1 Resident or Available?

What does it mean that the file collection is *resident*? It means that this group of files or directories is installed on the system. In other words, these files are a part of the configuration of this system. They can be found in the file system by using their proper paths. For example, the file collection user.admin includes (in the default setting) the following files:

```
# /var/sysman/sup/user.admin/list
symlinkall
upgrade ./etc/passwd
upgrade ./etc/group
upgrade ./etc/security/group
upgrade ./etc/security/passwd
upgrade ./etc/amd/amd-maps/amd.*
execute /etc/amd/refresh_amd (./etc/amd/amd-maps/amd.u)
```

These files are a part of the system configuration of all machines and are updated on the nodes if you make any changes to them on the Control Workstation. If you add an SP user to the system, /etc/passwd and its security-related files on the Control Workstation are automatically changed. These changes must be updated to the nodes. Therefore, the user.admin file collection is resident everywhere, both on the supper_servers and on the nodes, as follows:

```
# /var/sysman/supper status
 Collection          Resident  Access Point         Filesystem          Size
==============================================================================
 node.root              Yes    /                        -                 -
 power_system           Yes    /share/power/system      -                 -
 sup.admin              Yes    /var/sysman              -                 -
 user.admin             Yes    /                        -                 -
```

File collections like sup.admin and user.admin are resident on the Control Workstation, boot/install servers, and SP nodes.

Now what about file collections that are *available*? And what are *primary* and *secondary* file collections?

Primary	A file collection (resident) that is used by the servers and served to the nodes (for example, sup.admin).
Secondary	A file collection (available) that is served by the servers but is not used by the servers (for example, node.root).

A primary file collection can be a stand-alone collection (such as sup.admin) or can contain a secondary collection (such as power_system containing node.root). The secondary file collection, in other words, available, allows you to have a group of files that are not applied to the system but can be distributed to the nodes, and will be resident on the nodes.

The power_system and node.root file collections are good examples of the difference between resident and available.

The node.root file collection contains files that are node-specific. It is resident on the nodes, but available on the Control Workstation. It is not applied to the system of the Control Workstation because this collection might get into a conflict with the root environment files on the Control Workstation.

Let us take the node.root collection as an example for a case study.

If you want to have one /.profile on the Control Workstation and a different /.profile on the nodes, you can use the power_system collection, which is used for files that are system-dependent. It contains only files defined in the node.root file collection, which is resident on the nodes but available on the Control Workstation.

Examine the meaning of the last sentence. The power_system collection has a pointer to the directory that contains the node.root files. This directory is somewhere in the system, but not in the directory of the file collection, as follows:

```
# cat /var/sysman/sup/power_system/list
symlinkall
upgrade ./share/power/system/3.2
```

Now examine the node.root directory and see what the content of the prefix file is for this file collection.

The prefix file contains the name of the directory for the file references and the starting point for the `supper scan <file collection>` process. Also determine what activities should be done:

```
# cat /var/sysman/sup/node.root/prefix
/share/power/system/3.2
# cat /var/sysman/sup/node.root/list
upgrade ./.profile
```

So how can you use two different /.profile files for the Control Workstation and the nodes?   .

1. Copy the required .profile to the /share/power/system/3.2 directory on the Control Workstation.
2. Make sure that the power_system has the following configuration:

```
# cat /var/sysman/sup/power_system/prefix
/
# cat /var/sysman/sup/power_system/list
symlinkall
upgrade ./share/power/system/3.2
```

3. Change the node.root file collection files to the following:

```
# cat /var/sysman/sup/node.root/prefix
/share/power/system/3.2
# cat /var/sysman/sup/node.root/list
upgrade ./.profile
```

4. Make sure that the power_system file collection is resident on the master server (which, by default, is the Control Workstation), as follows:

```
# /var/sysman/supper status
 Collection      Resident  Access Point         Filesystem  Size
===================================================================
...
power_system       Yes    /share/power/system      -         -
...
```

   If it is not resident, install it by issuing:
   `supper install power_server`.

5. Make sure that the node.root file collection is resident on the nodes, as follows:

```
# dsh -a /var/sysman/supper status
sp2n01:  Collection     Resident  Access Point      Filesystem      Size
============================================================================
.....
sp2n01:  node.root        Yes    /                    -             -
.....
sp2n02:  node.root        Yes    /                    -             -
.....
sp2n05:  node.root        Yes    /                    -             -
.....
sp2n06:  node.root        Yes    /                    -             -
.....
```

   If it is not installed, install it by issuing:
   `dsh -a /var/sysman/supper install node.root`.

6. Update the power_server file collection on the nodes (or boot/install server first, if any) by issuing: `dsh -a /var/sysman/supper update power_system`.

After performing these steps, log on to the nodes. You will see that the root environment set on the nodes differs from the root environment on the Control Workstation.

3.2.2 The Supper Server Story

From which server does the node you are working on get updates to the file collections? To determine this, you normally issue the supper where command and the output looks similar to the following:

```
# supper where
supper:  Collection node.root would be updated from server sp2en3.
supper:  Collection power_system would be updated from server sp2en3.
supper:  Collection sup.admin would be updated from server sp2en3.
supper:  Collection user.admin would be updated from server sp2en3.
```

Examine this action more closely. The supper Perl script invokes a function called findserver(), as follows:

```
# Find a server for collections and print out the results.  If called with no
# arguments, find a server for all collections.
#
sub where {
        local($c) = @_;
        foreach $p (&query($IS_COLL, split(' ', $c))) {
                $server = &findserver($p);
.....
```

This findserver() function uses the Korn shell script called /etc/mastername, as follows:

```
sub findserver {
        local($c) = @_;
        chop($servername = /etc/mastername 2>/dev/null);
        $servername;
}
```

The /etc/mastername file obtains the name of the server machine from the /etc/ssp/server_name file:

```
.....

SERV_FN=/etc/ssp/server_name
if [[ -f $SERV_FN ]]
then
  cat $SERV_FN | cut -d' ' -f2
fi
```

And last but not least, the content of the /etc/ssp/server_name file is as follows:

```
# cat /etc/ssp/server_name
192.168.3.40 sp2en3.msc.itso.ibm.com sp2en3
```

This is an example of an installation using four partitions. The Control Workstation is the server for the supper activities, and supper refers to the name of the partition (in this case, sp2en3) to which the node you are working on belongs. In our configuration, we do not have any boot/install servers for PSSP

2.2, but if we did, they would be valid servers from which supper could pull its updates for the nodes that have been installed from them.

Now what is the relationship between all these servers that should support supper activities? What happens if you have a Control Workstation and a boot/install server that are configured as a file collection server? What if the file collection server for my node is a boot/install server and not the Control Workstation? Where and how has it been defined?

During system installation, you use the `spsitenv` command and define filecoll_config=true by default. That means that setup_server configures the boot/install server (it could be the Control Workstation or another node) as the master server for all default file collections. You can install the nodes from the Control Workstation or any boot/install server you have defined. If there are any boot/install servers in your SP system, they will also be file collection servers.

After the installation and configuration of the SP is complete, you will notice a sort of "hierarchy" for the delivery of file collections:

- Nodes that have been installed from the Control Workstation will get their file collections updated from the Control Workstation.
- Nodes installed from the boot/install server will get file collection updates from this boot/install server.

For problem determination on the file collection server, it is very important to know exactly which actions are being performed, and in which order.

There is only one location for the master files (files that get updated) on an SP system and that is the Control Workstation (if you did not change the default setting).

Let us assume a case in which an SP user has been added to the SP system. The hierarchy of doing updates is as follows:

- The file collections update is performed on the Control Workstation.
- The `supper update` command is issued from the boot/install server, requesting updates for the user.admin file collection from the Control Workstation.
- The `supper update` command is issued from the nodes to the boot/install server, requesting updates for the files of the user.admin file collection.

So if you add a new SP user to the system, the /etc/passwd file and other security-related files are updated in the root.admin file collection. After this action, the new SP user can log on to the Control Workstation but *cannot* log on to the nodes because they still have the old version of the user.admin file collection. Therefore, after the update has been done on the Control Workstation, the nodes should update their files as well.

You have two possible ways to perform these updates:

- Do nothing and let cron do the work for you. (By default, cron is set up to pull the updates every hour.)
- If you do not want to wait for cron, you can do the updates manually by using `/var/sysman/supper update user.admin` for each node. To make it easier, use the `rsh`, `rexec` or `dsh` commands.

Important

If you do not have a boot/install server, you can use the command
`# dsh -a /var/sysman/supper update user.admin` without any problem.

But if you do have a boot/install server on your SP system, what happens if you issue the preceding `dsh` command?

For example, in our configuration, assume node 1 is a boot/install server from which node 2, node 3 and node 4 have been installed. If you issue the `dsh` command as shown above, nodes 2, 3 and 4 will pull their updates from node 1 as their boot/install server, and node 1 will pull its updates from the Control Workstation.

Because these actions are performed in parallel, after the processes complete, nodes 2, 3 and 4 will have the old version of the user.admin file collection and node 1 will have the new one.

So in this case, if nothing additional is done, the nodes will operate for one more hour with the old version of /etc/passwd and related files and the new SP user will be able to log on to the Control Workstation and node 1 only.

To avoid this side effect, issue the `dsh` command using the `-f1` option, which will cause the update processes to be performed sequentially: node 1 will be updated first, then node 2, and so on. This command is as follows:

```
# dsh -a -f1 /var/sysman/supper update user.admin
```

3.2.3 Changing the boot/install Server to Be a Master Server

What about the possibility of changing the function of a supper server from getting the updates from the master server (by default, the Control Workstation) to making the updates of the file collections locally on the boot/install server?

There are some default hierarchy rules about the delivery of the updates for the file collections to the nodes. The normal way is:

Control Workstation → boot/install server → the nodes.

Therefore, if you want to change the server for one or more file collections, you can do it by changing the hierarchy of the server.

As mentioned before, it is very important for problem determination purposes to know the exact configuration of the system, whether the defaults have been changed, and what the consequences of these changes are for the mechanism you are using (in our case, file collections).

Examine the following scenario, in which we want to change the server for the power_system file collection.

We have two frames (frame 1 and frame 2) and two boot/install servers, node 1 for the nodes in frame 1 and node 17 for the nodes in frame 2. We want node 17 to be a master server for the delivery of the power_system file collection to the nodes in frame 2, because the root environment of the nodes in frame 2 should differ from the environment of the nodes in frame 1.

This can be done by eliminating the logical path for the power_system file collection from the Control Workstation to node 17. After performing this step, if any change was made to the node.root file collection (which will, of course, be affected by this change because of its association with the power_server file collection), the systems that will get the updates from the Control Workstation are node 1 (as boot/install server for the nodes in frame 1) and the nodes in frame 1. Node 17 and other nodes in frame 2 will not get any update. Now we can change node.root on node 17 directly and let supper update the nodes in frame 2 (this time from node 17).

Let us make it easier to follow using the procedure to change the master server for the power_system file collection, as follows:

1. Break the path for the power_server file collection between the Control Workstation and node 17, as follows:

   Log on to node 17 and issue:
   `/var/sysman/supper offline power_system`.

2. Modify the /share/power/system/3.2/.profile file on node 17.

3. Issue `dsh -a /var/sysman/supper update power_server`.

   In this case, you can use it without the `-f1` option.

4. Log on to the Control Workstation and remove the entry from the crontabs file, updating the power_server file collection and producing error messages about failing every hour, since node 17 cannot get the updates.

   In this case, we used the standard (default) file collections that are created by the system automatically during installation. However, you can build your own file collections, resident or available, in order to keep your systems up-to-date by updating them from your own file collection pool.

   What about the possibility of choosing the supper server from which your node can pull the updates of the file collections?

   There is no supported possibility to choose “your” file collection server. The definition of the server is done while installing the nodes (from the Control Workstation or other boot/install server), and supper uses the information stored in the /etc/ssp/server_name file to get the address of the server, from which it can pull the updates of the file collections.

3.2.4 Scan, Simply the Best

Using the scan files is an optional feature, but it is highly recommended that you use them to increase the throughput of an update command when a large number of file collections are going to be updated.

If a scan file is present, there is no need to run install and update processes to do a directory search, and the install and update commands use the scan file.

If there is no scan file, the update goes through the same path as scan did to complete the requested actions. It also checks the refuse file in /var/sysman/sup at each location and bypasses any files it contains. The result of processing the refuse file creates a refuse file in the file collection directory, indicating which files were bypassed.

The scan file is an inventory of all files and directories belonging to this particular file collection. It is located in the /var/sysman/sup/file_collection directory. The file collection must be resident if you want to scan it.

Let us examine a scan file of the user.admin file set:

```
# cat /var/sysman/sup/user.admin/scan
V2
100754 857429912 857429912 etc/amd/amd-maps/amd.u
X/etc/amd/refresh_amd
100644 857429911 857429911 etc/group
100644 857430011 857430011 etc/passwd
100640 857261868 854148460 etc/security/group
100640 857430099 857430099 etc/security/passwd
240755 857340088 857340088 etc/my_admin_dir
```

The first field contains two parts:

1. The first two digits show you the type of the file (file or directory). The file digits are 10 and the directory digits are 24.
2. The next four digits are UNIX permissions for this file in octal form.

The next two fields are the time stamps of the date of the installation and the date of the last update.

Scan files require maintenance, and whenever modifications are done to the file collection, its scan file must be updated. If the scan files are not up-to-date, you run into the problem of updating the collection from an old master file.

Important

To be able to scan the file collection on a master server, that file collection must be resident on the system. If it is not, you must install it (use `supper install <file collection>`). During installation, the .resident file is updated with the new file collection.

Let us review what happens if the node requests an update of a file collection from the supper master server.

1. First, the client takes the when file located locally in the /var/sysman/sup/file collection, which has a list of files with the time stamp of their last update.
2. The server checks if there is a scan file for this file collection and compares the time stamps of the when file with the time stamps of the scan file.
3. If the time stamps of the when file are earlier than the ones in the scan file, any updated files listed in the scan file are sent to the node.
4. If there is no scan file, the time stamps of the when file are compared with the time stamps of the files in their locations.
5. As soon as the node receives the requested updates of the files, the last file, the when file, and the scan file are updated as well.

During updates, the lock and supperlock files prevent more than one update access by SUP. Supper also compares the update time in the when file with the time stamp of the file itself, to be sure that the file will not be overwritten by an old version. If it is an old copy, the update is not done.

3.2.5 Common Problems

Most problems related to the Software Update Protocol (SUP) occur because clients are not able to connect the server (Control Workstation). Therefore, you mostly have to deal with problems such as:

- Network problems.
- The server machine lacks the supman ID.

3.2.5.1 Mismatch between the SUP Versions

You may receive the following error message:

```
# supper install power_system
supper:  0018-258  Update of collection power_system failed.
supper:  0018-278  Server not available.
```

One of the problems that still occurs even though we now run PSSP 2.2, is the mismatch between the versions of the SUP binaries of the Control Workstation and the nodes.

When an old version on an SUP client tries to communicate with a new version on the SUP file server (the Control Workstation), the error above occurs. The solution is to copy the new version of the SUP, supfilesrv, and supscan binaries from the /usr/lpp/ssp/filec/etc directory to the /var/sysman/etc/sup directory on the nodes.

3.2.5.2 SUP Server Is Busy

You may receive the following error message:

```
0018-244 warning:
Server sp2en0.msc.itso.ibm.com is busy. Will retry in 30 seconds.
```

This error may occur if the SUP server is busy serving other clients. By default, it can only handle eight requests at a time. The retry period is two minutes and when it expires, the requested action is stopped. Also make sure that the supfilesrv daemon is running on the server, and check whether the supman user ID is valid on both the server system and the client system.

3.2.5.3 Troubles while Scanning the File Collection

You may receive the following error message:

```
0018-257 Collection user.admin cannot be scanned
or
0018-288 Collection is not resident
```

While scanning the user.admin file collection, supper has problems with finding information in the master files. Even if the file collection is defined as resident in the /var/sysman/sup/.resident file, if some information in the master file is missing, there is obviously a problem. Try to find out what information is missing:

```
# supper rlog
SUP 7.24 (4.3 BSD) for file /tmp/.sf16158 at Mar  5 23:10:01
SUP Upgrade of sup.admin at Wed Mar  5 23:10:01 1997
SUP Fileserver 7.12 (4.3 BSD) 49446 on sp2en3.msc.itso.ibm.com
SUP Locked collection sup.admin for exclusive access
SUP Requesting changes since Wed Mar  5 22:10:02 1997
SUP Upgrade of sup.admin completed at Wed Mar  5 23:10:01 1997
SUP Scan for sup.admin starting at Wed Mar  5 23:10:02 1997
SUP Scan for sup.admin completed at Wed Mar  5 23:10:02 1997
SUP 7.24 (4.3 BSD) for file /tmp/.sf16158 at Mar  5 23:10:03
SUP Upgrade of user.admin at Wed Mar  5 23:10:03 1997
SUP Fileserver 7.12 (4.3 BSD) 49468 on sp2en3.msc.itso.ibm.com
SUP Locked collection user.admin for exclusive access
SUP Requesting changes since Wed Mar  5 22:10:03 1997
SUP Upgrade of user.admin completed at Wed Mar  5 23:10:03 1997
SUP Scan for user.admin starting at Wed Mar  5 23:10:03 1997
SUP Scan for user.admin completed at Wed Mar  5 23:10:03 1997
```

For some reason supper thinks that user.admin is not resident. Check to see if there is an entry for user.admin in the /var/sysman/sup/.resident file. Make sure the entry is there and issue: `supper install user.admin`.

3.2.5.4 Supper Cannot Locate the Server

You may receive the following error message:

```
0018-278 Server not available
or
0018-269 Connection failed
```

This error message tells you that the attempt to connect the SUP server failed. It happens mostly when the supfilesrv daemon is not running on the server. The daemon may die because of problems with the network. Make sure that the network is stable. Try to use the debug tools in such a situation, as follows:

```
# supper
supper> log
                                Supper Log
       Host: sp2n01                             Date: 3/6/97  0:10:01
                                            CHANGES TO SYSTEM
Collection          Server           Updated    Removed    Errors
-------------------------------------------------------------------------
sup.admin           sp2en3.msc.its      0          0         0
user.admin          sp2en3.msc.its      0          0         0
node.root           sp2en3.msc.its      0          0         0
-------------------------------------------------------------------------
Totals                                  0          0         0
Supper Finished  3/6/97  0:10:05
```

You can read from the log that the supfilesrv daemon terminated at 0:10:05 on 3/6/97. Try to also use the debug option for supper. It may give you some useful information about the cause of your problem. The options you may use with supper are as follows:

```
status                 Show status of all collections.
when                   Show the last update time of all resident collections.
where                  Show current servers for collections.
log                    Show summary of last/current supper session.
rlog                   Show raw output of last/current supper session.
install <collection>   Install a collection.
update  <collection>   Update a collection.
remove  <collection>   Remove a collection.
scan    <collection>   Run a scan for a collection.
reset   <collection>   Set the last update time of a collection to The Epoch.
online  <collection>   Enable updates of a collection (this is the default).
offline <collection>   Disable updates of a collection.
serve                  List all collections we are able to serve.
files   <collection>   Show all files associated with a resident collection.
diskinfo               Show available disk space and active volume group.
activate <vg>          Set the active volume group.
debug    on|off        Turn on/off debgging messages.
verbose  on|off        Turn on/off SUP output messages.
quit                   Exit the program.
!<command>             Shell escape.
```

There can be several reasons why supper does not work, and most of them have to do with problems with the network or the file collection configuration. Let us go through a simple checklist before you start problem determination on your particular problem; you may find the roots of your conflict in this list.

Checklist for file collection problems:

1. Supper uses the Ethernet network only. Check the TCP/IP configuration. Ping the server from the client. Check the hostname resolution.
2. Use the supper status command (or check the .resident file) to find out whether the file collection is resident or not.

```
 Collection          Resident  Access Point          Filesystem          Size
===========================================================================
 node.root              Yes    /                        -                   -
 power_system            -     /share/power/system      -                   -
 sup.admin              Yes    /var/sysman              -                   -
 user.admin             Yes    /                        -                   -
===========================================================================
```

   If the update of the file collection failed and this file collection is not resident on the node, install it by issuing the supper install <file collection> command.

3. Use supper where on the node to see which machine is your supper server, as follows:

```
# supper where
supper:  Collection node.root would be updated from server sp2en3.
supper:  Collection power_system would be updated from server sp2en3.
supper:  Collection sup.admin would be updated from server sp2en3.
supper:  Collection user.admin would be updated from server sp2en3.
```

4. Check if the file collection server daemon is running on the Control Workstation:

```
# lssrc -s supfilesrv
Subsystem         Group            PID     Status
 supfilesrv                        15018   active
```

5. Does the server have the supman ID? Is this ID non-loggable? It should have an “*” in the /etc/passwd file, as follows:

```
# grep supman /etc/passwd
supman:*:102:7::/:/bin/ksh
```

6. Check the /etc/services file on the server machine. Is the supfilesrv daemon defined and does it have a correct port?

```
# grep supfilesrv /etc/services
supfilesrv      8431/tcp
```

7. Check the logfiles located in the /var/sysman/logs directory.

8. Check the logfiles located in the /vad/adm/SPlogs/filec directory.

3.3 AMD Problems

The majority of AMD problems are related to the following three components:

- Network
- Configuration (for example, unexported directories)
- Maps updates

Before you start problem determination, check first if those three components are working.

Checklist for the common problems that occur in connection with AMD:

1. Check if the AMD daemon is running. If not, use the /etc/amd/amd_start script to activate it.
2. Make sure that the required directories are exported. Check if they are in the list of the directories that should be exported and run the `exportfs -a` command.
3. Check the AMD map files. If you have problems with logging on to the system, check the /etc/amd/amd-maps/amd.u file for the existence of the entry for the user ID you want to use. An entry should look like the following:

   ```
   netinst      type:=link;fs:=/home
   .....
   efri    host==sp2en0;type:=link;fs:=/home/sp2en0 \
                 host!=sp2en0;type:=nfs;rhost:=sp2en0;rfs:=/home/sp2en0
   ```

   If there is no entry for the user ID you would like to use, add it to this file. Make sure that the updates are distributed after the change with the command:
   `dsh -w <nodelist> supper update user.admin sup.admin power_system`
4. Check whether the network connection is still working. You will not be able to log on to the system if you cannot access it. Ping the target host to check the connection. If the connection is down, use spmon to get the front panel for the node and open a TTY. As soon as you log on to the system, bring the interface up.
5. Get the information about the AMD mounts by issuing the `/etc/amd/amq` command. If the output of `amq` looks as follows:
   `amq: localhost: RPC: Program not registered`
   your problem could be:

   - The AMD daemon is not running.
   - The portmap daemon is not running.
   - The AMD daemon is waiting for a response from the NFS server that is not responding.

   Make sure that the portmap daemon is running and that your NFS server is responding. If the portmap daemon is inoperative, start it with the `startsrc -s portmap` command.

   If you have an NFS server problem, check the amd.log file located in the /var/adm/SPlogs/amd directory. It contains information about the AMD daemon activities and may look similar to the following:

```
.....
/etc/amd/amd_start: Starting Amd on Tue Feb 25 14:58:24 EST 1997
/etc/amd/amd_start: Started Amd on Tue Feb 25 14:58:24 EST 1997
Feb 25 14:58:25 sp2en0 amd[14640]/warn:  NIS domain name is not set. NIS ignored.
Feb 25 14:58:25 sp2en0 amd[14640]/info:  /etc/amd/amd-maps/amd.u mounted fstype toplvl on /u
Mar  3 18:22:06 sp2en0 amd[14640]/info:  amq says flush cache
Mar  3 18:22:17 sp2en0 amd[14640]/map:   Trying mount of ./home/sp2en0/efri on /u/efri fstype link
Mar  3 18:22:17 sp2en0 amd[14640]/info:  ./home/sp2en0/efri mounted fstype link on ./home/sp2en0/efri
Mar  3 18:27:17 sp2en0 amd[14640]/info:  ./home/sp2en0/efri unmounted fstype link from ./home/sp2en0/efri
```

You may find some information there about the non-responding NFS server. If the user files are mounted and the NFS server is not responding, the amd daemon may be hanging on the NFS mount request. Check to see if there are users working with AMD directories by using the amq command. Make sure that nobody is using any AMD directory, then unmount the AMD mounted directories (for example, amq -u /u/sp2en0/efri). Stop AMD (by issuing kill -15 <amd_pid>), solve your NFS problems, and start AMD again (with /etc/amd/amd_start).

6. If you have user access problems, do the following:

   - Verify that the login and rlogin options for your user are set to true. Otherwise, the login control facility may lock your user and you will be unable to login, and the parallel jobs will not run.
   - Check the user path or .rhosts on the node. If you have problems executing rsh to the node, check the user path to see if the user is supposed to be a Kerberos principal. Make sure that the kerberized version of rsh, located in /usr/lpp/ssp/rcmd/bin, is the first in the user′s PATH variable. If the user does not use the Kerberos authentication control, check the .rhosts file on the node. (If there is no .rhosts file, check the /etc/hosts.equiv file on the node.)

7. If you have problems executing an SP user administrative command, you may get an error message similar to the following:

   ```
   0027-153 The user administration function is already in use.
   ```

   In this case, the most probable cause is that another user administrative command is running and there is a lock in effect for the command to let it finish. If no other administrative command is running, check the /usr/lpp/ssp/config/admin directory for the existence of a .userlock file. If there is one, remove it and try to execute your command again.

3.3.1 Changing from an AMD to a non-AMD Configuration

The definition for using AMD, NTP, or File Collection is done when using the spsitenv command during installation. After you define which options you are going to use, the system generates the appropriate environment for their use, management, and controlling facilities. If you do not change any option, the definition of your system will look similar to the following:

```
# splstdata -e
             List Site Environment Database Information

attribute                 value
-------------------------------------------------------------------------------
control_workstation       sp2en0
cw_ipaddrs                9.12.1.37:192.168.3.40:192.168.3.39:192.168.3.38:192.168.3.37:
install_image             bos.obj.ssp.42
remove_image              false
primary_node              1
```

```
ntp_config              consensus
ntp_server              ""
ntp_version             3
amd_config              true
print_config            false
usermgmt_config         true
passwd_file             /etc/passwd
passwd_file_loc         sp2en0
homedir_server          sp2en0
homedir_path            /home/sp2en0
filecoll_config         true
supman_uid              102
supfilesrv_port         8431
spacct_enable           false
spacct_actnode_thresh   80
spacct_excluse_enable   false
acct_master             0
cw_has_usr_clients      false
code_version            PSSP-2.2
authent_server          ssp
layout_dir              /spdata/sys1/syspar_configs/1nsb0isb/config.4_4_4_4/layout.1
backup_cw               ""
ipaddrs_bucw            ""
active_cw               ""
cw_lppsource_name       aix42
```

As you see, AMD and user management set to true are the defaults.

If, after the SP system is installed and running, you wish to change the options to a non-AMD configuration, perform the following procedure:

1. Let all users log off from the SP system.
2. Save the user files located in the /home/hostname/user_id on the system you defined as a home_dir server.
3. Stop AMD daemon processes on the Control Workstation and the nodes. (For the nodes, you can use dsh.)
4. Unmount the /u directory on the Control Workstation and the nodes.
5. Before you change the spsitenv options, it is useful to define now whether or not you are still going to use the file collection mechanism. You can switch off both AMD and File Collection and update your files or file groups with the pcp command (or perhaps by using your own cron jobs).

   (If you are going to use the file collection mechanism, make sure that you have excluded the information about user.admin that is used by File Collection.)

   Proceed as follows on the Control Workstation and all nodes:

   - Comment out the line with user.admin in the /var/sysman/file.collections file.
   - Remove the link for user.admin from the /var/sysman/sup/lists directory.
   - Edit the /var/sysman/sup/.resident file and remove the entry for user.admin.
6. Change the spsitenv options:

```
                     Site Environment Information
Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                 [Entry Fields]

.....

  AMD Configuration                        false

.....

  User Administration Interface            false

.....

  File Collection Management               false

.....
```

7. Reboot the Control Workstation and the nodes. You can also run /etc/rc.sp instead of rebooting the systems. /etc/rc.sp is invoked from /etc/inittab.

   At line # 72 of /etc/rc.sp, the /usr/lpp/ssp/install/bin/services_config script is invoked if the system is the Control Workstation, as follows:

```
.....
     # Pick up any changes to the site environment - amd/ntp/filec...
                $SSP_INSTALL/services_config

.....
```

   The /usr/lpp/ssp/install/bin/services_config script checks the SP object in the SDR to see which services should run on the system. Then it calls the appropriate service config scripts. Possible services to be run are NTP, Print, User Management, AMD, and File Collection services.

   One of the service config scripts is admin_config, which prepares the environment for the selected option, as follows:

```
.....

if [[ $USERMGMT_CONFIG != "TRUE" ]]
  then
 # de-config user management configuration if it was previous configured.

.....
```

   In our case, the environment for the single point of user administration is deleted and we can use another form of user management for the SP system.

8. Make sure that all user home directories and other directories previously served by AMD are distributed and made available in another way.

3.4 NTP Problems

The default way to synchronize the time-of-day clocks on the Control Workstation and the nodes is to run Network Time Protocol (NTP) locally on the SP to generate a consensus time. If you do not wish to use the default option, you have the possibility of using your site′s existing NTP time server (timemaster), or an Internet time service.

In addition, you also have the possibility of not using NTP on the SP at all. In this case, you will have to use some other method to synchronize the system clocks.

If you need more information about how NTP synchronizes the time across the network, refer to the man pages for xntpd and xntpdc, or to RFC 1305.

3.4.1 Synchronizing the Time Setting

It is imperative to synchronize the time-of-day on your SP system. If this is not done, you may have several problems with the PSSP partition-sensitive daemons, and especially with Kerberos. Kerberos uses the time stamps for decoding the authorization of provided services.

If you follow the Open Systems Interconnect (OSI) Reference Model, the data is passed down the stack when being sent to the network, and up the stack when being received from the network. Each layer in the stack adds control information (header) in front of the transmitted data to ensure proper delivery. This header information includes the time stamp. During attempts to establish a connection between the communication adapters of the SP systems, the time stamps are compared and if they do not match, the connection between the adapters cannot succeed. On the SP system (apart from Kerberos problems), this will result in a lack of host response and confuse the other partition-sensitive daemons running on the system. (Also refer to Chapter 1, “RS/6000 SP Infrastructure” on page 1.)

The common configuration of an SP system provides all the nodes with the same time zone setting. But what about a more complex configuration with nodes running different time zones; how do you synchronize them?

If the clock is not synchronized on your SP system, the first application to get confused is Kerberos because it can only handle five-minute differences. It is not important how many time zones you run on your nodes, because the time stamp is the same for all of them. What is important is the *time setting*. If the time set on the node differs from the time of its time server by more than five minutes, Kerberos gets into trouble decoding the authorization, because the time stamp is part of it.

```
# dsh -a date
sp2n01: Sat Mar  1 19:07:36 CST 1997
sp2n02: Sat Mar  1 15:07:36 HST 1997
sp2n05: kshd: 0041-005 Kerberos rsh or rcp failed: 2504-031 Kerberos error:
 can′t decode authenticator
sp2n05: /usr/lpp/ssp/rcmd/bin/rsh: 0041-004 Kerberos rcmd failed:rcmd protocol failure.
sp2n05: trying normal rsh (/usr/bin/rsh)
sp2n05: rshd: 0826-813 Permission is denied.
sp2n06: Sun Mar  2 03:06:52 USAST 1997
dsh:  5025-509 sp2n05 rsh had exit code 1
```

In this partition, shown in the preceding figure, there are four nodes: sp2n01, sp2n02, sp2n05, and sp2n06. After the time has been changed on sp2n05, Kerberos can no longer access this node.

To solve this problem, set the correct time on your node. Make sure which machine is your time server. The file /etc/ntp.conf includes the information you need:

```
.....
# The following entries were added by ntp_config during sp
# configuration.  They are (generally) the ip addresses of
# the time servers for the system and any peers that may be
# present as well... The configuration will differ based on
# the environment variable NTP_TYPE (set in the CMI).
#
server 192.168.3.37
server 192.168.3.10
server 192.168.3.9
```

The three machines shown are the Control Workstation and the other boot/install servers configured on our SP system.

If you have problems calculating the proper time that should be set on the node, just set the same time zone and time as your time server (normally your Control Workstation) on your node, and then change the time zone to the one you are going to use on this node.

Instead of changing the time or recalculating it, you can kill the xntpd daemon on the node and restart it using the /etc/rc.ntp script.

If you want to synchronize all the nodes on the SP system, you should kill the xntpd daemons on all the nodes and restart them using the /etc/rc.ntp script. This action synchronizes the time on all nodes with the Control Workstation, which is the default SP time server.

Attention

Kerberos problems seldom occur alone. If setting the proper time is not enough to solve your problem, do the following:

- Recreate the /etc/krb-srvtab file on the node by executing `/usr/lpp/ssp/kerberos/etc/ext_srvtab -n node_name` and copying it over to the node as /etc/krb-srvtab.
- Destroy your tickets on the node with the `kdestroy` command.
- Obtain the tickets for your node with the `kinit` command.

3.4.2 Changing from Consensus to Timemaster

Most SP installations use NTP to generate a consensus time, which means that no time server outside the SP is used for synchronizing the system clocks.

But what about a situation where you have to use an already existing time server and must change the time server mode on the SP from consensus to, for example, timemaster?

In this case, examine the SDR information about the NTP server:

```
# SDRGetObjects SP ntp_config ntp_server ntp_version
ntp_config   ntp_server   ntp_version
consensus    ""           3
```

Change the NTP type from consensus to timemaster, as follows:

1. Check the hostname resolution of the new time server.
2. Check the accessibility of the new time server.
3. Stop the xntpd daemon on the Control Workstation and the nodes.
4. Change the site environment information:

```
                 Site Environment Information
Type or select values in entry fields.
Press Enter AFTER making all desired changes.

.....

  NTP Installation                         timemaster
  NTP Server Hostname(s)                   risc71.msc.itso.ibm.com
  NTP Version                              3

.....
```

5. Change the information about the time server in /etc/ntp.conf on the Control Workstation and on the nodes.
6. Start the xntpd daemon on the Control Workstation and on the nodes.

Chapter 4. Problem Management

To let the operator know the indications of a problem, or the problem itself, the originator should log such error information somewhere. On AIX, there are two error logging facilities: the AIX error log and the BSD syslog.

PSSP 2.2 provides a new tool for problem and logging management, which allows you to handle problem notification and recovery actions. This chapter focuses on this new tool, providing several examples of how to use it to set up an infrastructure for problem management.

4.1 Problem Management Subsystem

The Problem Management subsystem (pman) is an infrastructure for recognizing and acting on problem events within the SP system. If a system value that is being monitored periodically changes, or exceeds a predefined value, pman can execute a predefined program or issue an SNMP trap. It can also add an entry to the AIX error log or BSD syslog. This chapter describes how to use this subsystem to avoid problems, by notifying the operator and or by taking corrective actions.

4.1.1 General Concept

The Problem Management subsystem is part of the new High Availability infrastructure on PSSP 2.2 This new infrastructure is made up of the following daemons:

- haemd (Event Management daemon)

  This is the central component of the Event Management subsystem. The Event Manager daemon generates an event if some monitoring values are applied to the predefined condition.

- pmand (Problem Management daemon)

  This is a client program of haemd. If the event that subscribed to it occurs, pmand performs some action previously defined.

- pmanrmd (Problem Management Resource Monitor daemon)

  This daemon updates pman resource variables. If the values differ from the predefined value, pmand is notified of this situation by Event Manager.

Figure 52 on page 188 shows the relationship between the daemons.

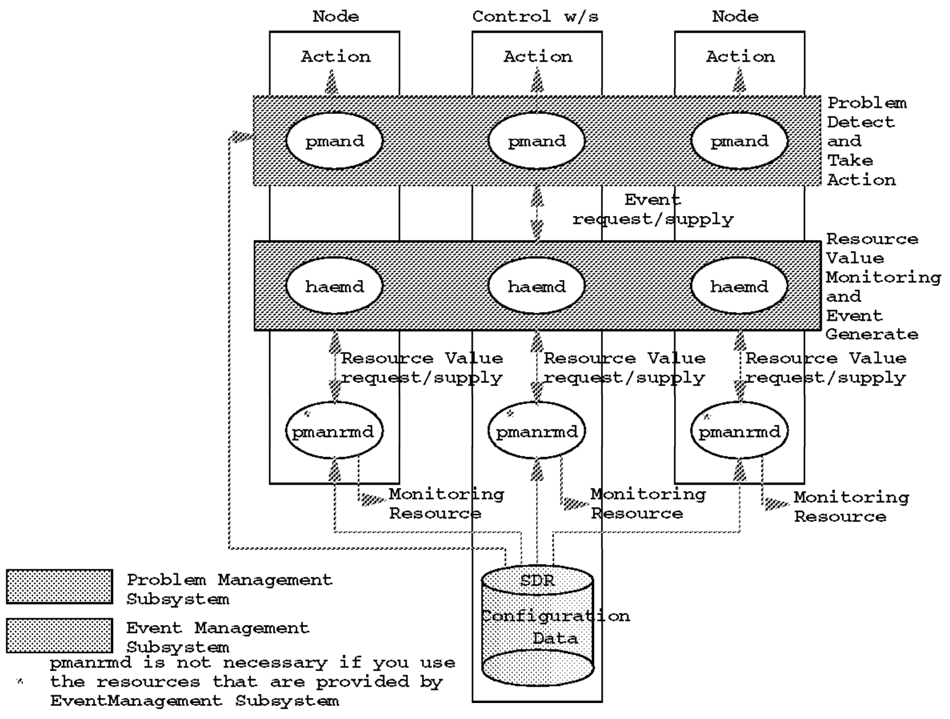

Node
Control w/s
Node
Action
Action
Action
pmand
pmand
pmand
Problem Detect and Take Action
Event request/supply
haemd
haemd
haemd
Resource Value Monitoring and Event Generate
Resource Value request/supply
Resource Value request/supply
Resource Value request/supply
pmanrmd
pmanrmd
pmanrmd
Monitoring Resource
Monitoring Resource
Monitoring Resource
SDR
Configuration Data
Problem Management Subsystem
Event Management Subsystem
pmanrmd is not necessary if you use the resources that are provided by EventManagement Subsystem

Figure 52. Relation between the pman Daemons

All definitions for pman are saved in the SDR via the pmandef command. pmand on each node accesses the SDR and so knows the configuration definitions. If these are changed, this is reflected to all pmand daemons.

The Event Management subsystem provides a resource variable that is monitored periodically by it. If you use this variable, you do not need to use pmanrmd, which is used to monitor special values that are not provided by the Event Manager.

To find out more about how to use the pman subsystem, refer to *RS/6000 SP Monitoring: Keeping It Alive*, SG24-4873.

4.1.2 The pmandefaults Script

This script sets default events to be monitored. If the installation of nodes and the Control Workstation were done successfully, that is, if the daemons described in 4.1.1, "General Concept" on page 187 are all running, then the following steps are needed in order to use the problem management subsystem.

1. You must become a root.admin user.
2. Add the root.admin Kerberos principal to the /etc/sysctl.pman.acl file, as follows:

```
#acl#

# These are the kerberos principals for the users that can configure
# Problem Management on this node.  They must be of the form as indicated
# in the commented out records below.  The pound sign (#) is the comment
# character, and the underscore (_) is part of the "_PRINCIPAL" keyword,
# so do not delete the underscore.

_PRINCIPAL root.admin@MSC.ITSO.IBM.COM
#_PRINCIPAL joeuser@PPD.POK.IBM.COM
```

The pmandef command, which defines events and resulting actions, is based on the Sysctl facility. Any user of the PMAN subsystem should be listed in this file.

The root.admin principal is commented out by default. To validate it, remove the first # character of the corresponding line and correct the realm name. The realm name is shown by the klist command. (The default setting in sysctl.pman.acl is PPD.POK.IBM.COM.)

3. Copy the /etc/sysctl.pman.acl file to all the nodes.

   The pcp command may be used for this. The following command copies the file to all the nodes in the same partition:

   ```
   # pcp -a /etc/sysctl.pman.acl
   ```

4. Execute pmandefaults.

   This script takes no arguments. Enter the following from the command line:

   ```
   # /usr/lpp/ssp/install/bin/pmandefaults
   ```

The following events are defined by this command for all nodes (including the Control Workstation) in the same partition. When one of these events occurs, the root user on the Control Workstation receives a notification (this is the default action).

- The /tmp file system is more than 95% full.
- The /var file system is more than 95% full.
- An error log record of type PERM has been written to the AIX error log.
- The inetd daemon has terminated.
- The sysctld daemon has terminated.
- The fault_service daemon has terminated.
- The sdrd has terminated.

  This applies to the Control Workstation only.
- The hrd daemon has terminated.

  This applies to the Control Workstation only.
- An error log record of label EPOW_SUS has been written to the AIX error log.

  These errors should only occur on SMP nodes.

This script is a suggested starting point for configuration of the pman subsystem.

The following section describes how to use pman to monitor critical subsystems on the SP.

4.1.3 Subscribe Events for Problems with Daemons

On the SP, communication with each node (including the Control Workstation) is accomplished by the use of daemons. If the system administrator can be made aware of a daemon failure early, the availability of SP is improved.

The inetd daemon on each node is monitored by executing the pmandefaults script:

```
#
#  Monitor inetd daemon on each node in the partition
#
pmandef -s Monitor_inetd_daemon  \
    -e 'IBM.PSSP.Prog.xpcount:NodeNum=*;ProgName=inetd;UserName=root:X@0==0' \
    -r 'X@0>0'  \
    -c /usr/lpp/ssp/bin/notify_event  \
    -C "/usr/lpp/ssp/bin/notify_event -r"  \
    -n 0 -U root
[[ $? -ne 0 ]] && print -u2 "Problem with Monitor_inetd_daemon" && exit 1
```

When inetd becomes inoperative on one of the nodes or on the Control Workstation (X@0==0), /usr/lpp/ssp/bin/notify_event is executed (specified by the -c option). After that, when inetd becomes operative (X@0>0), /usr/lpp/ssp/bin/notify_event -r is executed (specified by the -C option).

The following mail notification is sent to root at the Control Workstation by the notify_event command:

```
# mail
Mail [5.2 UCB] [AIX 4.1]  Type ? for help.
"/var/spool/mail/root": 1 message 1 new
>U  1 root              Tue Feb 11 15:22  25/744  "Monitor_inetd_daemon"
? 1
Message  1:
From root Tue Feb 11 15:15:05 1997
Date: Tue, 11 Feb 1997 15:15:04 -0500 (EST)
From: root
To: root
Subject: Monitor_inetd_daemon

Monitored situation exists on node 12.
Event Monitor_inetd_daemon reported at Tue Feb 11 15:15:03 1997.

Event Definition
----------------
Resource:   IBM.PSSP.Prog.xpcount
Instance:   UserName=root;ProgName=inetd;NodeNum=12
Condition:  X@0==0

Resource Value
--------------
Type:       sbs
Field0:     CurPIDCount=0
Field1:     PrevPIDCount=1
Field2:     CurPIDList=

?
```

The `notify_event` command gets the information displayed in this mail notification example from different environment variables. Following is a list of these variables and their values in the example:

PMAN_LOCATION
Monitored situation exists on Node 12.

PMAN_HANDLE
Event Monitor_inetd_daemon reported at Tue Feb 11 15:15:03 1997.

PMAN_TIME
Event Monitor_inetd_daemon reported at Tue Feb 11 15:15:03 1997.

PMAN_RVNAME
Resource: IBM.PSSP.Prog.xpcount

PMAN_IVECTOR
Instance: UserName=root;ProgName=inetd;NodeNum=12

PMAN_PRED
Condition: X@0==0

PMAN_RVTYPE
Type: sbs

PMAN_RVCOUNT
If PMAN_RVTYPE is sbs, this variable is the number of PMAN_RVFIELD# (# means a numeral).

PMAN_RVFIELD#
Field0: CurPIDCount=0
Field1: PrevPIDCount=1
Field2: CurPIDList=

You can monitor any process by replacing the ProgName and UserName.

In general, the following daemons of PSSP 2.2 should be running on both the Control Workstation and the nodes (listed in alphabetical order):

amd Automount daemon. If AMD is configured by `smitty site_env_dialog`, this daemon should be running.

haemd Event Manager daemon. On the Control Workstation, there should be one instance of this daemon running for each partition.

hagsd Group Services daemon. On the Control Workstation, there should be one instance of this daemon running for each partition.

hagsglsmd A Group Services daemon. On the Control Workstation, there should be one instance of this daemon running for each partition.

hatsd Topology Services daemon. On the Control Workstation, there should be one instance of this daemon running for each partition.

pmand Problem Management daemon. On the Control Workstation, there should be one instance of this daemon running for each partition.

sp_configd

This daemon creates and sends SNMP traps when selected types of errors are recorded.

sysctld Sysctl daemon.

xntpd Network time protocol daemon. If NTP is configured by `smitty site_env_dialog`, this daemon should be running.

The following daemon should be running only on nodes:

fault_service_Worm_RTG or fault_service_Worm_RTG_SP

These are fault-handling daemons for the Switch. The first one should be running on nodes using the High Performance Switch or HiPS. The second daemon should be running on nodes using the new SP Switch or SPS.

The following PSSP 2.2 daemons should be running only on the Control Workstation (in alphabetical order):

hardmon	Hardware monitor daemon.
hbd	Heartbeat daemon. There should be one instance of this daemon running for each PSSP 2.1 partition.
hrd	Host responds daemon. There should be one instance of this daemon running for each partition.
kadmin	Kerberos database administration daemon.
kerberos	Kerberos authentication ticket-granting service daemon.
sdrd	SDR daemon. The should be one instance of this daemon running for each partition.
supfilesrv	This daemon provides services for file collection. If File Collection Management is configured with the Control Workstation as file collection server, this daemon should be running.
splogd	This daemon logs SP hardware state changes and calls corresponding user exits.

4.1.4 Subscribe Events for Problems with File Systems

On the nodes, the file system that is likely to become full is /tmp. It is open to all users on the system. If it becomes full, programs that use it as working space cannot continue executing, or even start.

Another file system that is likely to become full is /var. This file system keeps much data for each user, for example data concerning mail, editors, printers, cron, and so on. It is also used as the log directory for PSSP 2.2. If it suddenly becomes full, a daemon may experience problems and print many error messages to the file. To make problem determination easier, this directory should always retain some free space.

To avoid these file system problems, the system administrator can apply resource usage limits for each user. Of course, when hardware is ordered, necessary resource estimates should be done.

You can use pmandefaults to monitor /tmp and /var, as follows:

```
#
#  Watch /tmp space on each node in the partition
#
pmandef -s Tmp_filling_on_nodes  \
    -e 'IBM.PSSP.aixos.FS.%totused:NodeNum=*;VG=rootvg;LV=hd3:X>95'  \
    -r 'X<60'   \
    -c /usr/lpp/ssp/bin/notify_event  \
    -C "/usr/lpp/ssp/bin/notify_event -r"  \
    -n 0 -U root
#
#  Watch /var space on each node in the partition
#
pmandef -s Var_filling_on_nodes  \
    -e 'IBM.PSSP.aixos.FS.%totused:NodeNum=*;VG=rootvg;LV=hd9var:X>95'  \
    -r 'X<70'  \
    -c /usr/lpp/ssp/bin/notify_event  \
    -C "/usr/lpp/ssp/bin/notify_event -r"  \
    -n 0 -U root
```

A mail notification is sent to root on the Control Workstation when the usage of /var or /tmp exceeds 95%.

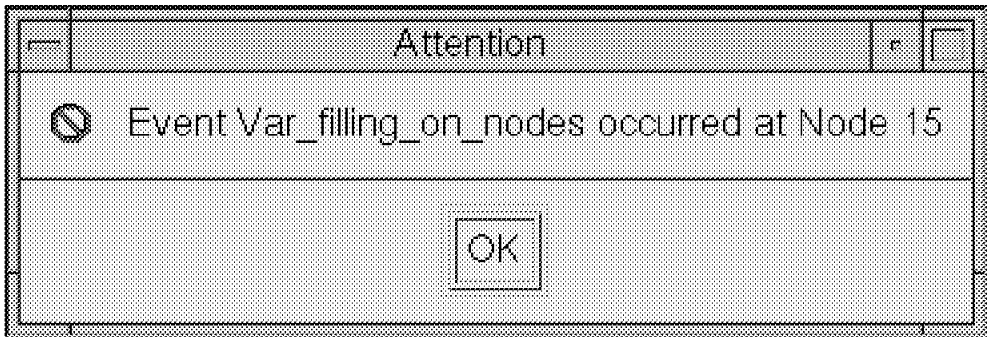
For example, if X server is always running on the Control Workstation, a GUI that alerts the user to the problem can be shown instead of a mail notification. The /usr/dt/bin/dterror.ds script is available if X11.Dt.rte is installed. The following example shows a simple alert script:

```
#!/bin/ksh
#  salert.sh
#  Very Simple Alert Program for pman
#
export DISPLAY=:0
function bellring
{
 while :
 do
     sleep 1
     echo ^G > $DEST
 done
}

DEST=/usr/bin/tty
bellring &
/usr/dt/bin/dterror.ds "Event $PMAN_HANDLE occurred at Node $PMAN_LOCATION" \
                       "Attention" \
                       "OK"
```

This script must be used with the `aixterm` command, as follows:

```
aixterm -i -e /usr/local/bin/salert.sh
```

As an alert, the following window is displayed by this command and the keyboard bell is rung:



Event Var_filling_on_nodes occurred at Node 15

Figure 53. Sample Alert Window

The `echo ^G` command rings the keyboard bell. However, in order to execute this command properly, it must be associated to a TTY. Also, the `dterror.ds` command cannot handle child process properly, so this script must be executed by `aixterm -e -i`, which means that `aixterm` is displayed iconified.

Note

To input ^G from the keyboard, push Ctrl+v, then Ctrl+g.

To replace the command for the Var_filling_on_nodes event, use the following procedure:

1. Enter `pmandef -u Var_filling_on_nodes`
2. Enter `pmandef -s Var_filling_on_nodes \`

```
             -e
 'IBM.PSSP.aixos.FS.%totused:NodeNum=*;VG=rootvg;LV=hd9var:X>95' \
             -r 'X<70'
             -c "export DISPLAY=:0;/usr/lpp/ssp/bin/aixterm -i -e \
                 /usr/local/bin/salert.sh" \
             -C "/usr/lpp/ssp/bin/notify_event -r"  \
             -n 0 -U root
```

The value for the *display* environment variable is `0`. Because `-n 0` means the program specified by `-c, -C` is executed on Node 0, that is, the Control Workstation

Note

To display the window, the screen lock and screen saver functions must be off. And if the X authentication mechanism (access control to the X server per user basis) is used, the `xhost <hostname>` command should be executed in advance.

4.1.5 Subscribe Events for Problems with Memory and Paging Space

The pmandefaults script does not subscribe events about memory or paging space. However, Event Manager provides the following resource variables for memory and paging space (the redbook *RS/6000 SP Monitoring: Keeping It Alive*, SG24-4873 provides further information):

IBM.PSSP.aixos.Mem.Kmem.calls
: Number of requests for a kernel memory buffer

IBM.PSSP.aixos.Mem.Kmem.failures
: Number of failed requests for a kernel memory buffer

IBM.PSSP.aixos.Mem.Kmem.inuse
: Count of kernel memory buffers in use

IBM.PSSP.aixos.Mem.Kmem.memuse
: Current memory use

IBM.PSSP.aixos.Mem.Real.%free
: % of free memory

IBM.PSSP.aixos.Mem.Real.%pinned
: % of pinned memory

IBM.PSSP.aixos.Mem.Real.numfrb
: Number of pages on free list

IBM.PSSP.aixos.Mem.Real.size
: Size of physical memory (4K pages)

IBM.PSSP.aixos.Mem.Virt.pagein
: 4K pages read by VMM

IBM.PSSP.aixos.Mem.Virt.pageout
: 4K pages written by VMM

IBM.PSSP.aixos.Mem.Virt.pagexct
: Total page faults

IBM.PSSP.aixos.Mem.Virt.pgspgin
: 4K pages read from paging space by VMM

IBM.PSSP.aixos.Mem.Virt.pgspgout
: 4K pages written to paging space by VMM

IBM.PSSP.aixos.PagSp.%totalfree
: Total free disk paging space (percent)

IBM.PSSP.aixos.PagSp.%totalused
: Total used disk paging space (percent)

IBM.PSSP.aixos.PagSp.totalfree
: Total free disk paging space (4K pages)

IBM.PSSP.aixos.PagSp.totalsize
: Total active paging space

To avoid performance degradation caused by insufficient memory, resource limitations for each user, or limits on the number of users logged on, should be applied to the system in advance. In general, if there is not enough memory, many pages are consumed in paging space, which causes many page I/Os from paging space to memory, which degrades system performance. Therefore, monitoring paging space is a good way to know the current status of system memory.

For example, to generate an event when free disk paging space becomes low, the following `pmandef` command is used:

```
/usr/lpp/ssp/bin/pmandef -s Paging_space_free \
    -e 'IBM.PSSP.aixos.PagSp.%totalused:NodeNum=1-16:X>70'  \
    -r 'X<50'  \
    -c "/usr/lpp/ssp/bin/notify_event" \
    -C "/usr/lpp/ssp/bin/notify_event -r"  \
    -n 0 -U root
```

When usage of paging space is too high, Event Manager or pman itself do not work correctly. Therefore, this subscription defines that when the usage of paging space exceeds 70%, events are generated.

Because the Control Workstation is only used for system administration purposes, this events do not need to be generated on it. `NodeNum=1-16` means that the event should be generated on all nodes except the Control Workstation (`NodeNum=0` is the Control Workstation).

Instead of using the default command `notify_event`, you may run specific commands to get snapshots of the system. The following command uses the `svmon -P` command and mails it to root.

```
/usr/lpp/ssp/bin/pmandef -s Paging_space_free \
    -e 'IBM.PSSP.aixos.PagSp.%totalused:NodeNum=1-16:X>70'  \
    -r 'X<50'  \
    -c "/usr/bin/svmon -P | mail -s All_Processes root@sp2en0" \
    -C "/usr/lpp/ssp/bin/notify_event -r"  \
    -h local -U root
```

`-h local` means that the commands that are specified by `-c` or `-C` are executed on the node that generated the event.

Attention

Note that `svmon` cannot be used unless the perfagent.tools file set is installed.

4.1.6 Subscribe Events for Problems with the CPU

When system performance is degraded by a CPU bottleneck, it should be monitored as CPU idle time is 0% for a while. If the CPU usage information of each process is obtained at this time, it is useful for problem determination or performance tuning.

The `tprof` command is suitable for that purpose. It can record CPU usage information of all processes on the system for a specified period. The following is the `pmandef` command for this purpose:

```
/usr/lpp/ssp/bin/pmandef -s CPU_idle_time \
    -e 'IBM.PSSP.aixos.CPU.glidle:NodeNum=1-16:X<10'  \
    -r 'X>20'  \
    -c "/usr/local/bin/starttprof" \
    -C "/usr/local/bin/stoptprof"  \
    -U root -h local
```

When CPU idle time becomes less than 10%, the script starttprof, which has the `tprof` command in it, is executed on the node that generated the event. Then, when CPU idle time is more than 20%, stoptprof runs to stop profiling. Following is an example of starttprof:

```
#!/bin/ksh
#
# starttprof
#
STARTTIME=date +"%m%d%H%M%S"
WORKDIR=/usr/local/data/$STARTTIME

mkdir $WORKDIR
cd  $WORKDIR

tprof -x /usr/local/bin/dummy.sh 2>&1 > tprof.out
```

tprof produces its report to the current directory. And tprof -x means it monitors the CPU usage until the specified command, in this case dummy.sh, is completed.

Here is an example of dummy.sh:

```
#!/bin/ksh
#
# dummy.sh
#
echo $$ > /tmp/dummy.pid
suspend
```

We see that it suspends itself after recording its process ID. The stoptprof script kills this program to see the /tmp/dummy.pid file. Then tprof produces its report to the /usr/local/data/<MMDDhhmmss> directory:

```
#!/bin/ksh
#
# stoptprof
#
kill cat /tmp/dummy.pid
```

Of course, all these programs must be put in the /usr/local/bin directory on all the nodes in advance. And the /usr/local/data directory must also be created in advance.

Note

The tprof command cannot be used unless the perfagent.tools file set is installed.

4.1.7 Subscribe Events for Problems with the Network and Switch

Event Manager provides the following resource variables. They correspond to hostResponds and switchResponds, which are monitored by spmon -g.

- IBM.PSSP.Response.Host.state

  This indicates whether the node has connectivity over a network. A value of 1 indicates connectivity; 0 indicates no connectivity. If the node is running a

PSSP release that does not include Event Manager, only the SP Ethernet is checked.

- IBM.PSSP.Response.Switch.state

  This indicates whether the switch adapter has been initialized on the switch. A value of 1 indicates it has; 0 indicates it has not.

The following two pmandef commands define these events:

```
# /usr/lpp/ssp/bin/pmandef -s Host_Responds_Status \
    -e 'IBM.PSSP.Response.Host.state:NodeNum=1-16:X==0'\
    -r 'X>0'  \
    -c "export DISPLAY=:0;/usr/lpp/ssp/bin/aixterm -i -e \
        /usr/local/bin/salert.sh" \
    -C "/usr/lpp/ssp/bin/notify_event -r"  \
    -n 0 -U root

# /usr/lpp/ssp/bin/pmandef -s Switch_Responds_Status \
    -e 'IBM.PSSP.Response.Switch.state:NodeNum=1-16:X==0' \
    -r 'X>0'  \
    -c "export DISPLAY=:0;/usr/lpp/ssp/bin/aixterm -i -e \
        /usr/local/bin/salert.sh" \
    -C "/usr/lpp/ssp/bin/notify_event -r"  \
    -n 0 -U root
```

Event Manager provides resource variables related to Switch and LAN adapters. The following are resource variables for the Switch:

IBM.PSSP.CSS.bcast_rx_ok
Number of broadcast packets received

IBM.PSSP.CSS.bcast_tx_ok
Number of broadcast packets sent

IBM.PSSP.CSS.ibadpackets
Number of bad packets received from adapter

IBM.PSSP.CSS.ibytes_dlt
Total number of octets received (delta)

IBM.PSSP.CSS.ibytes_lsw
Total number of octets received (lsw bits 0-30)

IBM.PSSP.CSS.ibytes_msw
Total number of octets received (msw bits 31-61)

IBM.PSSP.CSS.ierrors
Input errors on interface

IBM.PSSP.CSS.ipackets_dlt
Packets received on interface (delta)

IBM.PSSP.CSS.ipackets_drop
Number of packets not passed up

IBM.PSSP.CSS.ipackets_lsw
Packets received on interface (lsw bits 0-30)

IBM.PSSP.CSS.ipackets_msw
Packets received on interface (msw bits 31-61)

IBM.PSSP.CSS.nobufs
No buffers available

IBM.PSSP.CSS.obytes_dlt
Total number of octets sent (delta)

IBM.PSSP.CSS.obytes_lsw
Total number of octets sent (lsw bits 0-30)

IBM.PSSP.CSS.obytes_msw
Total number of octets sent (msw bits 31-61)

IBM.PSSP.CSS.oerrors
Output errors on interface

IBM.PSSP.CSS.opackets_dlt
Packets sent on interface (delta)

IBM.PSSP.CSS.opackets_drop
Number of packets not transmitted

IBM.PSSP.CSS.opackets_lsw
Packets sent on interface (lsw bits 0-30)

IBM.PSSP.CSS.opackets_msw
Packets sent on interface (msw bits 31-61)

IBM.PSSP.CSS.recvintr_dlt
Number of receive interrupts (delta)

IBM.PSSP.CSS.recvintr_lsw
Number of receive interrupts (lsw bits 0-30)

IBM.PSSP.CSS.recvintr_msw
Number of receive interrupts (msw bits 31-61)

IBM.PSSP.CSS.xmitintr_dlt
Number of transmit interrupts (delta)

IBM.PSSP.CSS.xmitintr_lsw
Number of transmit interrupts (lsw bits 0-30)

IBM.PSSP.CSS.xmitintr_msw
Number of transmit interrupts (msw bits 31-61)

IBM.PSSP.CSS.xmitque_cur
Sum of driver+adapter xmit queues

IBM.PSSP.CSS.xmitque_max
Max transmits ever queued

IBM.PSSP.CSS.xmitque_ovf
Number of transmit queue overflows

The following are resource values related to LAN:

IBM.PSSP.aixos.LAN.rcverrors
Count of frame receive errors at adapter level

IBM.PSSP.aixos.LAN.recvdrops
Count of receive packets dropped at device driver level

IBM.PSSP.aixos.LAN.xmitdrops
Count of transmit packets dropped at device driver level

IBM.PSSP.aixos.LAN.xmiterrors
Count of frame transmit errors at adapter level

IBM.PSSP.aixos.LAN.xmitovfl
Count of transmit queue overflows

Most of these resources are not suitable for being handled as events because they are too detailed. But normal network status monitoring commands, such as `netstat -I`, cannot get such kind of information. These resources are helpful in determining the cause of network problems.

These events are also monitored from Perspectives GUI or PTX. See *RS/6000 SP Monitoring: Keeping It Alive*, SG24-4873 for details.

4.1.8 How to Notify the Operator of Problems

There are several methods for informing the operator or users about pman problems.

mail This method is more reliable than other methods. The operator in front of the machine is aware of the problem sooner. Even if he is not there, he will know the status of the problem later. And it can be sent to a remote host. See the previous examples on how to use `notify_event` command.

wall The `wall` command can broadcast messages to all TTYs that belong to the system. All users can see the message that relates to a problem as soon as the event occurs. But if it is missed, it is then difficult to be aware of the message later on.

GUI This method is suitable for a site where the operator is in front of the machine. See Figure 53 on page 194 on how to implement this function.

SNMP (Simple Network Management Protocol) This protocol used to be managed mainly by distributed environment management tools, such as SystemView or TME10. The pman subsystem can raise an SNMP trap for each problem. The following `pmandef` command specifies that when hostResponds is down on one of the nodes, SNMP trap 1234 is generated on the Control Workstation:

```
# /usr/lpp/ssp/bin/pmandef -s Host_Responds_Status \
    -e 'IBM.PSSP.Response.Host.state:NodeNum=1-16:X==0'\
    -t 1234 -n 0
```

The trap is generated by the sp_configd daemon when the event occurs. It can be controlled with the `syspar_ctrl` command. It starts automatically when the system is booted by inittab. And the snmpd daemon should be active before sp_configd starts. Both daemons are controlled by the SRC. The `lssrc -a` command shows the status of both daemons. To start snmpd automatically when the system boots, the /etc/rc.tcpip file should contain the following line (it is commented out by default):

```
start /usr/sbin/snmpd "$src_running"
```

To specify which machine is the SNMP manager (that manages the SNMP information), /etc/snmpd.conf should be edited in advance. To find out how to edit the file, see the header part of the file or *AIX Version 4 System Management Guide: Communications and Networks*.

logging This is a passive notification method. Unless the operator tries to see the information, its existence is not known. It should be used for status information that does not indicate a problem directly. Or it should be combined with other notification methods. The -l parameter of the pmandef command should be used:

```
# pmandef -s Filesystem_Monitor
    -e 'IBM.PSSP.aixos.FS.%totused:NodeNum=*;VG=myvg;LV=mylv:x>95'
    -l "file system is almost full" -h local
```

When an event occurs, the text "file system is almost full" is written to the AIX error log and the BSD syslog on the machine where the event occurred.

4.2 Error Log Management Facility

AIX itself has error logging facilities to make problem determination easier. One is the AIX error log, the other the BSD syslog. Because every node in SP has AIX installed, these functions are, of course, available.

PSSP 2.2 provides functions that manage distributed logs from the Control Workstation as the single control point. Because these functions are built on the sysctl facility, let us do a short sysctl overview.

4.2.1 Sysctl Overview

Sysctl is an authenticated client/server system for running commands remotely and in parallel. It consists of the sysctld server daemon and a sysctl client front-end. Since the sysctld server daemon runs with UID 0, commands are executed with root privilege. Authorization is provided with the callback script. Authentication is done via SP Authentication Services (Kerberos).

For information about the sysctl utility, see section 8.3.5 in *RS/6000 SP System Management: Easy, Lean and Mean*.

The sysctl files are:

- /etc/sysctl.conf

  Contains definitions and configurations for the sysctl environment.

- /etc/sysctl.acl

  Contains configurations of ACL callback from remote nodes.

- sysctld

  The sysctld daemon, which runs on every node as root.

- sysctl

  This command is a client of the sysctld server.

- /var/adm/SPlogs/sysctl/sysctld.log

  The log file for sysctld.

- /etc/services

  The sysctld reserve port in /etc/services:

  ```
  sysctl          6680/tcp        sysctld
  ```

- /etc/inittab

This entry is made during sysctl installation.

```
sysctld:2:once:/usr/bin/startsrc -s sysctld
```

For starting with sysctl-based commands, you need to at least add an entry for root.admin in the /etc/logmgt.acl file on all nodes.

```
# cat /etc/logmgt.acl
#acl#
# This sample acl file for log management commands contains a commented
# line for a principal
_PRINCIPAL root.admin@MSC.ITSO.IBM.COM
# Principal for trimming SPdaemon.log by cleanup.logs.wsministration
_PRINCIPAL rcmd.sp2en0
```

The `pcp` command can be used to copy this file to all the nodes in the same partition.

```
# pcp -a /etc/logmgt.acl
```

See *RS6000 SP: Administration Guide*, for details.

4.2.2 Managing the AIX Error Log

The error log is a binary file. The absolute path name is /var/adm/ras/errlog. The /var/adm/ras/errtmplt file is a template that defines how the binary data in the error log is interpreted. The `errpt` command uses this template to format the log. The `errpt -t -a` command displays all template information, which shows you what error can be logged on the system.

Messages for the AIX error log are created by a kernel errsave() call, or an application errlog() system call, or by the `errlogger` command. These calls write to the /dev/error special file, that is constantly checked by errdemon, which checks it with the Error Record Template Repository, and makes an entry in the /var/adm/ras/errlog file.

There is a SMIT interface to manage the error log. `smit sperrlog` provides the following menu:

- Generates an error report
- Shows the characteristics of the error log
- Changes the characteristics of the error log
- Cleans the error log
- Adds a notification object
- Removes a notification object
- Shows a notification object
- Adds an error template
- Removes an error template
- Shows an error template

The function of a notification object is to maintain the AIX Error Notification Facility, which it does by invoking some predefined action upon occurrence of an error event.

The errdemon daemon logs the information that is taken from /dev/error. At that time, if the entry matches the criteria that is defined in the /etc/objrepos/errnotify ODM file, it takes the action indicated by the criteria.

For example, when an application program is terminated with a core dump, the following entry is added to the error log:

```
# errpt -a | more
LABEL:          CORE_DUMP
IDENTIFIER:     DE0A8DC4

Date/Time:       Thu Feb 13 16:00:30
Sequence Number: 39
Machine Id:      000504936700
Node Id:         risc71
Class:           S
Type:            PERM
Resource Name:   SYSPROC

Description
SOFTWARE PROGRAM ABNORMALLY TERMINATED

Probable Causes
SOFTWARE PROGRAM

User Causes
USER GENERATED SIGNAL

        Recommended Actions
        CORRECT THEN RETRY

Failure Causes
SOFTWARE PROGRAM

        Recommended Actions
        RERUN THE APPLICATION PROGRAM
        IF PROBLEM PERSISTS THEN DO THE FOLLOWING
        CONTACT APPROPRIATE SERVICE REPRESENTATIVE

Detail Data
SIGNAL NUMBER
           4
USER'S PROCESS ID:
       29654
FILE SYSTEM SERIAL NUMBER
          28
INODE NUMBER
       61440
PROGRAM NAME
a.out
```

To create an error notification object that sends mail when the error whose LABEL is CORE_DUMP is logged, do the following:

1. Create the file /tmp/nobj.tmp:

```
errnotify:
      en_label = CORE_DUMP
      en_method = "errpt -a -l $1 | mail -s 'CORE_DUMP' root@sp2en0"
```

2. Add it to ODM:

```
# odmadd /tmp/nobj.tmp
```

Now you can get mail when a CORE_DUMP entry is logged. To verify the entry of /etc/objrepos/errnotify, use the odmget command:

```
# odmget -q"en_label=CORE_DUMP" errnotify
errnotify:
        en_pid = 0
        en_name = ""
        en_persistenceflg = 0
        en_label = "CORE_DUMP"
        en_crcid = 0
        en_class = ""
        en_type = ""
        en_alertflg = ""
        en_resource = ""
        en_rtype = ""
        en_rclass = ""
        en_symptom = ""
        en_method = "errpt -a -l $1 | mail -s 'CORE_DUMP' root@sp2en0"
```

As the output shows, many attributes can be specified to categorize the trigger event. For example, you can specify the error class (en_class) or error type (en_type).

PSSP 2.2 provides a function that can Add, Remove or Show the entry in the errnotify ODM object class. For example, the smitty padd_en command provides a screen to add a stanza to the class:

```
                 Add a Notification Object

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                         [Entry Fields]
  Add Object to all Nodes in Partition  no             +
  Add Object to Hosts                   []
* Notify Object Name                    []
  Process ID for use by Notify Method   []             #
  Persist across system restart         no             +
  Error Label                           []             +
  Error Class                           All            +
  Error Type                            All            +
  Match Alertable Errors                All            +
  Resource Name                         []
  Resource Type                         []
  Resource Class                        []
* Notification Method                   []
```

Each field in the errnotify ODM class corresponds to a SMIT data field as follows:

en_name Notify Object Name. You can specify any name for it.

en_pid Process ID for use by Notify Method. Use this option to specify a process ID for use by the Notify Method. Generally, if the method does not influence other processes, nothing is specified.

en_persistenceflg	Persist across system restart. Unless you set this field to "yes," the definition is removed from ODM when the system is rebooted.
en_label	Error Label.
en_class	Error Class.
en_type	Error Type.
en_alertflg	Match Alertable Errors.
en_resource	Resource Name.
et_rtype	Resource Type.
en_rclass	Resource Class.
en_method	Notification Method. You can specify the command or shell script to be run when the specified error is logged.

The `penotify` command is executed from the `smitty padd_en` screen. To add an error notification object that is related to the CORE_DUMP label (as in the previous example), to Node 1:

```
# /usr/lpp/ssp/bin/penotify -f add  -w "sp2n01" -n "notify_test" \
-l "CORE_DUMP" -m "errpt -a -l  $1 | mail -s 'COREDUMP' root@sp2en0"
```

To keep the object even after the system reboots, use the `-P` flag.

For details about error notification of the error log, see *AIX V4.1 Problem Solving Guide and Reference*, SC23-2606.

4.2.3 When penotify Fails

The `penotify` command was issued from the command line as follows:

```
# penotify - f show
```

But the following error occurred:

```
>> sp21en0
sysctl:  2501-122 perrnot: Insufficient Authorization.
>>
```

This error message requires a check for insufficient authorization:

- See *IBM Parallel System Support Programs for AIX Diagnosis and Messages Guide*, SC23-3899, Section 2: "SP Messages."
- Code 2501 is for sysctl messages.
- Code 122 indicates lack of a user's authorization.

Ensure that you have no problem with Kerberos.

Examine sysctld:

- Stop sysctld with the command `stopsrc -s sysctld`

- Start sysctld with “full debug information” on:

```
# sysctld -sd -l /var/adm/sysctld.mydebug
```

- Now try the problematic command again:

```
# penotify - f show
```

- Examine the log file:

```
Feb 13 17:50:01  <33039a6c> AUDIT[var]: Creating variable SCPRINCIPAL = ″root.admin@MSC.ITSO.IBM.COM″, callBack = {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Setting callback for variable ″SCPRINCIPAL″ => {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[var]: Creating variable SCUSER = ″root″, callBack = {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Setting callback for variable ″SCUSER″ => {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[var]: Creating variable SCINSTANCE = ″admin″, callBack = {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Setting callback for variable ″SCINSTANCE″ => {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[var]: Creating variable SCREALM = ″MSC.ITSO.IBM.COM″, callBack = {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Setting callback for variable ″SCREALM″ => {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[var]: Creating variable SCHOST = ″sp21en0.msc.itso.ibm.com″, callBack = {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Setting callback for variable ″SCHOST″ => {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[var]: Creating variable SCLHOST = ″sp21en0.msc.itso.ibm.com″, callBack = {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Setting callback for variable ″SCLHOST″ => {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[var]: Creating variable SCLPRINCIPAL = ″rcmd.sp21en0@MSC.ITSO.IBM.COM″, callBack = {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Setting callback for variable ″SCLPRINCIPAL″ => {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[var]: Creating variable SCLREALM = ″MSC.ITSO.IBM.COM″, callBack = {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Setting callback for variable ″SCLREALM″ => {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[var]: Creating variable SCSCRIPT = ″perrnot show - 0 - - - 0 - - - - -
″, callBack = {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Setting callback for variable ″SCSCRIPT″ => {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[var]: Creating variable SCPRIVATE = ″0″, callBack = {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Setting callback for variable ″SCPRIVATE″ => {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[var]: Creating variable SCMODE = ″SOCKET″, callBack = {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Setting callback for variable ″SCMODE″ => {NONE}
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Setting callback for variable ″env″ => {SYSTEM}
Feb 13 17:50:01  <33039a6c> AUDIT[log]: Executing svclogevent callback ...
Feb 13 17:50:01  <33039a6c> root.admin@MSC.ITSO.IBM.COM on sp21en0.msc.itso.ibm.com
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Executing svcconnect callback ...
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Checking object authorization callback for command ″svcconnect″: AUTH
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: interp->result from object callback {AUTH} = ″Authorization OK.
″, retCode = 0
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Authorization granted for command ″svcconnect″ via object callback
Feb 13 17:50:01  <33039a6c> ==========================================================
Feb 13 17:50:01  <33039a6c> AUDIT[tcl]: Evaluating: perrnot show - 0 - - - 0 - - - - -
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Checking object authorization callback for command ″perrnot″: ACL /etc/logmgt.acl
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: interp->result from object callback {ACL /etc/logmgt.acl} = ″sysctl:  2501-139 Auth
orization
Denied.
″, retCode = 1
Feb 13 17:50:01  <33039a6c> AUDIT[sec]: Authorization denied for command ″perrnot″
Feb 13 17:50:01  <33039a6c> DEBUG: Read 55 bytes of child stderr
Feb 13 17:50:01  <33039a6c> DEBUG: Reaped child 3156, status=115
```

The line:

```
interp->result from object callback {ACL /etc/logmgt.acl} = ″sys  /etc/logmgt.acl}
= ″sysctl:  2501-139 Authorization Denied.″, retCode = 1:
```

shows that your principal has no entry in the Access Control list file, /etc/sysctl.acl.

Check who your principal is:

```
# sysctl whoami
root.admin@MSC.ITSO.IBM.COM
```

Add your principal to the /etc/sysctl.file, which can be done with the sysctl command:

```
# sysctl acladd -p root.admin
```

But this command requires authorization, so you probably have to add this line to the /etc/sysctl.acl file:

```
#acl#
# This sample acl file contains commented out lines for a principal
# and an acl file.
#_PRINCIPAL root.admin@HPSSL.KGN.IBM.COM
#_ACL_FILE /etc/mycmd.sysctl.acl
_PRINCIPAL root.admin@MSC.ITSO.IBM.COM
```

Now you should be able to use the command.

4.2.4 Managing BSD Syslog

Syslog is mainly used by daemons. If a program uses syslog to write error information, the destination of the output can be controlled by the /etc/syslog.conf file. /usr/sbin/syslogd, which is controlled by SRC (System Resource Controller), reads the file and changes the destination of the messages. For example, /etc/syslog.conf has the following entry on the Control Workstation:

```
daemon.notice           /var/adm/SPlogs/SPdaemon.log
```

daemon and notice are predefined keywords for syslog, which correspond to facility and priority, respectively. This means that the messages for system daemons whose priority level is equal to or greater than notice are logged in the /var/adm/SPlogs/SPdaemon.log file.

PSSP 2.2 provides the function that displays the syslog log file on nodes and trims the file.

But first, the /etc/syslog.conf file on all nodes should be modified to specify the destination of the log. The syslog management function reads this file and determines which file should be managed. The following example defines that all messages should go to the /var/adm/SPlogs/syslog/syslog.out file.

```
*.debug  /var/adm/SPlogs/syslog/syslog.out
```

dsh and pcp can be used to create a destination file and copy syslog.conf. In the following example, the /tmp/syslog.conf file is created on the Control Workstation to deliver it to nodes:

```
# dsh -a mkdir  /var/adm/SPlogs/syslog
# dsh -a touch  /var/adm/SPlogs/syslog/syslog.out
# pcp -a /tmp/syslog.tmp /etc/syslog.conf
```

Then, syslogd on each node should be refreshed to read the new /etc/syslog.conf:

```
# dsh -a refresh -s syslogd
```

Now, syslog writes messages to the specified file. Each file can be displayed or trimmed from the Control Workstation.

4.2.5 How to Utilize Error Log Information

The error log is used by many programs and device drivers. It is formatted by a predefined template file so that it can provide not only an error message itself, but also recommended actions to fix possible causes of the problem. But because SP has many nodes, the Errlog Management system or error notification function are not sufficient to inform the operator of the problem sooner. They should be combined with the pman (Problem Management) subsystem.

Actually, arrangements to do this have already been made. When a node is installed correctly, the following entry is automatically added to the ODM errnotify class on each node and the Control Workstation:

```
errnotify:
        en_pid = 0
        en_name = "SP_Problem_Mgmt"
        en_persistenceflg = 1
        en_label = ""
        en_crcid = 0
        en_class = ""
        en_type = ""
        en_alertflg = ""
        en_resource = ""
        en_rtype = ""
        en_rclass = ""
        en_symptom = ""
        en_method = "/usr/lpp/ssp/bin/errlog_rm $1 $2 $3 $4 $5 $6 $7 $8 $
```

This stanza means that when any log is written to the error log, the data is sent to the pman subsystem by the `errlog_rm` command. The logs that should be reported to the operator by pman are filtered when an event is subscribed to pman. The following example sends mail to root on the Control Workstation when an error log entry whose error type is PERM occurs:

```
pmandef -s Errlog_write_on_nodes  \
    -e 'IBM.PSSP.pm.Errlog:NodeNum=*:X@0!=X@P0 && X@3=="PERM"'  \
    -c /usr/lpp/ssp/bin/notify_event  \
    -n 0 -U root
```

Error type PERM means that the situation cannot be recovered automatically (permanent); it means some action is needed to remedy the problem. You can use the Errlog Management function to check the error log on the node to see whether there is some related error.

For details about pman, see 4.1, "Problem Management Subsystem" on page 187.

4.2.6 How to Use Syslog Information

Syslog is less flexible than the error log. It does not handle the log by each label, nor does it allow registering some command that is executed when the specified error is logged. Because important error information is almost covered by the error log and many PSSP 2.2 daemons make their own log file, syslog is information logging rather than error logging. But syslog also categorizes messages by error type, as does the error log. The following syslog.conf file defines two output files, one for all messages and another for high priority messages:

```
*.debug   /var/adm/SPlogs/syslog/syslog.out
*.err     /var/adm/SPlogs/syslog/syslog.err
```

err has a higher priority than debug. By this setting, all messages whose priority is equal to or higher than debug are written to the syslog.out file, and those whose priority is equal to or higher than err are written to syslog.err. However, some entries are duplicated between these files because syslog cannot handle priority in a more detailed way.

Syslog can also be configured to send all messages to one host. The following example is the syslog.conf file on nodes. It sends all messages to the Control Workstation where the syslogd daemon handles them, based on its local syslog.conf file.

```
*.debug   @sp2en0
```

With the above settings, syslog messages become more useful for getting information for problem determination, or for monitoring each daemon′s status. The following are the major programs in PSSP 2.2 that log information into syslog:

- haemd
- hagsd
- hagsglsmd
- hardmon
- hrd
- jmd
- pmand
- sdrd
- sp_configd
- splogd
- sup
- supfilesrv
- xntpd
- harmld
- harmpd
- hmrmd

Note

Many of these also make their own log files under /var. See *IBM Parallel System Support Programs for AIX Command and Technical Reference*, GC23-3900 for details.

4.2.7 Cron Settings to Trim Various Error Logs

PSSP 2.2, during installation, adds some entry into crontab for trimming SP error logs:

```
# @(#)08 1.15.1.3 src/bos/usr/sbin/cron/root, cmdcntl,bos411,9428A410j
#
# COMPONENT_NAME: (CMDCNTL) commands needed for basic system needs
#
# FUNCTIONS:
#
# ORIGINS: 27
#
# (C) COPYRIGHT International Business Machines Corp. 1989,1994
# All Rights Reserved
# Licensed Materials - Property of IBM
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
#0 3 * * * /usr/sbin/skulker
#45 2 * * 0 /usr/lib/spell/compress
#45 23 * * * ulimit 5000; /usr/lib/smdemon.cleanu > /dev/null
0 11 * * * /usr/bin/errclear -d S,O 30
0 12 * * * /usr/bin/errclear -d H 90
01 5 * * * /usr/lpp/diagnostics/bin/run_ssa_ela 1>/dev/null 2>/dev/null
0 * * * * /usr/lpp/diagnostics/bin/run_ssa_healthcheck 1>/dev/null 2>/dev/null
0 0  * * *  /usr/lpp/ssp/bin/cleanup.logs.ws
```

The first two non-commented lines are inteded to trim AIX Errlog (/var/adm/ras/errlog). The first one clears, 11 minutes after midnight each day, all software (S) and errorlogger (O) entries that are older than 30 days from the error log file.

The second line clear, 12 minutes after midnight each day, all hardware related (H) entries that are older that 90 days.

The last line is a sysctl-based script for trimming various SP error logs. It:

- Truncates AMD logs on the CWS to 400 lines at midnight each night.
- Deletes an SDR configuration log if over a week old at midnight each night.
- Deletes the oldest SDR server log if over a week old and not the current (only) one.
- Deletes startup/shutdown logs if over a week old at midnight each night.
- Closes the splogd state change log each night. Deletes state change logs older than one week.
- Changes the hardware monitor log each night. Deletes hardware monitor logs older than a week, or older than one day if they are bigger than 32000 bytes.

- Deletes entries from the syslog daemon facility log file that are older than six days.
- Deletes supper logs if over two days old at midnight each night.

Chapter 5. Diagnosing Switch and Partitioning Problems

This chapter provides you with a path to follow to help you diagnose any switch problems you may encounter.

Because the only way to determine whether there is a problem is through the symptoms you see, we list these possible symptoms in Table 5 on page 214 and Table 6 on page 214. By following the appropriate links, you can follow a path through to problem determination.

PSSP 2.2 supports two types of switch, the High Performance Switch (HPS) and the newer Performance Switch (SPS).

In some cases, the symptoms or output of files are specific for each type of switch. In those cases, a separate table is provided for each.

In other situations, the type of switch is not a relevant factor; in those situations, a generic sequence of steps or output is followed or examined.

It is possible that the problem could remain unresolved and undiagnosed even after you follow the problem determination paths, because we cannot cover every scenario in this chapter. In such cases, we recommend that you read and follow the advice given in 5.9, "Collecting Information for IBM Software Support" on page 230, and then contact IBM Software Support.

Note: Any action that you are requested to perform assumes that root authority is granted on either the nodes or the Control Workstation (CWS).

Table 5 lists the most common HPS failure symptoms.

Table 5. HPS Failure Symptoms

Symptom	**Action**
Estart failures: System cannot find Estart command. Primary node not reachable. Estart fails or times out. Expected number of nodes not initialized, or not all links initialized.	See: 5.1, “Software Installation Verification” on page 215. 5.2, “Verifying an HPS Node” on page 216. 5.16, “Diagnosing HPS Estart Problems” on page 241. 5.13, “Device and Link Problems” on page 234.
Node drops off the switch during normal operation, switch_responds is no.	See 5.2, “Verifying an HPS Node” on page 216.
Node fails to communicate over the switch, but switch_responds is yes (ping or CSS_test fail).	See 5.2, “Verifying an HPS Node” on page 216 and 5.15, “Isolating Adapter and Switch Errors” on page 237.
Node crashes.	See 5.10, “Node Crash” on page 232.
Node fails to Eunfence.	See 5.12, “Eunfence Problems” on page 233.
Other Ecommand failures.	See 5.14, “Ecommand Problems” on page 236.

Table 6 lists the most common SP failure symptoms.

Table 6. SPS Failure Symptoms

Symptom	**Action**
Estart failures: System cannot find Estart command. Primary node not reachable. Estart fails or times out. Expected number of nodes not initialized, or not all links initialized.	See: 5.1, “Software Installation Verification” on page 215. 5.3, “Verifying an SPS Node” on page 219. 5.17, “Diagnosing SPS Estart Problems” on page 242. 5.13, “Device and Link Problems” on page 234.
Node drops off the switch during normal operation, switch_responds is no.	See 5.3, “Verifying an SPS Node” on page 219.
Node fails to communicate over the switch but switch_responds is yes (ping or CSS_test fail).	See 5.3, “Verifying an SPS Node” on page 219 and 5.15, “Isolating Adapter and Switch Errors” on page 237.
Node crashes.	See 5.10, “Node Crash” on page 232.
Node fails to Eunfence.	See 5.12, “Eunfence Problems” on page 233.
Other Ecommand failures.	See 5.14, “Ecommand Problems” on page 236.
Oncoming Primary node is Efenced.	See 5.11, “Eunfencing the Oncoming Primary Node” on page 232.

5.1 Software Installation Verification

Software installation and verification is done using the CSS_test script from either the System Management Interface Tool (SMIT) panel or from the command line. If CSS_test is executed following a successful Estart, additional verification of the system is done to determine if each node in the system or system partition is able to be *pinged*. If you are using system partitions, CSS_test runs in the active partition only. For more information on managing system partitions, see *RS/6000 SP: Administration Guide*, GC23-3897.

To verify CSS installation from SMIT:

1. Enter the following:

   ```
   -> smitty SP_verify
   ```

   The RS/6000 SP Installation/Configuration Verification menu is displayed.

2. Select the **Communication Subsystem** option.

3. Review the output even if the command return status is OK.

To verify CSS installation from the command line enter:

```
-> /usr/lpp/ssp/bin/CSS_test
```

The command can be invoked with the options:

- -q to suppress messages
- -l to designate an alternate log file other than the default location of /var/adm/SPlogs/css/CSS_test.log.

Some additional items to consider while trying to run CSS_test are:

- Each node should have local access to the /usr/lpp/ssp directory.
- The local /etc/inittab for each node should contain an entry for rc.switch file similar to the following:

  ```
  fsd:2:once:/usr/lpp/ssp/css/rc.switch
  ```

For complete information on CSS_test, see the *RS/6000 SP: Command and Technical Reference*, GC23-3900.

5.2 Verifying an HPS Node

This procedure should be used to verify that a single HPS node is operating correctly. If the node you are attempting to verify is the primary node, start with step one. If it is a secondary node, start with step 2.

1. Determine which node is the primary node by issuing the `Eprimary` command on the Control Workstation. The output returned is similar to that shown in Figure 54.

```
-> Eprimary
1
```

Figure 54. Eprimary Query Output for HPS

   In this example, you can see that the primary node is node 1.
   If the primary node is not available (for example, if it is powered off), then `Eprimary` should be run to designate a new primary, followed by an `Estart` to start the switch. `Eprimary` should never be used to designate a new primary without running `Estart` if the current primary node is operational. The reason for this is that the output of querying which node is the primary will then report back a node that is not the current running primary node. The value reported will be the primary after the next start of the switch.

   `Eprimary` can always be used to query the designated primary node.

   Note

   On a primary node failure, the switch and switch_responds System Monitor display is inconsistent until `Eprimary` and `Estart` are run.

2. Ensure that the node that is being verified is accessible from the Control Workstation through Kerberos by using the Kerberos `rsh` command to remotely run a command to the node. Figure 55 shows an example.

```
-> /usr/lpp/ssp/rcmd/bin/rsh sp2n01 date
Tue Feb 11 16:53:02 EST 1997
```

Figure 55. Testing Node Accessibility

   If the command fails to return the current time and date, then you need to resolve this issue before continuing. Refer to *RS/6000 SP: Diagnosis and Messages Guide*, GC23-3899 or to *RS/6000 SP: Problem Determination Guide*, SG24-4778 for details on how to resolve Kerberos problems.

3. Verify that the switch adapter (css0) is configured and is ready for operation on the node. This is done by interrogating the adapter_config_status attribute in the switch_responds object class of the System Data Repository (SDR) using the `SDRGetObjects` command with the syntax of:

   `SDRGetObjects switch_responds node_number==<node>`

   Alternatively, if you only have a few nodes in your partition, you can list the entire switch_responds class by not specifying the node_number attribute. If you are unsure of the node number, you can procure it again from a different object class with `SDRGetObjects`. Figure 56 on page 217 shows an example.

```
-> SDRGetObjects Node node_number initial_hostname==sp21n01
node_number
          1
```

Figure 56. Find the Node Number from the Node Object Class in the SDR

Once you know the node_number, you can then proceed to check the adapter status as shown in the next example. The adapter_config_status should be seen as css_ready for all nodes at PSSP level 2.2 and 2.1, as shown in Figure 63 on page 220. If they are not, then refer to 5.6, “Configuration and Diagnostic Problems” on page 223. Nodes at PSSP level 1.2 will show the output in Figure 57.

```
-> SDRGetObjects switch_responds node_number==1
node_number  switch_responds autojoin     isolated     adapter_config_status
          9            1            0 ””           ””
```

Figure 57. Output from the switch_responds Objects

4. Verify that the fault_service_Worm_RTG daemon and the fs_monitor are running on the node by using the Kerberos rsh command again to issue a ps to the problem node, as shown in Figure 58.

```
-> /usr/lpp/ssp/rcmd/bin/rsh sp2n01 ps -ef | grep css
    root 12678     1   0   Jan 24      -  0:47 /usr/lpp/ssp/css/fs_monitor -t2
    root 14978     1   0   Jan 24      -  0:53 /usr/lpp/ssp/css/fault_service_Wor
m_RTG -r 0 -s 100015 -p 3 -a TB2 -d SW
```

Figure 58. Checking That the Worm is Running on the Node

If the fault_service_Worm_RTG daemon and fs_monitor daemon are running, the HPS node verification is complete. If one of the daemons is not running, then you can try recovering the node with the following steps.

a. Attempt to restart the fault_service_Worm_RTG and fs_monitor daemons on the node by logging on a running /usr/lpp/ssp/css/rc.switch. You should then see the screen shown in Figure 59.

```
-> /usr/lpp/ssp/css/rc.switch
”adapter/mca/tb2”
main(): (parent) we are on switch chip 100015, port 3
main(): (parent) we are switch node number 0
main(): (child) signal handlers (SIG_IGN) set up successfully
main(): (child) setsid() successful, daemon process group ID = 15018
main(): (parent) fork() successful, child PID (daemon PID) = 15018
main(): (parent) parent returns 0, child (daemon) continues ...
fs_monitor: specified time = 2
fs_monitor(): Started.  PID = 14766, timer = 2 secs.
```

Figure 59. Running rc.switch on the Node

b. Check once again that the fault_service_Worm_RTG and fs_monitor daemons are running. If both are listed, you should run an Estart to get the node back onto the switch.

   If one of the daemons has died again, try to determine why that particular daemon has a problem. Refer to 5.15, “Isolating Adapter and

Switch Errors” on page 237, and diagnose this node instead of the primary node.

Following are possible reasons why the fault_service_Worm_RTG daemon is not running:

- The daemon exited due to an abnormal error condition.
- A SIGTERM, SIGBUS or SIGDANGER signal was processed by the daemon.

5.3 Verifying an SPS Node

This procedure should be used to verify that a single SPS node is operating correctly. If the node you are attempting to verify is the primary node, start with step one. If it is a secondary node, start with step 2.

1. Determine which node is the primary node by issuing the `Eprimary` command on the Control Workstation. The output returned should look similar to that shown in Figure 60.

```
-> Eprimary
1       - primary
1       - oncoming primary
8       - primary backup
7       - oncoming primary backup
```

Figure 60. Eprimary Query Output for SPS

   If the command returns an oncoming primary value of none, re-execute the `Eprimary` command, specifying the node you wish to have as the primary node. Following the execution of the `Eprimary` command (to change the oncoming primary), an `Estart` is required to make the oncoming primary node the primary node.

   If the command returns a primary value of none, an `Estart` is required to make the oncoming primary node the primary.

   The primary node on the SPS system can have moved to another node if a primary node takeover is initiated by the backup. To determine if this has happened, look at the values of the primary and the oncoming primary backup. If they are the same value, then a takeover has occurred. All SDR switch_responds objects are kept up-to-date during this time.

2. Ensure that the node that is being verified is accessible from the Control Workstation through Kerberos by using the Kerberos `rsh` command to remotely run a command to the node, as shown in Figure 61.

```
-> /usr/lpp/ssp/rcmd/bin/rsh sp21n01 date
Tue Feb 11 15:05:02 CST 1997
```

Figure 61. Testing that the Node is Accessible

   If the command fails to return the current time and date, then you need to resolve this issue before continuing. Refer to *RS/6000 SP: Diagnosis and Messages Guide*, GC23-3899 or *RS/6000 SP: Problem Determination Guide*, SG24-4778 for details on how to resolve Kerberos problems.

3. Verify that the switch adapter (css0) is configured and is ready for operation on the node. This is done by interrogating the adapter_config_status attribute in the switch_responds object class of the System Data Repository (SDR), using the `SDRGetObjects` command with the syntax of:

   `SDRGetObjects switch_responds node_number==<node>`

   Alternatively, if you only have a few nodes in your partition, you can list the entire switch_responds class by not specifying the node_number attribute. If you are unsure of the node number, you can procure it again from a different object class with `SDRGetObjects`. Figure 62 on page 220 shows an example.

```
-> SDRGetObjects Node node_number initial_hostname==sp21n01
node_number
          1
```

Figure 62. Find the Node Number from the Node Object Class in the SDR

Once you know the node_number, you can then proceed to check the adapter status as shown in the next example. If the adapter_config_status is anything other than css_ready, then see 5.6, "Configuration and Diagnostic Problems" on page 223.

```
-> SDRGetObjects switch_responds node_number==1
node_number  switch_responds autojoin     isolated     adapter_config_status
          1            1            0            0 css_ready
```

Figure 63. Checking the Adapter Status in the switch_responds Objects Class in the SDR

4. Verify that the fault_service_Worm_RTG_SP daemon is running on the node by using the Kerberos rsh command again to issue a ps to the problem node, as shown in Figure 64.

```
-> /usr/lpp/ssp/rcmd/bin/rsh sp21n01 ps -ef | grep css
    root 14304     1   0   Feb 06      -  0:01 /usr/lpp/ssp/css/f
rm_RTG_SP -r 0 -b 1 -s 5 -p 3 -a TB3 -t 25
```

Figure 64. Checking To See If the Worm is Running on the Node

If the fault_service_Worm_RTG_SP daemon is running the SPS node, verification is complete. If the fault_service_Worm_RTG_SP daemon is not running, then you can try to recover the node with the following steps.

a. Attempt to restart the fault_service_Worm_RTG_SP daemon on the node by logging on a running /usr/lpp/ssp/css/rc.switch. You should then see the screen shown in Figure 65.

```
-> /usr/lpp/ssp/css/rc.switch
"adapter/mca/tb3"
```

Figure 65. Running rc.switch on the Node

b. Check once again to verify that the the fault_service_Worm_RTG_SP daemon is running. If it is, unfence the node with:

   `Eunfence <node_number>`

   If the daemon has died again, try to determine why that particular daemon has a problem. Refer to 5.15, "Isolating Adapter and Switch Errors" on page 237, and diagnose this node instead of the primary.

   Following are possible reasons why the fault_service_Worm_RTG daemon is not running:

   - The daemon exited due to an abnormal error condition.
   - A SIGTERM, SIGBUS or SIGDANGER signal was processed by the daemon.

5.4 Verifying Switch Topology Configuration

The switch topology file is used to define the hardware configuration to the CSS support software. It should reflect the number of switches and nodes installed, as well as define how they are connected. The topology file is partition-specific, meaning that only the information relevant to the current partition is displayed.

The topology file can reside in two places: in the SDR, or as expected.top in the /etc/SP directory of the primary node.

The configuration in the SDR is what is commonly used. The configuration in /etc/SP/expected.top is generally used for debug purposes.

Important

You should avoid having /etc/SP/expected.top on the primary node.

If /etc/SP/expected.top exists on the primary node, it overrides the configuration in the SDR.

To verify that the topology in the SDR is correct, it should first be read out of the SDR by issuing the Etopology command that follows:

```
Etopology -read <output file>
```

The Etopology command will read the switch topology from the SDR and place it into the specified file. You will now be able to read that file and check that the information contained is a correct representation of the installed hardware setup. If changes to the switch topology file are required, remember to place them back into the SDR after they have been completed using Etopology.

For more information on the Etopology command, see *RS/6000 SP: Command and Technical Reference*, GC23-3900.

5.5 Verifying the System Data Repository (SDR)

To verify that the SDR is installed and operating correctly, you can run SDR_test on the Control Workstation. It is run either through SMIT or from the command line.

To verify the SDR from SMIT, enter:

```
-> smitty SP_verify
```

The RS/6000 SP Installation/Configuration Verification menu is displayed.

- Select the **System Data Repository** option.
- Review the output even if the command return status is OK.

To verify the SDR from the command line, enter:

```
-> /usr/lpp/ssp/bin/SDR_test
```

Next log into the failing node and issue the SDRGetObjects command against the switch_responds object. The command should be as follows:

```
-> SDRGetObjects switch_responds
```

Examine the output that is returned. If the switch responds bits are returned, the SDR is operating. Additionally, you can determine which nodes are operational on the switch: a value of 1 indicates a node is operational. A value of 0 indicates a node is non-operational.

5.6 Configuration and Diagnostic Problems

Table 7 is based on the possible values of the adapter_config_status attribute of the switch_responds object class of the SDR. To obtain the value, issue the following command from the Control Workstation:

```
-> SDRGetObjects switch_responds
```

Use the value of the adapter_config_status attribute for the node in question to index into the table.

Note

In the adapter_config_status table that follows, the phrase *adapter configuration command* is used. This refers to the HPS or SPS adapter configuration method. The syntax to use when invoking these methods is as follows:

- For HPS: /usr/lpp/ssp/css/cfgtb2 -v -l css0 > <output_filename>
- For SPS: /usr/lpp/ssp/css/cfgtb3 -v -l css0 > <output_filename>

Table 7 (Page 1 of 2). Adapter_config_status Message

adapter_config_status	**Explanation and Recovery action**
odm_fail genmajor_fail genminor_fail getslot_fail build_dds_fail	An Object Data Manager (ODM) failure has occurred while configuring the CSS adapter. Re-run the adapter configuration command. If the problem persists, contact IBM Software Support and supply them with the output file from the command.
lname_error	The device logical name specified on the CSS adapter configuration command was invalid. Re-run the adapter configuration command. If the problem persists, contact IBM Software Support and supply them with the output file from the command.
undefine_system_fail define_system_fail xilinx_system_fail	The System Standard C Library Subroutine failed during CSS adapter configuration. Re-run the adapter configuration command. If the problem persists, contact IBM Software Support and supply them with the output file from the command.
undefine_fail define_fail	The current instance of the CSS logical device could not be redefined. Try removing manually, using the same procedure as used for normal network devices. Then re-run the adapter configuration command. If the problem persists, contact IBM Software Support and supply them with the output file from the command.
chkslot_fail	Verify that the CSS adapter is properly seated. (You may need the assistance of a hardware engineer to do this.) Then re-run the adapter configuration command. If the problem persists, contact IBM Software Support and supply them with the output file from the command.
busresolve_fail	There are insufficient bus resources to configure the CSS adapter. Contact IBM Software Support.
xilinx_load_fail dd_load_fail fs_load_fail	See 5.1, “Software Installation Verification” on page 215. If the Software Installation Verification is successful and the problem persists, contact IBM Software Support.
make_special_file	The CSS device special file could not be created during adapter configuration. Re-run the adapter configuration command. If the problem persists, contact IBM Software Support and supply them with the output file from the command.

Table 7 (Page 2 of 2). Adapter_config_status Message

adapter_config_status	**Explanation and Recovery action**
dd_config_fail fs_init_fail	An internal device driver error occurred during CSS adapter configuration. Use the section 5.9, “Collecting Information for IBM Software Support” on page 230 to collect all the pertinent information required and then contact IBM Software Support.
diag_fail	See 5.7, “Adapter Diagnostic Failures” on page 225.

5.7 Adapter Diagnostic Failures

The recovery actions to take for adapter diagnostic failure can, in most cases, be determined by examining the Service Request Number (SRN) posted in the error log entry for the diagnostic failure. To get this information, log into the failing node as root and issue the `errpt -a` command to view detailed information on the failing entry. Figure 66 shows a sample error log entry.

```
-------------------------------------------------------------------------
ERROR LABEL:    HPS_DIAG_ERROR1_ER
ERROR ID:       945BD4F7
Date/Time:       Mon Feb 10 20:49:16
Sequence Number: 3188
Machine Id:      000041005700
Node Id:         sp2n06
Error Class:     H
Error Type:      PERM
Resource Name:   css0
Resource Class:  adapter
Resource Type:   tb2
Location:        00-08
Error Description
HPS adapter failed POST diagnostics
Probable Causes
Switch clock signal missing
HPS adapter failure
User Causes
Switch cable disconnected
        Recommended Actions
        Run adapter diagnostics
Failure Causes
HPS adapter
        Recommended Actions
        Run adapter diagnostics
Detail Data
DETECTING MODULE
LPP=PSSP,Fn=dtb2.c,SID=1.35,L#=1379,
Service Request Number
83c-562
------------------------------------------------------------------------
```

Figure 66. Sample error log Entry for HPS Adapter Failing Diagnostics

Now you can use the SRN number listed at the bottom of the error log entry to reference against Table 8, where the *x*'s should be interpreted as any value:

Table 8. Switch Adapter Failure Types

SRN	Actions
1xx	See 5.1, “Software Installation Verification” on page 215. If the verification is successful and the problem persists, contact IBM Software Support.
28x	See 5.8, “External Clock and Cable Verification” on page 227. If the verification is successful and the problem persists, contact IBM Software Support.
Axx	See 5.8, “External Clock and Cable Verification” on page 227. If the verification is successful and the problem persists, contact IBM Software Support.
All other SRN values	Contact your local Hardware Support to arrange for the adapter to be replaced.

You can also run diagnostics manually from the command line to doublecheck the output seen in the error log.

Important

Running the complete set of CSS adapter diagnostics on a particular node requires the exclusive use of the CSS adapter. You may have to kill processes that currently have the CSS device driver open. This would include the fault_service_Worm_RTG_SP on an SPS node, and both the fault_service_Worm_RTG and fs_monitor on an HPS node. Any other processes using the adapter would also have to be terminated, such as switch clock reader applications.

To run the same sequence of adapter tests that were run at Power On Self Test (POST) time, enter the following:

```
-> diag -c -d css0
```

This will run the css adapter diagnostics in unattended mode, echoing all written results and messages to standard out.

When cable or adapter problems are suspected and the POST diagnostics run successfully, it may be a good idea to run the adapter and adapter cable wrap test found in advanced diagnostics. To perform this test, enter the following from the command line:

```
-> diag -A -d css0
```

Note

You will need both the card and cable wrap plugs to complete these tests.

5.8 External Clock and Cable Verification

The following procedures should *not* be run on nodes that are operational on the switch. The utilities used for these verifications cannot coexist with normal switch operations on the node.

If clocking problems exist on all nodes in an entire frame, refer to 5.8.6, “Frame or System Clocking Problems” on page 230 for procedures. Otherwise, start by running the appropriate clock verification procedure for the problem node type in question.

5.8.1 HPS External Clock Verification

Use the following to determine if the external clock is operational at the node:

1. Login to the node.
2. Execute the following command, as shown in Figure 67.

```
-> /usr/lpp/ssp/css/read_regs

 Communications Adapter information
Kernel extension /usr/lpp/ssp/css/fault_service is loaded.

POS REGS :  0  1  2  3  4  5  6  7 31 32 33 34 35 36
CONTENTS : 7C 8F CF 6E 6F BC 00 05,A0 00 01 FF 6E 68

Clock signal present  --- GORE clock selected

MSMU registers:
     XMIT_FREE_SPACE       01F807F8
     RECV_FULL_SPACE       00000000
     INTERRUPT_PENDING     00000000
     INTERRUPT_MASK        00006880
     MSMU_CONTROL          141E0601
     MSMU_STATUS           00000800
     FAULT_ADDR            00001C00
     MSMU_DIAG             000000FF
```

Figure 67. Checking for External Clock Source on HPS Node

   The POS bytes are numbered from left to right starting at 0. The output seen from running read_regs may differ from the previous screen, depending on the level of ssp.css running on the node. However, the contents will remain the same.

3. Examine bit 7 of POS register 5 and verify that the bit is set to “0,” which is off.

   Look at the output shown in Figure 67. You can see that POS register 5 has a *hex* value of BC.

   What you now need to do is convert these register values into *binary* and examine the bits. See Figure 68 on page 228 for an example.

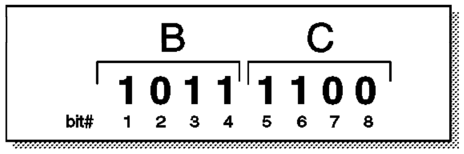


Figure 68. Converting Register Value from Hex to Binary

Here you can see that indeed bit 0 is set to off.

4. Examine bit 0 of POS register 3.1 (or 31, in this case), and verify that the bit is set to "1," which is on, as in the previous step.

5. If both of these conditions are correct, then the external clock is operational at the node. If either one of these conditions is not correct, then the external clock is not operational. Refer to 5.8.3, "HPS External Clock Restoration" for instructions on how to restore the external clock at the HPS node.

5.8.2 SPS External Clock Verification

Use the following to determine if the external clock is operational at the node:

1. Login to the node.

2. Execute the following command:

```
-> /usr/lpp/ssp/css/diags/read_tbic -s
TBIC status register        : 78000000
```

   The bits are numbered from left to right, starting at 0.

3. Examine bits 3 and 4. If they are "11," the external clock is operational at the node. If either bit is off, the external clock is not operational at the node. Refer to 5.8.4, "SPS External Clock Restoration" on page 229 for instructions on how to restore the external clock at the node.

5.8.3 HPS External Clock Restoration

You can try the following to restore the external clock at the node:

1. Execute `css_restart_node` on the problem node as follows:

```
-> /usr/lpp/ssp/css/css_restart_node
```

   This will reconfigure the adapter and restart the Worm and fs_monitor.

2. Determine whether the clock is now present with 5.8.1, "HPS External Clock Verification" on page 227.

3. If the clock still is not present, try running the `Eclock` command as shown below. For more information on the `Eclock` command refer to *RS/6000 SP: Command and Technical Reference*, GC23-3900.

```
-> Eclock -d
```

   Warning

   The `Eclock` command will effect all switch boards in the system and requires exclusive use of the switch. This should not be run if applications are currently running across the switch.

4. Again determine whether the clock is now present using 5.8.1, "HPS External Clock Verification" on page 227.

5. If the clock is still not present, then there are two courses of action. Either run the 5.8.5, "Cable Verification" on page 229 if you feel qualified, or contact your local Hardware Support to perform the verification themselves.

5.8.4 SPS External Clock Restoration

You can try the following to restore the external clock at the problem node:

1. Run the `/usr/lpp/ssp/css/rc.switch` command
2. Determine if the clock is now present by following the steps given in 5.8.2, "SPS External Clock Verification" on page 228.
3. If the clock still is not present, try running the `Eclock` command as shown below. For more information on the `Eclock` command, refer to *RS/6000 SP: Command and Technical Reference*, GC23-3900.

   ```
   -> Eclock -d
   ```

   Attention

   The `Eclock` command will affect all switch boards in the system and requires *exclusive* use of the switch. This should not be run if applications are currently running across the switch.

4. Again determine if the clock is now present by following the steps given in 5.8.2, "SPS External Clock Verification" on page 228.
5. If the clock is still not present, then there are two courses of action. Either run 5.8.5, "Cable Verification" if you feel qualified to do so, or contact your local Hardware Support to perform the verification.

5.8.5 Cable Verification

This section covers node-to-switch and switch-to-switch cables, for both HPS and SPS systems.

5.8.5.1 Node-to-Switch Cable Verification

The first step is always to visually verify the cable in question. This is performed as follows:

1. Remove the cable from the back of the node and examine the connectors (on the cable and on the back of the adapter) for bent pins or other visible damage. If every thing looks okay, reconnect the cable to the adapter, if there is damage contact your local Hardware Support and have them replace or repair the damaged components.
2. Remove the cable from the back of the switch and examine the connectors (on the cable and on the switch bulkhead jack) for bent pins or other visible damage. If every thing looks okay, reconnect the cable to the switch bulkhead jack, if there is damage contact your local Hardware Support and have them replace or repair the damaged components.
3. If everything checks out visually, run Advanced Diagnostics on the suspect adapter and cable. The procedure for doing this is outlined in 5.7, "Adapter Diagnostic Failures" on page 225. Follow the online instructions. If the diagnostics identify a failure contact IBM Hardware Service and have the failing component(s) replaced. If diagnostics pass and the problem persists contact IBM Software Support.

5.8.5.2 Switch-to-Switch Cable Verification

The first step is always to visually verify the cable in question. This is performed as follows:

1. Remove the cable from the back of the switch and examine the connectors (on the cable and on the switch bulkhead jack) for bent pins or other visible damage. If every thing looks okay, reconnect the cable to the switch bulkhead jack, if there is damage contact your local Hardware Support and have them replace or repair the damaged components.
2. Repeat step 1 for the other end of the switch-to-switch cable
3. If everything checks out visually, run Advanced Diagnostics on the suspect adapter and cable. If the problem persists, contact IBM Software Support.

5.8.6 Frame or System Clocking Problems

The following list shows the possible causes as to why an entire frame (or frames), or even the whole system has a clock loss problem.

- The Switch may be powered off.
  Check the power on the switch. Repower if necessary and execute `Eclock -d`, and then restart the Switch with an `Estart`.
- The Switch is not eclocked.
  Run `Eclock -d`, and then restart the Switch with an `Estart`.
- The clock topology file does not match the physical system hardware topology setup or the master clock switch is bad. Contact your local Hardware Support.

5.9 Collecting Information for IBM Software Support

The css.snap script collects log files created by switch support code such as device drivers, the Worm, diagnostic outputs and so on, into a single package. Under normal circumstances the following files are collected:

- cable_miswire
- cable_miswire.old
- core, which is the fault service daemon dump file
- css.snap.log
- daemon.stderr
- daemon.stdout
- dtbx.trace
- dtbx.failed.trace
- errpt.out
- flt
- fs_daeon_print.file
- netstat.out, this is the current output of `netstat -I css0` and `netstat -m`
- out.top
- rc.switch.log
- regs.out

- router.log
- scan_out.log
- scan_save.log
- tb_dump.out
- vdidl.out
- worm.out

The files ending with the .out extension are produced by running the appropriate command to dump internal memory trace information or dump data to a file. The completed output file is found in the /var/adm/SPlogs/css directory and will have the naming convention of css.snap.<date-time>.tar.Z.

The `css.snap` command is called automatically from the fault service daemon when certain serious errors are detected. However it can also be issued from the command line when a switch or adapter related problem is indicated with:

```
-> /usr/lpp/ssp/css/css.snap
```

Important

`css.snap` uses a number of undocumented utilities to collect the information required. Some of these such as the `read_regs` and `tbXdump` routines can have a disruptive effect when used on a running system. After running css.snap to collect diagnostic information, it is advisable to run `/usr/lpp/ssp/css/rc.switch` in order to reset or reload the switch adapter and eliminate the residual effects of these utilities.

The `css.snap` command avoids filling up the /var file system by using the following rules:

- If less than 10% of /var is free, then `css.snap` exits.
- If the css portion of /var is more than 30% of the total size of /var, then `css.snap` erases the old snap files until the css portion shrinks below 30% If this is successful, `css.snap` continues; otherwise, it exits.

Taking these rules into consideration, it is a good idea to give the /var file system plenty of free space. A general recommendation is to allocate at least 20Mb for the css files alone, especially on the primary node.

5.10 Node Crash

A node crash is generally identified by the LED display flashing 888 on the node. The recovery action for this event is to save the dump that was created, so that if the node crashes again, you do not lose the current dump. This procedure is documented in the *RS/6000 SP: Administration Guide*, GC23-3897. Once the dump is saved, contact IBM Software Support and supply them with the tar image in the /tmp/ibmsupt directory.

5.11 Eunfencing the Oncoming Primary Node

If the oncoming primary node becomes fenced from the switch use one of the following procedures to unfence it prior to Estarting:

- If the switch is up and operational with another primary node in control of the switch, then just Eunfence the oncoming primary and Estart to make it the active primary node as shown in Figure 69.

```
-> Eunfence sp21n01
All nodes successfully unfenced.
-> Estart
 Estart: Oncoming primary != primary, Estart directed to oncoming primary
Switch initialization started on sp21n01.
Initialized 5 node(s).
Switch initialization completed.
```

Figure 69. Unfencing and Starting the Switch for the Oncoming Primary Node

- If the switch is not operational and Estart is failing because the oncoming primary′s switch port is fenced, you must first change the primary to another node with Eprimary that is not fenced and then run an Estart. Once the switch is operational you can then Eunfence the original oncoming primary node and make it the active primary by issuing an Eprimary command followed by an Estart. Figure 70 shows an example.

```
-> Eprimary 1
 Eprimary: Defaulting oncoming primary backup node to sp21n08.msc.itso.ibm.com

-> Switch initialization started on sp21n01.
Initialized 2 node(s).
Switch initialization completed.
-> Eunfence 7
All nodes successfully unfenced.
-> Eprimary 7
 Eprimary: Defaulting oncoming primary backup node to sp21n08.msc.itso.ibm.com

-> Estart
 Estart: Oncoming primary != primary, Estart directed to oncoming primary
Switch initialization started on sp21n07.
Initialized 3 node(s).
Switch initialization completed.
```

Figure 70. Starting the Switch When the Oncoming Primary is Fenced

5.12 Eunfence Problems

You can isolate and correct most Eunfence problems by using the actions described in 5.14, “Ecommand Problems” on page 236. The following list indicates some additional reasons for an Eunfence command to fail on a particular node:

- Remember that you *cannot* unfence any nodes from a switch that has not been Estarted. Check that at least one node is running on the switch.
- The node is no longer reachable through the switch network. Refer to 5.13, “Device and Link Problems” on page 234 for help.
- If an SPS node fails to Eunfence, it could be because the switch topology could not be distributed. See *RS/6000 SP: Diagnosis and Messages Guide*, GC23-3899 or *RS/6000 SP: Problem Determination Guide*, SG24-4778 for details on how to resolve Kerberos problems.
- The node fails to respond when you attempt to Eunfence it. 5.2, “Verifying an HPS Node” on page 216 or 5.3, “Verifying an SPS Node” on page 219 may be useful in isolating and correcting the problem.
- If these procedures fail to resolve the problem, the css logs from both the primary node and the node attempting to be Unfenced should be gathered. This is accomplished by logging into these nodes and using the procedure outlined in 5.9, “Collecting Information for IBM Software Support” on page 230.

5.13 Device and Link Problems

The starting point in evaluating device and link problems on the system is to examine the out.top file in the /var/adm/SPlogs/css directory of the primary node. The out.top file will look similar to a switch topology file, except for the additional comments on lines where either the device or link is not operational. These additional comments are appended to the file by the fault_service daemon to reflect the current device and link status of the system. If there are no comments on any of the lines, or the only comments are for wrap plugs where they actually exist, all devices and links should be considered operational. However, if this is not the case, the following information should help resolve the problems.

Figure 71 is an example of a failing entry in the out.top file:

```
s 15 2  tb0 1 0           E01-S17-BH-J8 to E01-N02   -4 R: device has
been removed from network - faulty (link has been removed from network - fenced)
```

Figure 71. Example of a Failing Entry in the out.top File

This figure is read as follows:

- Switch chip 15, port 2 is connected to switch node number 1. The switch is located in frame E01 slot 17. Its bulkhead connection to the node is jack 8.
- The node is also in frame E01 and its node number is 02.
- -4R refers to the device status of the right side device, tb0 1. It has the worst device status of the two devices listed.
- The device status of the node is “device has been removed from the network - faulty.”
- The link status is “(link has been removed from the network or mis-wired - fenced).”

  The possible device and link status are listed in the next table with possible recovery actions:

Table 9 (Page 1 of 2). Device Status as Listed in out.top

No.	Comments	Explanation and Action
−4	Device has been removed from network - faulty	The device has been removed the switch network. This is seen for all empty slots in the frame. If this is not an empty slot, then a fault has occurred. If the device in question is a node, see 5.2, “Verifying an HPS Node” on page 216 or 5.3, “Verifying an SPS Node” on page 219. Otherwise, contact your local Hardware Support.
−5	Device has been removed from network by system administrator	The device was placed offline by the Systems Administrator with the `Efence` command. Run `Eunfence` against the device.
−6	Device has been removed from network - no AUTOJOIN	The device was removed and isolated from the switch network. The possible cause is that the node was fenced without the autojoin option. This happens when a node is rebooted or powered off, or the node faulted. You should first attempt to unfence the node. If the node fails to rejoin the switch network then see 5.15, “Isolating Adapter and Switch Errors” on page 237. If the problem persists, contact IBM Software Support.

Table 9 (Page 2 of 2). Device Status as Listed in out.top		
No.	**Comments**	**Explanation and Action**
−7	Device has been removed from network for not responding	The device was removed from the switch network. An attempt to contact the device was made, but there was no response. If the device in question is a node, then see the relevant verification section, either 5.2, "Verifying an HPS Node" on page 216 or 5.3, "Verifying an SPS Node" on page 219. Otherwise, contact IBM Software Support.
−8	Device has been removed from network because of a mis-wire	The device is not cabled properly. There are two possible causes for this condition. Either the switch network is mis-wired, or the the frame supervisor's TTY is not cabled properly. Examine the /var/adm/SPlogs/cable_miswire file. Verify or correct all the links listed in the file. Now issue `Eclock -d` and rerun `Estart`. If the problem persists, contact your Hardware Support.
−9	Destination not reachable	The device was not reachable through the switch network. This is generally due to other errors in the switch network fabric. Investigate the root of the problem; after correcting, rerun `rc.switch` on the primary node and then rerun `Estart`.

Table 10. Link Status as Listed in out.top		
No.	**Comments**	**Explanation and Action**
−2	Wrap plug is installed	This link is connected to a wrap plug. This is not normally a problem.
−4	Link has been removed from network or mis-wired - faulty	The link is not operational and was removed from the network. Possible causes are, either the link is mis-wired or the link has faulted. First check the /var/adm/SPlogs/css directory for the existence of a cable_miswire file. If the file exists, verify or correct all links listed in the file. Then execute `Eclock -d` and rerun `Estart`. If the cable_miswire file does not exist, check the /var/adm/SPlogs/css/flt file for entries relating to this link. If entries are found, verify that the cable is seated at both ends, then `rc.switch` the primary node and `Estart`. If the problem persists, contact IBM Software Support.
−6	Link has been removed from network - no AUTOJOIN	The device was removed and isolated from the switch network. The possible cause is that the node was fenced without the autojoin option. This happens when a node is rebooted or powered off, or the node faulted. First attempt to Eunfence the device. If the node fails to rejoin the switch network, see 5.15, "Isolating Adapter and Switch Errors" on page 237. If the problem persists, contact IBM Software Support.
−7	Link has been removed from network - fenced	The device was placed offline by the systems administrator with `Efence`. Run `Eunfence` against the associated node.
−8	Link has been removed from network - probable mis-wire	The link is not cabled properly. Review the file /var/adm/SPlogs/css/cable_miswire. Verify or correct all the links listed and then run `rc.switch` on the primary node followed by an `Estart`.
−9	Link has been removed from network - not connected	The link cannot be reached by the primary node, so initialization of the link is not possible. This is generally caused by other problems in the switch network, such as a switch chip being disabled. Investigate and correct the root of the problem; after correcting, rerun `rc.switch` on the primary node and then rerun `Estart`.

5.14 Ecommand Problems

Many of the Ecommands are global in nature (that is, they communicating with all nodes in the system or partition), and these commands can sometime fail because they are unable to communicate with every node. If you suspect this type of failure see *RS/6000 SP: Diagnosis and Messages Guide*, GC23-3899 or *RS/6000 SP: Problem Determination Guide*, SG24-4778 for details on how to resolve Kerberos problems.

Another area where Ecommands commonly fail is when accessing the SDR to perform their functions. These commands sometimes fail because they cannot access the SDR or the SDR is setup incorrectly. If you suspect this type of failure, see 5.5, "Verifying the System Data Repository (SDR)" on page 222 on how to eliminate this problem.

If the error is not isolated with the previous actions, view the error output returned from the command, specifically making a note of message number and text returned. Cross-reference the message number in the *RS/6000 SP: Diagnosis and Messages Guide*, GC23-3899 and execute the recommended corrective actions.

5.15 Isolating Adapter and Switch Errors

The isolation of adapter and switch errors starts the same way for both SPS and HPS. Begin by viewing the AIX error log. For switch-related errors, log on to the primary node. For adapter problems, log on to the suspect node. Once you are logged on, enter the following: as shown in Figure 72

```
-> errpt|more
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
D70626E9   0121120697 I H Worm           Switch primary node takeover
97D66668   0121120697 I S Worm           Links not initialized during Estart
694AEE61   0121120697 I S Worm           Nodes not initialized during Estart
```

Figure 72. Sample Error Report Output

There may be a situation on a node where the descriptions are not seen, but in their place is an EB*xx* code. This is because the error messages have not been added to the Error Record Template Repository, or have not been installed in the error log message catalog. Figure 73 shows an example:

```
9DBCFDEE   0205153697 T O errdemon       ERROR LOGGING TURNED ON
D70626E9   0121120697 I H Worm           EB32
97D66668   0121120697 I S Worm           EB36
```

Figure 73. An Example Where the css Messages are not Installed

To correct the problem issue the following commands on the node or nodes that are exhibiting this symptom:

```
-> errupdate -f /usr/lpp/ssp/ssp.css.err
```

This will update the Template Respository. Issue the following:

```
-> errinstall -f /usr/lpp/ssp/ssp.css.codepoint
```

to install Error description, Probable Cause, User Cause and so on into the message catalog. You should now see the current messages (and any messages posted in the future) correctly detailed.

The Resource Name in the error log should give you an indication of how the failure was detected.

Table 11. Resource Name Indication

Resource Name	**Indication**
Worm	The information was extracted from the switch and/or adapter hardware.
css	Incorrect status was detected by the css device driver.
css0	The css adapter failed diagnostics.

Table 12 on page 238 and Table 13 on page 240 show the information seen within the error log when examined in detail. They are specific to either HPS or SPS node types.

Table 12 (Page 1 of 2). Possible Causes for HPS Failures

Error Description	**Analysis**
Switch adapter failed POST diagnostics.	The switch adapter failed Power On Self Test diagnostics either due to a bad switch adapter or because of a missing external clock source. See 5.7, "Adapter Diagnostic Failures" on page 225.
Switch fault - bad packet checksum.	The local switch adapter detected a corrupt packet. This could be caused by a bad local or remote switch adapter or cable. Run switch adapter diagnostics on the failing node. If diagnostics fails to report any problems, then contact IBM Software Support.
Switch daemon terminated.	The fault_service_Worm_RTG daemon was terminated. The possible reasons for this are: • rc.switch was run on the node in which no action is required. • A SIGTERM signal was processed. Run `rc.switch` and then `Estart` on the node. • There is a bad switch adapter or missing external clock source. See 5.7, "Adapter Diagnostic Failures" on page 225. • There is a bad microchannel slot. Run complete diagnostics on the node. If this fails to isolate the problem, contact your local Hardware Support.
Switch fault - not isolated.	The switch was started at a user's request by running `Estart`. This is informational only; no action is required.
Switch fault - detected by switch chip.	A switch chip detected a fault condition either because of a bad switch chip or cable. Contact your local Hardware Support.
Switch fault - detected by adapter.	An adapter in the switch network caused a switch fault. This could be caused by a node that was on the switch being rebooted, powered off, or crashing. If this is the case, unfence the node once it becomes available again. If this is not the case then the fault may have been caused by a bad switch adapter or cable. Run switch adapter diagnostics on the failing node. If diagnostics fails to isolate the problem, contact your local Hardware Support.
Switch fault isolation failed - retries exhausted.	Switch recovery failed. This could be caused by: • The processor route table distribution failed, in which case contact IBM Software Support • There is a bad switch or adapter cable. See 5.13, "Device and Link Problems" on page 234. If the problem persists, contact your local Hardware Support.
Switch adapter - unexpected interrupt	An unexpected interrupt condition occurred on the local adapter, probably caused by a bad switch adapter. Run switch adapter diagnostics to isolate the problem, or contact your local Hardware Support.
Switch adapter failed online diagnostics.	Online switch adapter diagnostics failed. See 5.7, "Adapter Diagnostic Failures" on page 225.
Switch faulty - switch component.	A bad switch component was found in the network possibly, caused by a bad switch data cable or switch chip. Examine the /var/adm/SPlogs/css/daemon.stdout and /var/adm/SPlogs/css/out.top for additional information. Also refer to 5.8.5, "Cable Verification" on page 229 and 5.13, "Device and Link Problems" on page 234.
Switch adapter - bus error.	A bus error or channel check condition was detected on the local switch adapter. This may be caused by a bad switch adapter, bad microchannel slot, or a missing external clock source. Run complete diagnostics on the node. If this fails to isolate the problem, contact your local Hardware Support.

Table 12 (Page 2 of 2). Possible Causes for HPS Failures

Error Description	**Analysis**
Switch - cable mis-wired.	A switch cable mis-wire was detected because the switch network cabling does not match the switch topology file. Examine the /var/adm/SPlogs/css/cable_miswire file to determine which cables are in question. Then check, and if necessary, reconnect the associated cables. You may need to contact local Hardware Support to do this for you.
Switch adapter - error detected by adapter.	The local switch adapter detected an error and may be faulty. Run adapter diagnostics against the node. If problems persists contact your local Hardware Support.
Switch adapter -PIO error euring interrupt handling	An exception occurred accessing the switch adapter during interrupt handling, probably caused by a bad switch adapter. Run adapter diagnostics against the node. If problems persists contact your local Hardware Support.
Switch NI - nodes queued for fencing.	A node or nodes were queued to be fenced by the operator through the Efence command. This is purely an informational message.
Switch NI - nodes queued for unfencing.	A node or nodes were queued to be unfenced by the operator through the Eunfence command. This is purely an informational message.
Switch NI - node not on switch.	The node or nodes queued for fencing with the Efence command are not currently on the switch network. This is purely an informational message.
Switch NI - node not a fenced node.	The node or nodes queued for unfencing with the Eunfence command are already unfenced. This is purely an informational message.
Switch NI - node not found in topology.	The node or nodes queued for fencing were not found in the topology file, therefore an invalid node was specified with the Efence command. This is purely an informational message.
Switch NI - node already fenced.	The node specified with the Efence command is already fenced. This is purely an informational message.
Switch NI - reading TIME SR failed.	A software error has occurred during the operation of unfencing the specified node. Retry the operation. If it fails again, contact IBM Software Support.
Switch NI - reading TRCV SR failed.	A software error has occurred during the operation of unfencing the specified node. Retry the operation. If it fails again, contact IBM Software Support.
Switch NI - link tuning failed.	An error occurred tuning the node link during an unfence operation. This is probably caused by a bad cable. Refer to the node to switch cable verification in 5.8.5, "Cable Verification" on page 229.
Switch NI - device initialization failed.	The specified switch adapter failed to initialize during an unfence operation. This may mean that the adapter is faulty. Run switch adapter diagnostics to isolate the problem. If problems persists contact your local Hardware Support.
Switch NI - failed to distribute routes.	A software error was detected while attempting to distribute switch route information during an unfence operation; contact IBM Software Support.

Table 13. Possible causes for SPS Failures

Error Description	**Analysis**
A parity error on data was detected.	A switch chip detected a bad parity on a data packet due to either a bad switch, switch adapter or switch cable. Refer to 5.13, “Device and Link Problems” on page 234. If problems persists, contact your local Hardware Support.
Switch adapter transient error.	The switch adapter detected an error that may be software-correctable. If the node is still running (that is, if the fault_service_Worm_RTG_SP is still running), the software has corrected itself. If the Worm is not running you may have a faulty switch, switch adapter, or switch cable. Refer to 5.13, “Device and Link Problems” on page 234. If problems persist, contact your local Hardware Support.
An undefined control character was detected.	The switch chip received an undefined control character possibly due to a faulty switch, switch adapter, or switch cable. See 5.13, “Device and Link Problems” on page 234. If this procedure fails to isolate the problem contact your local Hardware Support.
An unexpected interrupt occurred.	An unexpected interrupt condition happened on the local switch adapter. This could be caused by a switch failure. Examine the file /var/adm/SPlogs/css/fs_daemon_print.file for information of a failure. If none is found, you may have a faulty switch adapter. Run switch adapter diagnostics on the failing node. If problems persists, contact your local Hardware Support.
An EDC error was detected.	A switch chip reported an EDC error, possibly caused by a bad switch, switch adapter, or switch cable. Refer to 5.13, “Device and Link Problems” on page 234. If problems persist, contact your local Hardware Support.
A recv port link synch failure occurred. A Sender Port Link Synch Failure occurred	A switch chip has lost clock synchronization on one of its send or receive ports. If the port is connected to a node, this may have been caused by the node being powered off or rebooted. If this is not the case, then you may have a faulty switch cable. See 5.8.5, “Cable Verification” on page 229.
Switch daemon process terminated.	The fault_service_Worm_RTG_SP daemon was terminated possibly due to a faulty microchannel slot, switch adapter or a missing external clock source. Run complete diagnostics to isolate the problem. If no errors are reported and the problems persist contact your local Hardware Support.
Switch adapter failed POST diagnostics	The switch adapter failed Power On Self Test diagnostics, which may be caused by a bad switch adapter or missing external clock source. Refer to 5.7, “Adapter Diagnostic Failures” on page 225.
Switch link outage.	The switch adapter-to-switch chip link is not operational, which could simply mean that the node in question is fenced. If this is not the case, check that it is not missing the external clock source. See 5.8.1, “HPS External Clock Verification” on page 227 and 5.8.2, “SPS External Clock Verification” on page 228. If problems persist, contact your local Hardware Support.
Bad packet received.	The switch adapter received an unexpected end-of-packet, or a packet failed Cyclic Redundancy Check (CRC). Unless this becomes a frequent message, no action is required. However, if the messages are numerous, contact your local Hardware Support.
Error threshold exceeded.	The switch adapter was reset because of excessive transient errors usually caused by a bad switch, switch adapter, or switch cable. Refer to 5.13, “Device and Link Problems” on page 234 to isolate the problem. If you cannot, contact local Hardware Support.
Off-line Request has been received.	This node has fenced from the switch network either by the user or as part of a recovery action. Simple unfence the node.

5.16 Diagnosing HPS Estart Problems

Use the following steps to diagnose Estart failures:

1. Log on to the primary node.
2. View the bottom of the /var/adm/SPlogs/css/fs_daemon_print.file
3. Use the messages seen to reference Table 14.

Table 14. HPS Estart Problems

Message	**Analysis**
Attempt to get the node list for this partition failed.	Unable to obtain a node list for the partition. Check that the SDR is set up properly. Refer to 5.5, "Verifying the System Data Repository (SDR)" on page 222.
Attempt to change switch_responds to 0, failed.	Unable to turn switch_responds off in the SDR. Check that the SDR is set up properly. Refer to 5.5, "Verifying the System Data Repository (SDR)" on page 222.
adapter_stop() failed.	Unable to shutdown the switch adapter. This may be caused by a faulty switch adapter. Run switch adapter diagnostics. If problems persist, contact your local Hardware Support.
worm_init_tune() failed.	Unable to initialize the switch either due to a corrupt topology file (in which case refer to 5.4, "Verifying Switch Topology Configuration" on page 221), or caused by a faulty switch, switch adapter, or switch cables, (if this is the case, contact your local Hardware Support).
bring_node_up() failed.	Unable to bring the node online because routes failed to load onto the adapter, or the adapter failed to start, or the adapter failed to become ready. Run switch adapter diagnostics on the failing node. If this fails to isolate the problem, contact your local Hardware Support.
build_node_list() failed.	Errors occurred building /tmp/node_list. Check that there is space in /tmp. If there is, then you may have a missing or corrupt topology file. See 5.4, "Verifying Switch Topology Configuration" on page 221.
link(out.top, act.top.PID) failed, errno = *x*.	Unable to write or access /var/adm/SPlogss/css/act.top. Check the permissions of the directory and that other files exist. If no progress is made, note the errno returned and contact IBM Software Support.

If the message found at the bottom of /var/adm/SPlogss/css/fs_daemon_print.file was not found in Table 14 or the problem could not be resolved, contact IBM Software Support.

5.17 Diagnosing SPS Estart Problems

Use the following steps to diagnose Estart failures:

1. Log on to the primary node.
2. View the bottom of the /var/adm/SPlogs/css/fs_daemon_print.file.
3. Use the messages seen to reference Table 15.

Table 15. SPS Estart Problems

Message	**Analysis**
Error in buildDeviceDatabase().	Unable to build the device database due to a missing or corrupt topology file or a malloc() system call failure. Check the topology file with 5.4, "Verifying Switch Topology Configuration" on page 221. If this does not identify the problem, contact IBM Software support.
Error in TBSswitchInit().	Unable to initialize the switch network. Refer to 5.18, "SPS Worm Errors" on page 243.
Error in writeDeviceDatabase().	Unable to write /var/adm/SPlogs/css/out.top either because there is no space left in the /var file system or due to a corrupt or missing topology file. See 5.4, "Verifying Switch Topology Configuration" on page 221.
No valid backup - SDR current backup being changed to none.	There is no node available as a backup node: this is purely an informational message.
Can not access SDR - SDR current backup not changed.	An SDR failure has occurred. Check that it is set up correctly. See 5.5, "Verifying the System Data Repository (SDR)" on page 222.

Error in
fopen(act.top.PID)
fprintf(act.top.PID)
fclose(act.top.PID)
rename(act.top, act.top.PID).
	An error occurred while accessing /var/adm/SPlogs/css/act.top.<PID>. Check the permissions of the directory and that other files exist. If no progress is made note the errno returned and contact IBM Software Support.

If the message found at the bottom of `/var/adm/SPlogss/css/fs_daemon_print.file` was not found in Table 15 or the problem could not be resolved, contact IBM Software Support.

5.18 SPS Worm Errors

The following steps are used to diagnose Worm initialization failures:

1. Log on to the primary node.
2. View the bottom of the /var/adm/SPlogs/css/worm.trace file. There should be a message similar to one of the following, where *xx* is any value:

   ```
   TBSworm_bfs_phase1() failed with rc = xx
   ```

   or

   ```
   TBSworm_bfs_phase2() failed with rc = xx
   ```

3. Use the value of *xx* returned on either of these messages to index the following table. If the return code you see is not listed then contact IBM Software support.

Table 16. SPS Worm Return Codes

Return Code	**Analysis**
−3	The local adapter receiver port is not enabled. • Check that the oncoming primary is not fenced. See 5.11, "Eunfencing the Oncoming Primary Node" on page 232. • The switch is not clocked. Issue `Eclock -d` from the control workstation
−4	Unable to generate routes for the network, probably due to a corrupt topology file. See 5.4, "Verifying Switch Topology Configuration" on page 221.
−5	The send packet from the local node failed. Check the switch adapter by running switch adapter diagnostics on the primary node. If this fails to isolate the problem, contact IBM Software support.
−6	A switch mis-wire was detected because the switch network cabling does not match the topology file. Review the /var/adm/SPlogs/css/cable_miswire file to determine which cables are in question and check or reconnect the associated cables. If problems persist, contact your local Hardware Support.
−7	A node mis-wire was detected because the switch network cabling does not match the topology file. There are two possible causes for this condition: the switch network is mis-wired or the frame′s supervisor TTY is not cabled properly. First review /var/adm/SPlogs/css/cable_miswire file to determine if any cables are in question and check or reconnect if there are. Otherwise, issue `Eclock -d` and rerun `Estart`. If problems persist, contact your local Hardware Support.
−8	The receive FIFO queue is full, probably due to a faulty switch adapter. Run switch adapter diagnostics to identify the problem. If nothing is found contact, your local Hardware Support.
−9	Unable to initialize FIFOs. Check the switch adapter by running switch adapter diagnostics to identify the problem. If nothing is found contact, your local Hardware Support.
−27	The TBIC was not initialized. This could simply be because the switch adapter is uninitialized. Run `rc.switch` on the primary node followed by an `Estart`. If this has no effect, check the switch adapter by running switch adapter diagnostics on the primary node. If problems persist, contact your local Hardware Support.
−36	This node resigned as the primary node because it could no longer control and monitor the switch. This is an informational message informing you that the backup has taken over control.
−43	A read or write operation to the switch failed, probably due to a faulty switch adapter. Check the adapter by running switch adapter diagnostics on the primary node. If nothing was found, contact your local Hardware Support.

5.19 Example Test Case

We will show how to use the previous tables through the following case.

In this scenario, we have an SPS system where the switch network is down and we need to restart. We issue an Estart from the CWS and see the following as shown in Figure 74.

```
-> Estart
 Estart: Oncoming primary != primary, Estart directed to oncoming primary
 Estart: Oncoming primary backup node sp21n07.msc.itso.ibm.com is fenced.
         Switch initialization will select an available backup.
         After switch initialization is complete, use Eprimary to
         see which node was selected as the primary backup
Switch initialization started on sp21n05.
Estart_sw: 0028-071 Switch initialization failed.
The fault_service_Worm_RTG_SP daemon exited due to errors.
See /var/adm/SPlogs/css/fs_daemon_print.file and
/var/adm/SPlogs/css/worm.trace for more information.
```

Figure 74. Initial Errors from Estart

As we can see, we have an Estart problem. The recommendation from the error is to review /var/adm/SPlogs/css/fs_daemon_print.file and /var/adm/SPlogs/css/worm.trace.

Also, Table 6 on page 214 tells us to see 5.17, "Diagnosing SPS Estart Problems" on page 242. The steps in 5.17, "Diagnosing SPS Estart Problems" on page 242 nopage. also tell us to examine the fs_daemon_print.file on the primary node. See Figure 75.

```
print_the_time: Time = Mon Feb 17 17:28:03 1997
fs_daemon_fsm_main: got request, type = startup msg
init_on_startup_msg: 2510-318 Error in TBSswitchInit()
fs_daemon_fsm_main: 2510-394  Error in init_on_startup_msg()
main: 2510-385 fs_daemon_fsm_main() returned -1
send_receive_deinit: Entry
send_receive_deinit: Returning 0
fs_daemon_exit: Enodes does not exist, so we assume SDR is in use
fs_daemon_exit: Turning off this nodes switchResponds bits in the SDR
```

Figure 75. Errors in the fs_daemon_print.file

We can see there is a *TBSswitchInit* error. Cross-referencing this error in Table 15 on page 242, we are referred to 5.18, "SPS Worm Errors" on page 243. 5.18, "SPS Worm Errors" on page 243 tells us to examine the /var/adm/SPlogs/css/worm.trace file, again as recommended by the initial error. Figure 76 on page 245 shows the output from the file:

```
Entering Entering Entering print_the_time_worm: The date and time  = Mon Feb 17

17:28:03 1997
TBSswitchInit: Switch network Initialization Started!
TBSswitchInit: backup_id = 6
TBSworm_bfs_phase1: Switch Phase1 network Initialization Started!
syncFifoPh1: Cleaning up the Receive FIFO
TBSworm_bfs_phase1: Device ID = 100015
TBSswitchInit: 2510-711 TBSworm_bfs_phase1() failed with rc=-3
```

Figure 76. The Return Code from worm.trace

By cross-referencing Table 16 on page 243, the suggestions are that the Oncoming primary is fenced or the switch is not clocked. We cannot check that the node is not fenced with the Efence command because the switch has not been started. We see the message shown in Figure 77.

```
-> Efence
Efence:  0028-036  No functional switch primary node, run Estart.
```

Figure 77. Trying to Use Efence on a Switch That Has Not Been Started

The only way we can see which nodes are fenced is through the SDR. To view this information we use the following, as shown in Figure 78.

```
-> SDRGetObjects switch_responds
node_number  switch_responds autojoin     isolated     adapter_config_status
          1           0           0            1 css_ready
          5           0           0            0 css_ready
          6           0           0            1 css_ready
          7           0           0            1 css_ready
          8           0           0            1 css_ready
```

Figure 78. Viewing the switch_responds Object Class

The only node listed as not being fenced, as seen in the *isolated* stanza is our current primary, node 5. Therefore, the possibility of the Oncoming Primary being fenced is not the cause of this failure. The other option mentioned was that the switch was not clocked. To correct this, we issue Eclock -d from the CWS and rerun Estart. This has no effect, and we receive exactly the same error message and outputs as before.

We appear to have run out of options and need to contact Support; however, there is one other test we can perform first. We can try to move the primary to another node to see if the same symptoms are exhibited, which would infer a system-wide problem, as opposed to a problem attached to one node.

To attempt to start the switch on a new primary at this stage, we need to perform two actions. First we must issue the Eprimary command. It is a good idea to select a node on a different switch chip. We will select node 7. See Figure 79 on page 246.

```
-> Eprimary 7
Eprimary:  0028-040 Warning:  Oncoming primary backup node is fenced.
Eprimary: Eprimary will continue. sp21n08.msc.itso.ibm.com
Eprimary: Defaulting oncoming primary backup node to sp21n08.msc.itso.ibm.com
-> Eprimary

none    - primary
7       - oncoming primary
none    - primary backup
8       - oncoming primary backup
```

Figure 79. Setting the New Primary

As shown previously, the switch will not start if an Oncoming Primary node is fenced, and from the SDR output we have already seen that this node is fenced. Therefore our second task is to unfence the node. This is not possible on a switch that is not started, so we have to change the SDR with the SDRChangeAttrValues command. See Figure 80.

Note

This is a "last resort" action and should never be performed if there is another way, through command-level actions, to perform the same function.

Another point to be aware of is the usage of = and ==, which is demonstrated in Figure 80. We recommend that you make a copy of the object class before altering for recovery actions. This is a flat file located in /spdata/sys1/sdr/partitions/<IP address>/classes with the filename of switch_responds. Simply use the AIX cp command to copy it to an alternative name, or you may run the SDRArchive command.

```
-> SDRChangeAttrValues switch_responds node_number==7 isolated=0

-> SDRGetObjects switch_responds node_number==7
node_number  switch_responds autojoin     isolated     adapter_config_status
           1           0           0            0 css_ready
```

Figure 80. Changing the Node to Non-Isolated

Now we can issue another Estart to start the switch. See Figure 81.

```
-> Estart
 Estart: Oncoming primary != primary, Estart directed to oncoming primary
 Estart: Oncoming primary backup node sp21n08.msc.itso.ibm.com is fenced.
         Switch initialization will select an available backup.
         After switch initialization is complete, use Eprimary to
         see which node was selected as the primary backup
Switch initialization started on sp21n07.
Initialized 1 node(s).
Switch initialization completed.
Estart_sw: 0028-044 Unable to find suitable primary backup node. No primary back
up node running.
```

Figure 81. Latest Estart Output

As we can see, some progress has been made. The switch has started on the new primary node, node 7. This would indicate a possible problem with our previous primary node. The first stage, however, is to unfence the other nodes so that they communicate over the switch, using Eunfence. This fails for nodes 1, 5 and 6. Currently node 6 is powered down so this result is expected, but nodes 1 and 5 are operational. Figure 82 shows the error seen:

```
-> Eunfence sp21n01
Unable to unfence the following nodes:
sp21n01.msc.itso.ibm.com Timeout
-> Eunfence sp21n05
Unable to unfence the following nodes:
sp21n05.msc.itso.ibm.com Timeout
```

Figure 82. Eunfence Fails for Two Nodes

Refer back to Table 6 on page 214, which refers us to 5.13, “Device and Link Problems” on page 234. In 5.13, “Device and Link Problems” on page 234 we are told to examine the /var/adm/SPlogs/css/out.top file on the primary node for information on the state of the devices and links. The entry on the primary node looks similar to Figure 83.

```
# Node connections in frame L01 to switch 1 in L01
s 15 3  tb0 0 0           E01-S17-BH-J7 to E01-N1   -6 R: device has been remov
ed from network - no AUTOJOIN (link has been removed from network - no AUTOJOIN)
s 15 2  tb0 1 0           E01-S17-BH-J8 to Exx-Nxx   -4 R: device has been remo
ved from network - faulty (link has been removed from network - fenced)
s 16 0  tb0 2 0           E01-S17-BH-J26 to Exx-Nxx   -4 R: device has been rem
oved from network - faulty (link has been removed from network - fenced)
s 16 1  tb0 3 0           E01-S17-BH-J25 to Exx-Nxx   -4 R: device has been rem
oved from network - faulty (link has been removed from network - fenced)
s 15 1  tb0 4 0           E01-S17-BH-J9 to E01-N5   -6 R: device has been remov
ed from network - no AUTOJOIN (link has been removed from network - no AUTOJOIN)
s 15 0  tb0 5 0           E01-S17-BH-J10 to E01-N6   -6 R: device has been remo
ved from network - no AUTOJOIN (link has been removed from network - no AUTOJOIN
)
```

Figure 83. Output from out.top file on the Primary

From this output we cross-reference Table 9 on page 234 and Table 10 on page 235 and find that, as the message says, both the device and links for nodes 1 and 5 have been isolated from the network. The suggestion is to attempt to unfence, which we know fails, and to refer to 5.15, “Isolating Adapter and Switch Errors” on page 237. On looking at 5.15, “Isolating Adapter and Switch Errors” on page 237, we are told to examine the error log from the primary node using the errpt command. Figure 84 on page 248 shows the output:

```
-> errpt | more
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
97D66668   0217194197 I S Worm           Links not initialized during Est
97D66668   0217194197 I S Worm           Links not initialized during Est
694AEE61   0217194197 I S Worm           Nodes not initialized during Est
694AEE61   0217194197 I S Worm           Nodes not initialized during Est
89F34F83   0217194197 P S Worm           Switch Fault Service Daemon Term

-> errpt -a | more
-------------------------------------------------------------------------
LABEL:          SP_SW_UNINIT_LINK_R
IDENTIFIER:     97D66668

Date/Time:       Mon Feb 17 18:53:33
Sequence Number: 462
Machine Id:      000168785700
Node Id:         sp21n07
Class:           S
Type:            INFO
Resource Name:   Worm

Description
Links not initialized during Estart

Probable Causes
Switch cable mis-wired
Switch cable or switch failure

User Causes
Switch cable disconnected
Switch cable mis-wired
Switch cable faulty or unseated

        Recommended Actions
        Check / reconnect cables
        See /var/adm/SPlogs/css/cable_miswire for possible mis-wire.

Detail Data
Software ID String
LPP=PSSP,Fn=fsd_fsm.c,SID=1.56.2.1,L#=3207,

Chip and Port Number(s)
      100015          1      100015          2      100015          3
-------------------------------------------------------------------------
```

Figure 84. Error Log Output on the Primary Node

It should be noted from the output that all three nodes are on the same switch chip, so we may well have seen exactly the same problem when changing the primary if we had used a node on the same switch chip. The error shown in the previous error log example is not listed in the tables. However, there are some actions suggested in the error log that we can perform. On checking the cabling setup it was found that nodes 5 and the powered-off node, node 6, had their cables swapped. There was no /var/adm/SPlogs/css/cable_miswire file, since this file is created only in some specific cabling problems that involve the clock signal. In the HPS, it is more common to see this file when a miswire occurs. The SP Switch uses the AIX error log as the main logging mechanism for cabling problems.

The final stage to get the nodes back onto the switch, is to correct the cabling mis-wire, execute the `/usr/lpp/ssp/css/rc.switch` script on the two nodes, run an `Estart` on the CWS, and then finally unfence them.

5.20 Fault File (flt) Enhancement for SP Switch

This section describes an enhancement to the switch logging system that will be available in the form of a fix. At the time of writing, this enhancement is still being tested and is not currently available. However, by the time of publication, this fix should be available to order.

5.20.1 Enhancement Requirements

The design requirements for the fault file enhancements are as follows:

- Log more information about the TBS switch failures and recovery actions in the fault (flt) file in a concise format. The new concise format will make it simpler to understand what failures have occurred and what recovery action has been taken by the fault service daemon. The new format will support parsing by fault analysis tools, and be readable by the customer or trained personnel.
- The messages logged in the fault files will be explained in documentation available to the customer, and should only require limited understanding of the internal design and design terminology.

5.20.2 Internal Code Changes

The code changes for this design are small. They consist of insertions of the FSD_DEBUG_FLTx macro into key sections of the code where logging should be added, and changing the function name to the time and date for each entry. The following types of messages will be logged in the flt file:

- Changing the node′s personality.
- List of switch chips disabled and reasons.
- List of nodes disabled and reasons.
- Switch error status bits found during switch initialization.
- Failures in Service packet broadcasts.
- Executing switch initialization as a recovery action or as a command.
- Starting Primary node takeover.
- Completing Primary node takeover.
- Disabling port during fence.
- Enabling port during unfence.
- List of nodes fenced and unfenced successfully.
- Timeout waiting for Error/Status from the switch.
- Executing Estart because switch routes need to be updated.
- Switch scan failures.
- Failure changing node personality to Secondary.
- Estart failure because Primary′s link to switch is disabled.
- The second phase of switch initialization being retried.
- The number of second phase switch initialization retries has reached it′s limit.
- Failures generating service routes.

- Catching signals sent to fault service daemon (SIGBUS, SIGTERM or SIGDANGER).

5.20.3 The flt File Messages

The fault file (flt) contains a history of actions taken by the fault service daemon. These actions are a result of commands issued by the system administrator, or errors in the switch network that required a recovery action. This information is useful for problem determination or to recount past history switch operations.

5.20.3.1 New flt File Messages

Following are some examples of the new messages that get logged in the fault file (flt).

EMSG812 ″%1$s: 2510-812 Backup node starting Primary node take-over.″

Insert: time.

Explanation: The Primary Backup node has timed-out waiting for the Primary node to contact it. The Primary Backup is starting Primary node take-over.

EMSG818 ″%1$s: 2510-818 Switch Scan failed with a return code of %2$d. Estart will be executed.″

Insert: time, switch scan return code.

Explanation: The Primary fault service daemon periodically scans the switch network for errors. The scan detected a problem and as a result Estart will be executed as a recovery action.

EMSG829 ″%1$s: 2510-829 Device ID %2$d port %3$d has been disabled.″

Insert: program name, device ID.

Explanation: The port on the device specified has been disabled because of a error reported by the link. This link will not be used.

5.20.3.2 Changed flt File Messages

Following are examples of messages that previously existed in the flt file and have now been updated.

EMSG196 ″%1$s: 2510-196 The fault service daemon got a SIGDANGER signal, probably because the system is starting to get low on paging space.″

Insert: time.

Explanation: The fault service daemon was sent a SIGDANGER signal because the system is starting to get low on paging space.

INFO760 ″%1$s: No route to destination %2$d could be generated.″

Insert: time, device ID.

Explanation: No switch route could be generated to the specified destination. The most probable cause is that the destination is isolated from the switch Primary node due to disabled or faulty switch chips.

EMSG743 "%1$s: 2510-743 Disabling port %2$d (jack %3$d) of chip %4$d on the switch in slot %5$d of frame %6$d"

Insert: time, port number, jack number, chip ID, slot, frame number.

Explanation: Switch recovery code disabling a port on the switch as a recovery action.

5.20.3.3 Testing and flt File Samples

Following are samples of the fault file (flt) output for several events. Testing of this design is performed by running Ecommands and then looking in /var/adm/SPlogs/flt for the correct output. Nodes are identified in the flt file by their switch node number.

```
09/11/96 13:07:21 : 2510-744 Time of Estart event is: Fri Oct 11 13:07:21 1996
09/11/96 13:07:37 : The date and time  = Fri Oct 11 13:07:37 1996
09/11/96 13:07:37 : Switch and Adapter Error bits found during switch initializa
tion.
09/11/96 13:07:37 : Device ID = 100016
09/11/96 13:07:37 : 2510-793 First  Error Capture Register  = 000008.
09/11/96 13:07:37 : 2510-741 Second Error Capture Registers = 20002000 000000a0
00000000 00000000 000000a0 000000
09/11/96 13:07:37 : The Primary backup is switch node number = 10
09/11/96 13:07:37 : Switch initialization completed successfully!
```

Figure 85. An Estart with Error Bits Found

To test, execute `Eclock`, then `Estart`. The output in the /var/adm/SPlogs/css/flt on the Primary node should look similar to Figure 85. The values in the First and Second Error Capture Registers will be different and there will be non-zero values values for each switch chip in the partition.

```
09/11/96 13:08:34 : 2510-744 Time of Estart event is: Fri Oct 11 13:08:34 1996
09/11/96 13:08:34 : The Primary backup is switch node number = 10
09/11/96 13:08:34 : Switch initialization completed successfully!
```

Figure 86. An Estart with No Error Bits Found

To test, execute `Estart` at least twice. The output of the last `Estart` in the flt file on the Primary node will look similar to Figure 86. The Primary backup node listed should agree with the one listed by running Eprimary. The Eprimary output will identify the node number, and the flt file will identify the node by switch node number.

```
09/11/96 13:15:09 : The date and time  = Fri Oct 11 13:15:09 1996
09/11/96 13:15:09 : 2510-814 Port disable for fence of node 7 completed.
09/11/96 13:15:11 : Node 7 Fenced.
```

Figure 87. An Efence of Node 8 (Switch Node Number 7)

To test, `Efence` a node that is not currently fenced. The output in the flt file on the Primary node should look similar to Figure 87. The flt file will identify the node being fenced by switch node number.

```
09/22/96 11:39:46 : 2510-812 Backup node starting Primary node take-over.
09/22/96 11:39:46 : The Primary backup is switch node number = 10
____________________________________________________________________________
09/22/96 11:39:46 : 2510-744 Time of Estart event is: Tue Oct 22 11:39:46 1996
09/22/96 11:39:47 : 2510-811 Fault service daemon′s personality has been changed
to Primary.
09/22/96 11:40:03 : The date and time  = Tue Oct 22 11:40:03 1996
09/22/96 11:40:03 : Switch and Adapter Error bits found during switch initializa
tion.
09/22/96 11:40:03 : Device ID = 100016
09/22/96 11:40:03 : 2510-793 First  Error Capture Register  = 000008.
09/22/96 11:40:03 : 2510-741 Second Error Capture Registers = 00000000 00000080
00000000
00000000 00000080 000000
09/22/96 11:40:03 : 2510-826 Device ID 2 un-initialized during switch initializa
tion. Disabling
device.
09/22/96 11:40:06 : The Primary backup is switch node number = 10
09/22/96 11:40:07 : Switch initialization completed successfully!
09/22/96 11:40:07 : 2510-813 Primary backup node completed Primary node take-over.
```

Figure 88. A Primary Backup Takeover

To test, use the kill command to send a SIGKILL (kill -9 process_id) to the fault service daemon process on the Primary node, then wait at least six minutes for the Primary backup node to take over. The Eprimary command is used to show when the Primary backup has taken over. The flt file on the backup node that took over should look similar to Figure 88.

```
09/16/96 13:13:23 : 2510-195 daemon got a SIGTERM signal
09/16/96 13:13:23 : 2510-823 Fault service daemon process has exited.
```

Figure 89. Fault Service Daemon Handling SIGTERM

To test, execute /usr/lpp/ssp/css/rc.switch on one of the nodes in the partition, then check the /var/adm/SPlogs/css/flt file for the output below. This test should be run on a secondary node. Eunfence can be used after running rc.switch to bring the node back onto the switch.

```
09/16/96 13:13:23 : 2510-197 daemon got a SIGBUS signal.
09/16/96 13:13:23 : 2510-823 Fault service daemon process has exited.
```

Figure 90. Fault Service Daemon Handling SIGBUS

To test, use the kill command to send a SIGBUS (kill -10 process_id) to the fault service daemon on a secondary node, then check the flt file for the following output. The Eunfence command is used to bring the node back onto the switch after the test is finished.

5.21 Partitioning Problems

In this section we assume the following. You decided to partition your system and decided which nodes should reside in which partition to best suit your needs. You then fully completed the worksheets in the *RS/6000 SP: System Planning Guide*, GC23-3902, and selected the appropriate topology file. Finally you start to perform the steps required to partition the system through the System Management Interface Tool (SMIT) panels as follows:

- Archive System Data Repository.
- Select **system partition configuration**.
- Display information for given configuration or Layout.
- Select **system partition layout**.
- Enter customization information for a selected system partition.
- Apply system partition configuration (verify).

Everything looked correct and ran smoothly, so you attempted to apply the partitioning setup and it failed.

In this section we give you an example of how to correct a failed system partitioning attempt. It is most important to get the system to a working state, because it will currently be in an intermediate state.

5.21.1 Basic Checkpoints

There are a few important rules that need to be adhered to prior to applying system partitioning. If these rules are not followed, you may well receive fatal or unstable results from attempted partitioning.

1. An IP alias must be set up for each new partition that is being created. This allows you to associate the new partition names with the same adapter that is currently being used to talk to the nodes in the existing partition. You need to use the *alias* flag of the `ifconfig` command to attach another IP address to the Control Workstation′s (CWS) current interface. Figure 91 shows where you first determine the hostname of the CWS, then the interface it is associated with, and finally the aliasing of a new, so far unused, IP address to the interface:

```
-> hostname
sp2en0
-> netstat -i | grep sp2en0
en0   1500  192.168.4   sp21en0.msc.its 16146746     0 15672219     0     0
-> ifconfig en0 alias 192.168.3.38 netmask 255.255.255.0
-> netstat -i
Name  Mtu   Network     Address           Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16896 <Link>                      6242730     0  6267879     0     0
lo0   16896 127         loopback.msc.it  6242730     0  6267879     0     0
en0   1500  <Link>2.60.8c.2d.7.ec       16146746     0 15672219     0     0
en0   1500  192.168.4   sp21en0.msc.its 16146746     0 15672219     0     0
en0   1500  192.168.4   sp21en1.msc.its 16146746     0 15672219     0     0
tr0   1492  <Link>10.0.5a.c9.67.51       9934713     0  7743030  2032     0
tr0   1492  9.12.1      sp21cw0.msc.its  9934713     0  7743030  2032     0
```

Figure 91. Setting Up a New Alias

The new IP aliases need to be resolved. Add an entry to /etc/hosts, or the Domain Name Server (DNS), or whatever method you use for name resolution. It is also a good idea to add this `ifconfig` command to the /etc/rc.net file to make the aliases permanent.

2. All affected nodes must be shut down.
   What do we mean by affected nodes? Basically, these are any nodes that are moving from the existing partition or partitions. This includes nodes that are in what is commonly known as a single-partition system. This is actually a single partition system and therefore nodes are moving to a new partition.

   For example, suppose we have an entire SP system, frame and CWS, running AIX 4.1 and PSSP 2.2, with a full 16 thin node frame all in a single partition. We then wish to partition the system with half the nodes in one partition and half in the other partition, with nodes 9 to 16 residing in the new partition. We will have to power down all these nodes before applying our selected partition setup.

3. Always run a verification before applying the new partition layout and carefully review the output even if the command return status is OK. As we will see in a later example, it can save a great deal of corrective procedures if errors are identified now instead of when actually applying.

Since these are all pre-requisite procedures and you have already encountered a problem, they are of little help now, but are definitely worth bearing in mind in the future.

5.21.2 Cleaning Up after a Failed Apply

Once the attempt to apply a new system partitioning setup has failed, where do our options lie?

1. Our first and most obvious choice would be to back out of the entire partitioning process by selecting to restore the SDR through SMIT. This will call the command `/usr/lpp/ssp/bin/sprestore_config` to retrieve the archived SDR from /spdata/sys1/sdr/archive/<filename>, which is simply a `tar` archive file of all SDR files. It then uses `tar` to extract and overwrite the current existing files. This, of course, eradicates all entries related to the change to the partition. Finally, the script attempts to clean up the partition-sensitive subsystems with `syspar_ctrl`.

   Important

   Do not restore the archived SDR if the affected nodes have been rebooted, especially if there are mixed partitions of AIX or PSSP code levels. This scenario is entirely unsupported and almost definitely will not work. If you find that you are in the situation where the affected nodes have been rebooted with a failed apply, contact IBM Software Support if you require help in remedying the situation.

2. Our second option is to understand why the apply failed and remedy the situation, then re-apply the new partition setup.

   The most effective way to debug an error at this stage is to carefully examine the SMIT screen output for error messages or unexpected output. In some cases the error message is apparent and we can pursue corrective procedures immediately. However, this will not always be the case, as we see in the example shown in Figure 92 on page 256. This will require us to find the failing command.

```
spapply_config:  Changed system partitions:

/usr/lpp/ssp/bin/SDRGetObjects: 0025-004 Item specified for query, insertion or
deletion was not found.
.
.
.
0 Nodes in 2 syspars processed.

nodes  VSD_nodes in_error  GVGs in_error  VSDs   HSDs in_error  syspar
-----  --------- --------  ---- --------  ----   ---- --------  ------
    0        0        0       0      0       0      0      0    _TOTALS_

Number of Renames due to name duplication:

 GVGs   VSDs   HSDs   syspar
 ----   ----   ----   -------
    0      0      0   _TOTALS_

0 errors and  duplicate name renames.

rc = 0 (-force flag is NOT set).


spapply_config:  Deleting hb, hr and emon subsystems for changed system part
itions...
0513-044 The stop of the hb.sp2en0 Subsystem was completed successfully.
waiting for 19038 (heartbeat) to exit
19038 has exited
removing SRC object "hb.sp2en0"
0513-083 Subsystem has been Deleted.
0513-044 The stop of the hr.sp2en0 Subsystem was completed successfully.
waiting for 18826 to exit
18826 has exited
removing SRC object "hr.sp2en0"
0513-083 Subsystem has been Deleted.


spapply_config:  Creating necessary new system partitions and new hb, hr
and emon subsystems...
0513-071 The sdr.sp2en1 Subsystem has been added.
0513-059 The sdr.sp2en1 Subsystem has been started. Subsystem PID is 18844.
spapply_config:  0022-160 Error updating Syspar_map for syspar: sp2en1,
switch-node-number: 0; rc= 4 .
```

Figure 92. Sample Error and Erroneous Output

Here we can see that an SDRGetObjects command has been issued to obtain some information from the SDR. This has failed, and further on in the output we see that no nodes are listed as being in either of the two system partitions. This is unexpected, and an error is returned from the spapply_config script.

It is often the case that if there is more than one error in the output, the first, and possibly non-fatal error, is the cause of subsequent error messages and should therefore be corrected first. In Figure 92 we see that the first problem encountered was an error with SDRGetObjects, later followed by an error from the spapply script. Our start point will therefore be to deal with the first error encountered, the SDRGetObjects error. We will follow problem determination procedures for this error.

So we are back to the question that has been asked many times throughout this book: where do we start?

The start point with partitioning failures, as we have already established, is to examine the error. If this is obscure and gives us no clues as to where the problem lies, we need to find, as in most cases, the exact failing command. To do this we can look at the script being run when partitioning is initiated. The full path name for this script is /usr/lpp/ssp/bin/spapply_config.

We already know that it was SDRGetObjects that failed, but unfortunately there are a number of these requests issued. By using the messages output just before the failure, we can further identify the failing command. In the current running example Figure 93 shows the relevant part of the spapply_config script:

```
      system "$SPMSG spapply_config $MSGCAT INFO65 '\n\nMsg INFO65 not found.
spapply_config:  Changed system partitions:\n'";
      foreach $syspar (@aff_syspar) {       # Go thru affected list and
          print "\n  $syspar";              #   print each syspar
      }                                     # End foreach $syspar
      print"\n\n";                            # Make it look good...
  }                                         # End unless
#
# See if each node in each affected syspar is down; if not issue        12166
# warning/attention  message.
#
# First build array of used node numbers and their current syspars...
#
  chop(@temp = $BINDIR/SDRGetObjects -x Syspar_map used==1 syspar_name node_num
ber);
```

Figure 93. Finding the Command in the Partition Script

From this output we see that the SDRGetObjects query is being run against the Syspar_map object class of the SDR, with the variable $SDRSyspar_fetch being a list of the attributes required from that object class.

Our first step then is to check that we can manually obtain information from this object class by running SDRGetObjects from the command line, as shown in Figure 94.

```
-> SDRGetObjects -x Syspar_map used==1
SDRGetObjects: 0025-004 Item specified for query, insertion or deletion was not
found.
```

Figure 94. Manually Running the SDR Query Command

As you see, the query failed. On checking for the existence of the file in /spdata/sys1/sdr/defs for the definition file, we found that it did exist. However, when checking for the existence of the actual data in /spdata/sys1/sdr/system/classes, we found that the file was indeed missing.

The Syspar_map object class should be created early on during the installation process when the /usr/lpp/ssp/bin/spframe command is run. We could recreate the object class by manually adding the entry back into the SDR using the specific SDR commands, for example:

```
-> SDRCreateObjects Syspar_map syspar_name=sp2en0 syspar_addr=192.168.3.37 \
node_number=1 switch_node_number=0 used=1
```

This would have to be done for every node that existed, a rather long way to go to recreate the object class. The other option we have available is to rerun the spframe command to create the entries for us. This is probably the best course of action:

```
-> spframe -r yes 1 1 /dev/tty0
```

After running the spframe command, the Syspar_map now exists and contain entries for all of the nodes that existed in the frame, as shown in Figure 95.

```
-> SDRGetObjects Syspar_map used==1
syspar_name  syspar_addr  node_number  switch_node_number used
sp2en0       192.168.3.37           1            0           1
sp2en0       192.168.3.37           2            1           1
sp2en0       192.168.3.37           3            2           1
sp2en0       192.168.3.37           4            3           1
sp2en0       192.168.3.37           5            4           1
sp2en0       192.168.3.37           6            5           1
sp2en0       192.168.3.37           7            6           1
sp2en0       192.168.3.37           8            7           1
sp2en0       192.168.3.37           9            8           1
sp2en0       192.168.3.37          10            9           1
sp2en0       192.168.3.37          11           10           1
sp2en0       192.168.3.37          12           11           1
sp2en0       192.168.3.37          13           12           1
sp2en0       192.168.3.37          14           13           1
sp2en0       192.168.3.37          15           14           1
sp2en0       192.168.3.37          16           15           1
```

Figure 95. Output from the Newly Populated Syspar_map Object Class

The final part of cleaning up after a failed installation before retrying is to remove the phantom partition structure in the SDR. This resides under /spdata/sys1/sdr/partitions and has the directory name equivalent to the *aliased* IP address that was added onto the en0 interface.

Now we are ready to retry the apply of the partition.

We see that this retry progressed further than the previous occasion by reviewing the SMIT output, but unfortunately still failed. Figure 96 shows the output seen:

```


spapply_config:  Changed system partitions:

  sp2en0


spapply_config:  Checking impact on VSD (verparvsd)...
syspar: nodes_in_syspar being processed
sp2en1:           1   2   3   4   5   6   7   8
sp2en0:           9  10  11  12  13  14  15  16
```

Figure 96. Output of smit.log Showing Successful Read of Syspar_map

Here we can see that the Syspar_map is now successfully read and the nodes allocated to their new partitions. Now we can look at the new error that was seen further on in the log as shown in Figure 97 on page 259.

```
spapply_config:  Deleting hb, hr and emon subsystems for changed system  partiti
ons...
0513-085 The hb.sp2en0 Subsystem is not on file.
removing SRC object "hb.sp2en0"
0513-084 There were no records that matched your request.
0513-085 The hr.sp2en0 Subsystem is not on file.
removing SRC object "hr.sp2en0"
0513-084 There were no records that matched your request.
spapply_config:  0022-155 Error removing hr daemon for syspar sp2en0; rc= 1 .
```

Figure 97. Another Partitioning Failure

The errors in the previous example are fairly self-explanatory. The attempted deletion of the existing subsystems failed because they could not be found on file. Again we can manually check from the command line for the existence of these subsystems with the `lssrc` command. This showed us that all of these subsystems were not listed. The reason for this is that when the original partitioning attempt failed, it happened after the point where the old subsystems were deleted, without recreating the new partition subsystems. The output from SMIT shows us this. Refer back to Figure 92 on page 256 to see the subsystems that have been deleted.

How can we go about correcting this new problem?
Unfortunately there is no way as in the previous problem to use a *high level* command to correct this problem for us. We will have to add the subsystems back one by one as a listed entry for source master (SRC) control and into the /etc/inittab file. This is accomplished with the `mksrc` command. The syntax of this command is:

```
<subsystem> -spname <cws_hostname> mksrc
```

We need to perform this command for each subsystem deleted, in this case *hr* and *hb* Figure 98 shows this being performed:

```
-> hr -spname sp2en0 mksrc
0513-071 The hr.sp2en0 Subsystem has been added.
-> startsrc -s hr.sp2en0
0513-059 The hr.sp2en0 Subsystem has been started. Subsystem PID is 23388.
-> hb -spname sp2en0 mksrc
0513-071 The hb.sp2en0 Subsystem has been added.
-> startsrc -s hb.sp2en0
0513-059 The hb.sp2en0 Subsystem has been started. Subsystem PID is 15474.
```

Figure 98. Adding the Missing Subsystems

We also have to remove the “phantom” second SDR daemon *sdr.sp2en1* with a similar command, `rmsrc`, as shown in Figure 99.

```
-> stopsrc -s sdr.sp2en1
0513-044 The stop of the sdr.sp2en0 Subsystem was completed successfully.
-> sdr -spname sp2en1 rmsrc
0513-083 Subsystem has been Deleted.
```

Figure 99. Removing Erroneous Subsystem Entry

The last part of the cleanup process is to change the /etc/inittab file to reflect the changes we have just made. It is a good idea to make a copy of the file before changing it so that if any corruption occurs we can simply put the original back. It should also be noted that it is not recommended to edit the file directly; instead, use the mkitab command as shown in Figure 100.

```
-> mkitab ″hb:2:once:/usr/bin/startsrc -g hb > /dev/console 2>&1″
-> mkitab ″hr:2:once:/usr/bin/startsrc -g hr″
```

Figure 100. Manually Adding the Missing Entries to /etc/inittab

Finally we can proceed with re-applying the new partition. After having corrected the previous problems, partitioning now proceeded through to completion without further errors.

5.21.3 Confirming a Partitioned System is Functioning

Once the apply has completed successfully, it is still a good idea to perform a few last checks before powering on the affected nodes.

- Run the command `splstdata -p` and check the output to make sure the information listed is correct.
- Issue an `lssrc -a` command to make sure all the partition-sensitive subsystems are listed; there should one entry for each partition.

Appendix A. Debugging Perl Scripts

The Practical Extraction Report Language (Perl) is an interpreted language designed to manipulate text, files, and processes. It is powerful and offers a convenient alternative for writing scripts instead of using some of the shell flavors available in AIX. Perl is very cryptic; however, it provides several tools that help programmers to debug and test their code.

Perl has been used by several SP scripts since PSSP 2.1 was introduced. Before that, debugging a PSSP script to diagnose a problem consisted of setting the debug flag for the Korn shell and modifying the code, simply to print out some intermediate information in order to understand why the script was failing.

However, debugging Perl is not a simple task. You have to be familiar with some of the basic concepts, and how Perl programs are organized. Fortunately, the PSSP script code does not use all the richness provided by Perl packaging, which keeps the code as simple as possible when you write Perl programs.

The debugging examples provided in this appendix are based on the setup_server script. If you understand how to debug setup_server and wrappers, you should be able to debug any Perl script present in PSSP.

Although the PSSP scripts are written using Perl Version 4, you will note in this appendix that we used a Perl Version 5 debugger and libraries, since Version 5 provides a debugger that is much more powerful and easy to use than the previous version.

A.1 What Do You Need to Debug Perl Scripts

To start, see if you have everything you need to debug Perl scripts. Start running setup_server in debug mode, as shown in Figure 101.

```
# perl -d setup_server
Can't locate perl5db.pl in @INC.
BEGIN failed--compilation aborted.
```

Figure 101. Running setup_server in Debug Mode

As you can see, an error is encountered: Perl did not find the library with all the functions needed to start up the debugger. The library for Perl Version 5 does not come with the Perl package distributed with PSSP; you have to get it from an Internet repository. Also, be aware that all PSSP Perl scripts use common subroutines for argument passing and validation, data management, and so on. All these subroutines are available when you install the Perl package (ssp.perlpkg) under the /usr/lpp/ssp/perl5/lib directory. Therefore, this is a good place from which to copy the Perl library for the debugger. Once you have copied the file to that directory, you can try to run setup_server in debug mode again. as shown in Figure 102 on page 262. (You will need to copy the dumpvar.pl file also, which is part of the standard Perl package.)

```
# perl -d setup_server
Stack dump during die enabled outside of evals.

Loading DB routines from perl5db.pl patch level 0.94
Emacs support available.

Enter h or h′ for help.

Can′t locate Carp.pm in @INC at /usr/local/lib/perl5/perl5db.pl line 1356
```

Figure 102. Running setup_server in Debug Mode (Missing Path)

This message indicates you have forgotten to include the /usr/lpp/ssp/perl5/lib directory in the search path. Try again, but this time include the path shown in Figure 103.

```
# perl -I /usr/lpp/ssp/perl5/lib -d setup_server
Stack dump during die enabled outside of evals.

Loading DB routines from perl5db.pl patch level 0.94
Emacs support available.

Enter h or h′ for help.

main::(setup_server:146): require ″/usr/lpp/ssp/install/bin/install_def.pl″;

″; # Install definitions
  DB<1> _
```

Figure 103. Running setup_server in Debug Mode (Successfully)

It might be useful to execute the help command, as shown in Figure 104 on page 263.

```
  DB<1> h h
List/search source lines:               Control script execution:
  l [ln|sub]  List source code            T           Stack trace
  - or .      List previous/current line  s [expr]    Single step [in expr]
  w [line]    List around line            n [expr]    Next, steps over subs
  f filename  View source in file         <CR>        Repeat last n or s
  /pattern/   Search forward              r           Return from subroutine
  ?pattern?   Search backward             c [line]    Continue until line
Debugger controls:                        L           List break pts & actions
  O [...]     Set debugger options        t [expr]    Toggle trace [trace expr]
  < command   Command for before prompt   b [ln] [c]  Set breakpoint
  > command   Command for after prompt    b sub [c]   Set breakpoint for sub
  ! [N|pat]   Redo a previous command     d [line]    Delete a breakpoint
  H [-num]    Display last num commands   D           Delete all breakpoints
  = [a val]   Define/list an alias        a [ln] cmd  Do cmd before line
  h [db_cmd]  Get help on command         A           Delete all actions
  |[|]dbcmd   Send output to pager        ![!] syscmd Run cmd in a subprocess
  q or ¬D     Quit                        R           Attempt a restart
Data Examination:             expr     Execute perl code, also see: s,n,t expr
  S [[!]pat]    List subroutine names [not] matching pattern
  V [Pk [Vars]] List Variables in Package.  Vars can be pattern or !pattern
  X [Vars]      Same as "V current_package [Vars]".
  x expr        Evals expression in array context, dumps the result.
  p expr        Print expression (uses script's current package).
```

Figure 104. Getting the Help Menu in Perl Debugger

A.2 Using Debugger Commands

Figure 104 shows all the commands you have available from the command line. You can use just some of them to debug your script. You need very few commands in order to debug a Perl script. The first thing you will want to know is what functions or subroutines the script is calling. Use the S command to list the routines in the script, as shown in Figure 105 on page 264.

```
  DB<2> S
Term::ReadLine::Stub::Features
Term::ReadLine::Stub::IN
Term::ReadLine::Stub::MinLine
Term::ReadLine::Stub::OUT
Term::ReadLine::Stub::ReadLine
Term::ReadLine::Stub::addhistory
Term::ReadLine::Stub::findConsole
Term::ReadLine::Stub::new
Term::ReadLine::Stub::readline
main::BEGIN
main::CWS_tasks
main::allocate_resources
main::check_prereqs
main::config_services
main::create_files
main::exit_setup_server
main::export_clients
main::get_ticket
main::make_CWS_client
main::make_clients
main::make_config
main::make_install
main::make_interfaces
main::make_master
main::make_resources
main::parse_args
main::read_SDR
main::remove_ticket
main::unconfig_clients
main::unconfig_nim
```

Figure 105. List of Subroutines

After you get the list of subroutines, you may want to know where in the script you are. This can be done by using the w command, as shown in Figure 106.

```
  DB<2> w
143:    # 21 Remove authentication ticket.     /bin/rm /tmp/tkt_rcmd
144:    #
145:    #-------------------------------------#-------------------------
146==>  require "/usr/lpp/ssp/install/bin/install_def.pl"; # Install defi
147:    require "/usr/lpp/ssp/install/bin/install_lib.pl"; # Install subr
148:    require "/usr/lpp/ssp/bin/sminstmsgs.pl";         # Install mess
149:
150:    #--------------------------------------------------------------
151:    # Common Variables - The following variables are common (global)
152:    #                    main and all subroutines.  All other variabl
```

Figure 106. Visualizing a Window Around the Current Line

The arrow in Figure 106 indicates the line which will be executed the next time. You can see that the *require* statements are used to include external files into the script at run time. If you look carefully at the help menu, you will find many variations of this command, and other listing commands as well.

Now that you know where you are, you can start executing the script step-by-step, or you can execute the script in normal mode until you reach a desirable breakpoint. For step-by-step execution, you have two choices, depending upon whether you want to go into a subroutine or just execute over that subroutine. Let us go to the starting point of the main code. To do that, you need to know the line number. After listing a bigger window using the l command, we know that the starting point is in line number 195, so let us go there, as shown in Figure 107.

```
  DB<3> c 195
main::(setup_server:195):      unless(&parse_args){    # Parse the comma
```

Figure 107. Jumping to a Specific Location within the Script

If you are using Perl 4, you have to set a breakpoint first by using the b command. Then continue with:

```
  DB<3> b 195
  DB<4> c
main::(setup_server:199):      unless(&read_SDR){   # Get info from SDR
```

Figure 108. Setting a Breakpoint Using Perl 4

At this point, you have the option of either going into the subroutine (in this case the subroutine is parse_args), or stepping over it. Let us step over it, as shown in Figure 109.

```
  DB<4> n
main::(setup_server:199):      unless(&read_SDR){   # Get info from SDR
```

Figure 109. Stepping over a Subroutine

As you can see, you are stepped over the subroutine, and now the current line is in the next call (read_SDR subroutine).

Now that you know how to move around the script, let′s go into more detail to see what you can do with variables and expressions within subroutines.

A.3 Subroutines, Variables, and Expressions

When you debug a script, you want to know why the script is failing in a specific location. Perhaps you want to check if the variable being used contains the correct value, or if the value that is being set into the SDR is the one you expected to be.

Let′s jump to a specific subroutine and check its variable values, as shown in Figure 110 on page 266. Let us go to CWS_task, one of the first subroutines within setup_server:

```
  DB<4> c main::CWS_tasks
setup_server: Running services_config script to configure SSP services.Th
rc.ntp: NTP already running - not starting ntp
0513-029 The supfilesrv Subsystem is already active.
Multiple instances are not supported.
main::CWS_tasks(setup_server:465):        local($rc);
  DB<5> w
462:    # CWS_tasks - CWS-specific tasks to be performed on the CWS ONLY!
463:    #------------------------------------#------------------------
464:    sub CWS_tasks {
465==>    local($rc);
466:
467:      if($local_node_numb == 0){
468:          system("$SETUP_CWS");              # Run "setup_CWS"...
469:          if($? > 0) {                       # If it failed then
470:              $rc = $? /256;                     # convert rc; con
471:              &cat_sminstmsg("emsg279",$progname,$SETUP_CWS,$rc); # m
```

Figure 110. Jumping to a Subroutine

In the Perl debugger, variables and expression are managed similarly. The only difference is with the command you use to retrieve their values.

While a variable contains a value itself, an expression is evaluated each time you retrieve its value. Figure 111 shows some examples.

```
  DB<5> X local_node_numb
$local_node_numb = 0
  DB<6> X SETUP_CWS
$SETUP_CWS = '/usr/lpp/ssp/bin/setup_CWS'
  DB<7> x $local_node_numb == 0
0  1
  DB<8> x $SETUP_CWS
0  '/usr/lpp/ssp/bin/setup_CWS'
```

Figure 111. Variables and Expression in Perl Debugger

As you can see, the X command is used to *get* variable values, while the x command is used to *evaluate* expressions.

If you are using Perl 4, use the p command to evaluate expressions, instead of the x command.

A.4 Problem Determination Using the Perl Debugger

Now let's use what you have learned about debugging Perl scripts. The following example corresponds to a common problem related to the NFS file system previously exported. Although this is a simple problem that does not necessarily have to be solved using the debugger, it allows us to navigate through setup_server, and discover the problem in one of its wrappers.

A.4.1 Symptom

We want to install node 5. In order to install it we can use the spbootins command, as shown in Figure 112.

```
# spbootins -r install -s yes -l 5
```

Figure 112. Running the spbootins Command

This command will run setup_server for us. Assuming that node 5 is using the Control Workstation as boot/install server, setup_server will run locally. The output of setup_server is partially shown in Figure 113.

```
partial output from setup_server .......
mknimclient: 0016-242: Client node 5 (sp21n05.msc.itso.ibm.com) already d
mknimclient: 0016-242: Client node 6 (sp21n06.msc.itso.ibm.com) already d
mknimclient: 0016-242: Client node 7 (sp21n07.msc.itso.ibm.com) already d
mknimclient: 0016-242: Client node 8 (sp21n08.msc.itso.ibm.com) already d
mknimclient: 0016-242: Client node 9 (sp21n09.msc.itso.ibm.com) already d
mknimclient: 0016-242: Client node 11 (sp21n11.msc.itso.ibm.com) already
mknimclient: 0016-242: Client node 13 (sp21n13.msc.itso.ibm.com) already
mknimclient: 0016-242: Client node 15 (sp21n15.msc.itso.ibm.com) already
Usage: chnfsexp -d directory [ -f exports_file ]
        [-t {rw|ro}{rm -h hostname[,hostname]...}]
        [-a uid]
        [-r hostname[,hostname]...
        [-c hostname[,hostname]...
        [-s|-n] [-I|-B|-N]
export_clients: 0016-015 Error found while trying to export
/spdata/sys1/install/pssplpp on install server.  Return code from /usr/sb
setup_server: 0016-279 Failure of internally called command: /usr/lpp/ssp
```

Figure 113. Error Running setup_server

As you can see the error is clearly related to the exported NFS file system. It looks like the chnfsexp command is misformed in the *export_clients* wrapper.

A.4.2 Researching the Problem

To research the problem, let us start by running setup_server again, using debug mode this time, as shown in Figure 114.

```
perl -I /usr/lpp/ssp/perl5/lib -d setup_server
Stack dump during die enabled outside of evals.

Loading DB routines from perl5db.pl patch level 0.94
Emacs support available.

Enter h or h h′ for help.

main::(setup_server:146):       require ″/usr/lpp/ssp/install/bin/install
″; # Install definitions
  DB<1>
```

Figure 114. Debugging setup_server

Now let us see the subroutines that are present, and try to find the one related to the export_client wrapper. Figure 115 on page 268 shows these subroutines.

```
  DB<1> S
Term::ReadLine::Stub::Features
Term::ReadLine::Stub::IN
Term::ReadLine::Stub::MinLine
Term::ReadLine::Stub::OUT
Term::ReadLine::Stub::ReadLine
Term::ReadLine::Stub::addhistory
Term::ReadLine::Stub::findConsole
Term::ReadLine::Stub::new
Term::ReadLine::Stub::readline
main::BEGIN
main::CWS_tasks
main::allocate_resources
main::check_prereqs
main::config_services
main::create_files
main::exit_setup_server
main::export_clients
main::get_ticket
main::make_CWS_client
main::make_clients
main::make_config
main::make_install
main::make_interfaces
main::make_master
main::make_resources
main::parse_args
main::read_SDR
main::remove_ticket
main::unconfig_clients
main::unconfig_nim
  DB<1>
```

Figure 115. List Subroutines

From the list you can see the subroutine we are looking for. It is called export_clients and is part of the main. package. Let us advance to that subroutine by using the c command, as shown in Figure 116.

```
  DB<1> c main::export_clients

output from the previous part of setup_server .......

mknimclient: 0016-242: Client node 11 (sp21n11.msc.itso.ibm.com) already
mknimclient: 0016-242: Client node 13 (sp21n13.msc.itso.ibm.com) already
mknimclient: 0016-242: Client node 15 (sp21n15.msc.itso.ibm.com) already
main::export_clients(setup_server:711):
711:      local($rc);
  DB<2>
```

Figure 116. Advancing to export_client Subroutine

Now you can display a window arround the current line in the export_clients subroutine, as shown in Figure 117 on page 269.

```
  DB<2> w
708:    # export_clients
709:    #-----------------------------------------------------------------
710:    sub export_clients {
711==>    local($rc);
712:
713:      system("$EXPORTCLIENTS");                    # Export pssplpp to
714:      if($? > 0) {                                 # If we failed then
715:          $rc = $? /256;                             # convert rc; con
716:          &cat_sminstmsg("emsg279",$progname,$EXPORTCLIENTS,$rc); # m
717:          print (STDERR $CAT_MSG);                   # print message
  DB<2>
```

Figure 117. Displaying a Window Arround the Current Line

The subroutine made a system call using EXPORTCLIENTS as a variable. Take a look at its value, as shown in Figure 118.

```
  DB<2> X EXPORTCLIENTS
$EXPORTCLIENTS = '/usr/lpp/ssp/bin/export_clients'
  DB<3>
```

Figure 118. The EXPORTCLIENTS Variable

There is not much we can do here. As you can see, *export_clients* made the system call to run another Perl script. There are no arguments passed to that script, so you can exit from both the debugger and from running the script in debug mode, and continue seeking the problem elsewhere.

Run the script `/usr/lpp/ssp/bin/export_clients` in debug mode, as shown in Figure 119 on page 270.

```
perl -I /usr/lpp/ssp/perl5/lib -d export_clients
Stack dump during die enabled outside of evals.

Loading DB routines from perl5db.pl patch level 0.94
Emacs support available.

Enter h or h h′ for help.

main::(export_clients:74):      require ″/usr/lpp/ssp/install/bin/install
″; # Install definitions
  DB<1>
  DB<1> S
Term::ReadLine::Stub::Features
Term::ReadLine::Stub::IN
Term::ReadLine::Stub::MinLine
Term::ReadLine::Stub::OUT
Term::ReadLine::Stub::ReadLine
Term::ReadLine::Stub::addhistory
Term::ReadLine::Stub::findConsole
Term::ReadLine::Stub::new
Term::ReadLine::Stub::readline
main::BEGIN
main::by_number
main::check_prereqs
main::export_fs
main::parse_args
main::read_SDR
  DB<1>
```

Figure 119. Running export_clients Script in Debug Mode

You have also displayed the subroutines list. As a second step, you need to know where in the script you are. To do this, use the window command, as shown in Figure 120.

```
  DB<1> l
81:     #                   should remain (or at least be considered) lo
82:     #                   their respective subroutines.
83:     #-----------------------------------#------------------------
84:     $progname=″export_clients″;          # Program name for message
85:
86:     $local_node_numb=″″;                 # Node number from ODM
87:
88:     %hostname=″″;                        # Node′s reliable hostname
89:     #%shostname=″″;                       # Node′s short hostname a
90:     %bootserver=″″;                      # Node′s boot server
  DB<1> c 84
```

Figure 120. Listing the export_clients Wrapper

As you can see, the program starts at line number 84. Move to line 84 and once again list the contents around the current line, as shown in Figure 121 on page 271.

```
  DB<1> c 84
main::(export_clients:84):      $progname=″export_clients″;            #
  DB<2>

  DB<2> l
97==>   $server_name=″″;                       # Current node′s boot serv
98:
99:     $CAT_MSG=″″;                           # String for hold message
100:    #
101:    #---------------------------------------------------------------
102:    # Main line code
103:    #-------------------------------------#-------------------------
104:    unless(&parse_args){                   # Parse the command line
105:        exit -1;
106:    }
  DB<3>
```

Figure 121. Jumping to the Main Line of Code

Now that everything is initialized, you can advance until you get to where you we think the problem is, as shown in Figure 122.

```
  DB<2> c 104
main::(export_clients:104):     unless(&parse_args){                   #
  DB<3> n
main::(export_clients:108):     unless(&read_SDR){                     #
  DB<3> n
main::(export_clients:112):     unless(&check_prereqs){                #
  DB<3> n
main::(export_clients:116):     unless(&export_fs){                    #
  DB<3> s
main::export_fs(export_clients:234):
234:      local($rc,$node_list,$test_list,@OUTLINES,@PSSPLPP_LINES,$num_l
  DB<3>
  DB<5> c 267
main::export_fs(export_clients:267):
267:      chop(@OUTLINES=$EXPORTFS);        # Get all exported filesys
```

Figure 122. Narrowing Down the Problem

In line number 267, you can see that the script is getting the list of exported file systems. Display the values of OUTLINE and EXPORTFS variables after this line to check their contents, as shown in Figure 123.

```
  DB<6> x $EXPORTFS
0  ′/usr/sbin/exportfs′
  DB<7> s
main::export_fs(export_clients:268):
268:      if($? > 0) {                                   # If error...
  DB<7> x @OUTLINES
0  ′/spdata/images                     -root=risc38′
1  ′/cdrom                             -ro′
2  ′/spdata/sys1/install/pssplpp/PSSP-2.2 -ro′
  DB<8>
```

Figure 123. Getting Closer to the Problem

As shown in Figure 123, the OUTLINES variable contains the list of all file systems currently exported in the system. See what this subroutine is doing with this value, as shown in Figure 124 on page 272.

```
  DB<8> w
271:          print (STDERR $CAT_MSG);                  #  print message
272:          return $NOK;                              #  return no good
273:      }
274==>    @PSSPLPP_LINES=grep(/$PSSPLPP_DIR/,@OUTLINES); # See if pssplpp
275:      if($#PSSPLPP_LINES < 0){                       # If it's case #
276:          system("$RMNFSEXP -d $PSSPLPP_DIR >/dev/null"); # Redo expo
277:          system("$MKNFSEXP -d $PSSPLPP_DIR -t ro -r $node_list -c $n
278:          if($? > 0) {                                  # If error...
279:              $rc = $? /256;                            #  convert rc
280:              &cat_sminstmsg("emsg015",$progname,$PSSPLPP_DIR,$MKNFSE
  DB<8> x grep(/$PSSPLPP_DIR/,@OUTLINES)
0  '/spdata/sys1/install/pssplpp/PSSP-2.2 -ro'
  DB<9>
  DB<10> s
main::export_fs(export_clients:290):
290:      unless($num_lines=grep(/$test_list/,@PSSPLPP_LINES)){ # If wron
  DB<10>
```

Figure 124. Evaluating OUTLINES Variable

According to line 275, the script is checking if the /spdata/sys1/install/pssplpp directory has been already exported. In the evaluation of the *grep* statement you can see that this is partially true, since the PSSP-2.2 sub-directory is exported read-only to everyone. Option 2 was chosen by the script which says that the pssplpp directory is exported, but this is an incorrect export list. Let's examine this portion of the script, as shown in Figure 125.

```
  DB<11> w
288:    # Now check for case # 2 - pssplpp exported but wrong export list
289:    #
290:      unless($num_lines=grep(/$test_list/,@PSSPLPP_LINES)){ # If wron
291==>        system("$CHNFSEXP -d $PSSPLPP_DIR -t ro -r $node_list -c $n
292:          if($? > 0) {                                  # If error...
293:              $rc = $? /256;                            #  convert rc
294:              &cat_sminstmsg("emsg015",$progname,$PSSPLPP_DIR,$CHNFSE
295:              print (STDERR $CAT_MSG);                  #  print mess
296:              return $NOK;                              #  return no
297:          }
  DB<11>
```

Figure 125. The pssplpp Directory is Exported but Wrong Export List

As we can see in Figure 125, the second option in the script executes the /usr/sbin/chnfsexp command using the PSSPLPP_DIR variable as one of the parameters.

However, the script made a wrong decision because the /spdata/sys1/install/pssplpp directory is not the one that is exported. The PSSP-2.2 directory under pssplpp is the exported directory.

Therefore, this command will fail with the error message you saw at the begining of this example. The solution to this problem is quite simple: Unexport the PSSP-2.2 directory and re-run setup_server.

In summary, this example was not intended to provide a complex setup_server problem, but to explain the main step needed to debug a complex Perl script.

Appendix B. How to Get FixDist

FixDist is a user-friendly graphical user interface that enables a customer to download software fixes from an anonymous FTP server.

FixDist provides access to the latest available fixes via the Internet. With FixDist, a customer can choose to download immediately or configure to download later. FixDist supports both AIX selective fixes as well as AIX Preventive Maintenance Packages.

The FixDist utility communicates to the server using anonymous FTP. There is no mail transport or Telnet requirement. See the section on FixDist Server information for instructions on how to download the FixDist client code.

- Features/Functions

  FixDist version 1.0 provides an array of functions, including the automatic notification when there is a newer version of the FixDist client code.

  With FixDist, customers can view fixes by fix number, APAR number, product names or by subsystem names. Once a fix is selected, the utility attempts to deliver only the necessary fixes to satisfy all requisites. This speeds up delivery of the fix package because any fixes you may have already received are not downloaded.

- Prerequisites to using FixDist

  A connection to the Internet is required to use FixDist. FixDist is supported on AIX Version 3.2.4 or above running AIXwindows 2D Version 1.2 or above.

- FixDist Server information

  The FixDist client code can be downloaded from the anonymous FTP site:

  ```
  aix.boulder.ibm.com (198.17.57.66)
  ```

- How to download FixDist

  Use the following procedure to download the FixDist client code from the server. The following procedure must be performed as root.

  1. Create a directory for the client code by entering:

     ```
     mkdir /tmp/fixdist
     cd /tmp/fixdist
     ```

  2. Connect to the FixDist server by entering:

     ```
     ftp aix.boulder.ibm.com
     ```

  3. At the login prompt, enter:

     ```
     anonymous
     ```

  4. At the password prompt, enter:

     ```
     YourEmailAddress
     ```

     where YourEmailAddress is your full e-mail address.

  5. When the system displays the ftp> prompt, enter:

     ```
     bin
     cd fixdist_client_code
     get fd.tar.Z
     quit
     ```

- How to install FixDist

  Use the following procedure to install the FixDist client code. The following procedure must be performed as root.

  1. Change to the /(root) directory.

     ```
     cd /
     ```

  2. Uncompress the downloaded fd.tar.Z file by entering:

     ```
     zcat /tmp/fixdist/fd.tar.Z | tar -xpvf -
     ```

B.1.1 FixDist Server Locations

• `rwww.aix.can.ibm.com`	`204.138.188.126`	(Canada)
• `www.ibm.de`	`192.109.81.2`	(Germany)
• `fixdist.yamato.ibm.co.jp`	`202.32.4.20`	(Japan)
• `ftp.nordic.ibm.com`	`192.12.214.80`	(Nordic)
• `ftp.europe.ibm.com`	`193.129.186.2`	(United Kingdom)
• `service.software.ibm.com`	`198.17.57.66`	(United States)

Appendix C. Setting NIM to Debug Mode

Use the following steps to set NIM in debug mode during install:

1. Check that the node to run NIM in debug mode is set to "disk." If it is not, then set bootp_response to "disk" by entering:

   `spbootins -r disk <Frame#> <Node#> <Number_of_nodes>`

2. Set the SPOT to debug mode:

   `nim -o check -a debug=yes <spot>`

3. Examine the SPOT for the NIM debug hooks by entering:

   `lsnim -l <spot>`

```
->lsnim -l spot_aix414
spot_aix414:
   class        = resources
   type         = spot
   version      = 04
   server       = master
   location     = /spdata/sys1/install/aix414/spot/spot_aix414/usr
   alloc_count  = 0
   Rstate       = ready for use
   prev_state   = ready for use
   release      = 01
   if_supported = rs6k ent
   if_supported = rs6k fddi
   if_supported = rs6k tok
   if_supported = rs6ksmp ent
   if_supported = rs6ksmp tok
   if_supported = rspc ent
   if_supported = rspc tok
   enter_dbg    = ″rs6k 0x0013f4b0″
   enter_dbg    = ″rs6ksmp 0x0015f4f8″
   enter_dbg    = ″rspc 0x0013f4b0″
```

Figure 126. Output from lsnim on a Debug SPOT

   As you can see from Figure 126, there are three entries which have the field value `enter_debug = <machine>`, where `machine` is the processor type and memory address. Pick the correct type and note the address. As an example, we will be running NIM in debug mode on a High node, so the address we need to note is `0x0015f4f8`.

4. Change the node′s bootp_response to "install" to allocate the new SPOT by entering:

   `spbootins -r install <Frame#> <Node#> <Number_of_nodes>`

5. Now manually condition the node or nodes for debugging. Note that if you wish to save the output to a file, you must use an aixterm and then the `s1term` command at the console. The reason for this is that the Logging option will be used and this is only available on the aixterm.

6. To start logging from the aixterm, place the mouse pointer in the TTY window and press the Ctrl key and the left mouse button simultaneously. A menu will be displayed.

Simply move the cursor to the Logging option and all screen output will now also be saved in a log. The log will be kept on the user ID that initiated the spmon GUI. The log file will have the filename of AixtermLog.*xxxxxx*, where *xxxxxx* is a random identifier.

Once the expected LED hang has occurred, rename the AixtermLog before halting logging to preserve the file. The reason for this is that the file is removed when logging is stopped.

7. Once the boot starts, you will be prompted with >. This is where you enter the address, except for the starting 0x. Therefore, using these addresses, we enter the following, as shown in Figure 127.

```
<< 299 >>GPR0  00000001 0015F254 0015F478 00268A10 002689F0 00158660 0000
0
000001
GPR8  00000000 00000000 001588B0 00000010 00154938 003001C8 0000009C 0000
GPR16 00000020 080000A4 1FFEB978 00000000 1FFEB988 0000354C 00003548 0000
GPR24 00003564 00003558 0000355C 1FF24000 0000357C 00000000 0008F9D0 0000

MSR 00021030  CR   48220422  LR   0015493C  CTR   00000000  MQ   00000000
XER 00000000  SRR0 00060DBC  SRR1 00021030  DSISR 40000000  DAR  D0273000

IAR 00060DBC  (ORG+00060DBC)  ORG=00000000   Mode: VIRTUAL
00060DB0   00000000 00000000 00000000 7C810808   |............|...|
                                      |     tweq   r1,r1
00060DC0   4E800020 7C0802A6 BFA1FFF4 90010008   |N.. |...........|

                                      |
00060DB0   00000000 00000000 00000000 7C810808   |............|...|
00060DC0   4E800020 7C0802A6 BFA1FFF4 90010008   |N.. |...........|
00060DD0   9421FFB0 38600000 4BFA8229 90610038   |.!..8..K..).a.8|
00060DE0   3C608000 38800000 4BFA67E9 800310B8   |<..8...K.g.....|
00060DF0   808310BC 3BA30000 7C002040 33DD10B8   |....;...|. @3...|
00060E00   33FD10BC 40810014 4800037D 387E0000   |3...@...H..}8..|
00060E10   389F0000 480004A5 387D0000 4BFA67B9   |8...H...8}..K.g.|

Trap instruction interrupt.
>0> st 0015f4f8 2
>0> g
```

Figure 127. Initial Screen Seen When Starting NIM in Debug Mode

The number 2 represents the detail level for the debugger. Level 2 is good enough for our purposes.

8. The netboot will be displayed “live” on this console. It may be helpful to have the 3DigitDisplay and the Node Front Panel displayed so you can watch the LEDs change as the netboot proceeds.

Important

There may be a case where the following message is seen repeatedly before you even have a chance to enter the NIM hook address:

```
> 032-001  You entered a command "" that is not valid.
```

This may occur because, in order to run NIM in debug mode, you must have a node supervisor card microcode level of at least 1294. To check the level of microcode, either open a codeVersion window from the "All node summary list" or run the following command:

```
/usr/lpp/ssp/bin/spmon -G  -q -l -M frame*/node*/codeVersion/value
```

If the level is below 1294, then you must contact your local hardware engineer to have the microcode upgraded.

9. When the nodes are installed, the only thing that needs to be done is to create a regular version of the SPOT. From the Control Workstation, enter `nim -Fo check psspspot`.

Appendix D. Manual Node Conditioning on a High Node

The following steps detail the procedure for performing manual node conditioning on a High node. In these steps, we used the spmon commands, as opposed to using the GUI, but either will work.

1. Power the node off by entering:

   ```
   spmon -power off node1
   ```

2. Set the key to service, by entering:

   ```
   spmon -key service node1
   ```

3. Open a console with the node powered off, by entering:

   ```
   spmon -o node1
   ```

4. Press Enter and you will get the bump prompt. Type sbb to enter the BUMP menus, as shown in Figure 128.

```
> sbb
```

Figure 128. The BUMP Console Prompt

5. Choose option **1** from the menu, as shown in Figure 129.

```
  STAND-BY MENU :  rev 17.03

0 Display Configuration
1 Set Flags
2 Set Unit Number
3 Set Configuration
4 SSbus Maintenance
5 I2C Maintenance






Select(x:exit): 1
```

Figure 129. The Stand-by Menu

6. Check that Fast IPL is enabled. If it is disabled, then set it to enabled. (Note that Fast IPL gets reset after booting, so it needs to be checked or changed after each reboot.)

```
  Set Flags

0 Remote Authorization                          Disabled
1 Bump Console Present                          Enabled
2 Autoservice IPL                               Disabled
3 Extended Tests                                Disabled
4 PowerOn Tests in Trace Mode                   Disabled
5 PowerOn Tests in Loop  Mode                   Disabled
6 Fast IPL                                      Disabled
7 Set Electronic Mode Switch to Normal          NRM




Select(x:exit): 6
```

Figure 130. Setting Fast IPL

7. Exit console:

   ```
   ctrl - x
   ```

8. Set the node to secure mode:

   ```
   spmon -key secure node1
   ```

9. Power the node on:

   ```
   spmon -power on node3
   ```

   At this point, you can open a read-only console to see the processor checks being made, as shown in Figure 131 on page 283.

```
        BUMP FIRMWARE   - Mar 25, 1996
        ID 09.22 - POWER_ON in EPROM

#
       FLOPPY NOT READY !
DO YOU WANT TO UPDATE FLASH FROM LINE S2 ·y/n“ ? n

        BUMP FIRMWARE   - Jul 19, 1996
        ID D9.23 - POWER_ON in FLASH PROM

 - Low Interleaving -
Initial test on processor 0  - * OK !
Initial test on processor 1  - * OK !
Initial test on processor 2  - * OK !
Initial test on processor 3  - * OK !
Initial test on processor 4  - * OK !
Initial test on processor 5  - * OK !
Init 1024 Kb L2 cache by processor 0 -> ** OK !
Init 1024 Kb L2 cache by processor 1 -> ** OK !
Init 1024 Kb L2 cache by processor 2 -> ** OK !
Init 1024 Kb L2 cache by processor 3 -> ** OK !
Init 1024 Kb L2 cache by processor 4 -> ** OK !
Init 1024 Kb L2 cache by processor 5 -> ** OK !
Clearing 512 Mb memory by processor 0 -> ********************************


        CPU FIRMWARE    - August 4, 1994
        Processor 0 on IPL INIT
{{ 216 }}
{{ 220 }}
{{ 288 }}
{{ 292 }}
{{ 292 }}
{{ 292 }}
{{ 280 }}
##WARNING    Local message : SECURE MODE !
        STALL STATUS
```

Figure 131. Initial Boot Checks on SMP Node

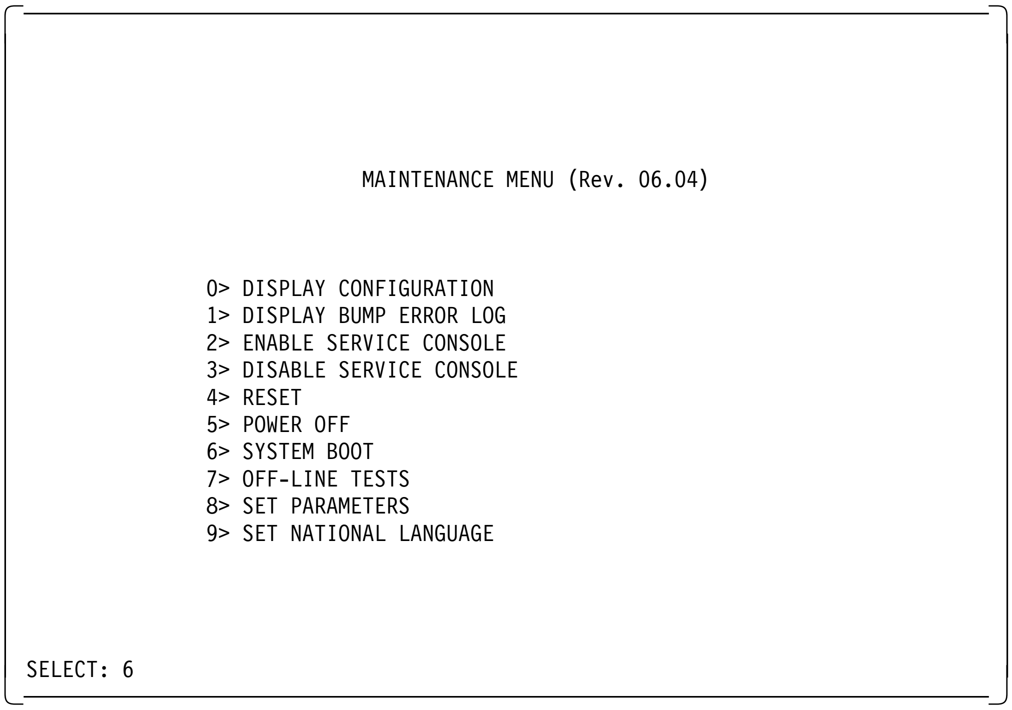
10. The node will now be at LED 200. Set the node to service mode and perform *two* resets, as follows:

    ```
    spmon -key secure node1
    ```

    ```
    spmon -reset node1
    ```

    ```
    spmon -reset node1
    ```

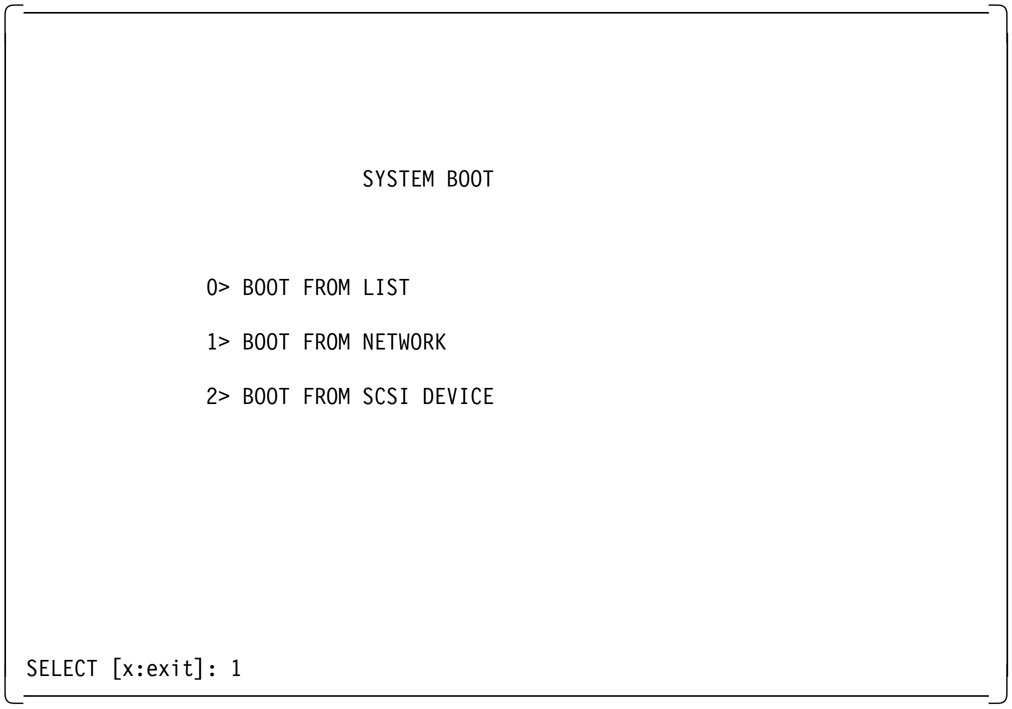
11. Open a console. When prompted with the screen shown in Figure 132 on page 284, select **SYSTEM BOOT**.



```
                    MAINTENANCE MENU (Rev. 06.04)


        0> DISPLAY CONFIGURATION
        1> DISPLAY BUMP ERROR LOG
        2> ENABLE SERVICE CONSOLE
        3> DISABLE SERVICE CONSOLE
        4> RESET
        5> POWER OFF
        6> SYSTEM BOOT
        7> OFF-LINE TESTS
        8> SET PARAMETERS
        9> SET NATIONAL LANGUAGE




SELECT: 6
```

Figure 132. SMP Node's Maintenance Menu

12. You will then be prompted with the system boot menu. Select option **1** to boot from the network, as shown in Figure 133.



```
                    SYSTEM BOOT


        0> BOOT FROM LIST

        1> BOOT FROM NETWORK

        2> BOOT FROM SCSI DEVICE






SELECT [x:exit]: 1
```

Figure 133. System Boot Selection Menu

13. You will now be prompted with the normal node conditioning main menu. Select option **1**:

    1. Select BOOT (Startup) Device

Then select option **3**; this is okay for an SMP node.

```
3. Ethernet:  Slot 1/7, BNC / modular Jacks
```

14. Return to the main menu and select option **4** to boot the system.

15. Set node to normal (this will have to be done from another screen or from the GUI front panel for the node), as follows:

    ```
    spmon -key normal node1
    ```

16. Press Enter to start the network boot.

Appendix E. Flowcharts of setup_server Wrappers

This appendix provides flowcharts of all setup_server wrappers.

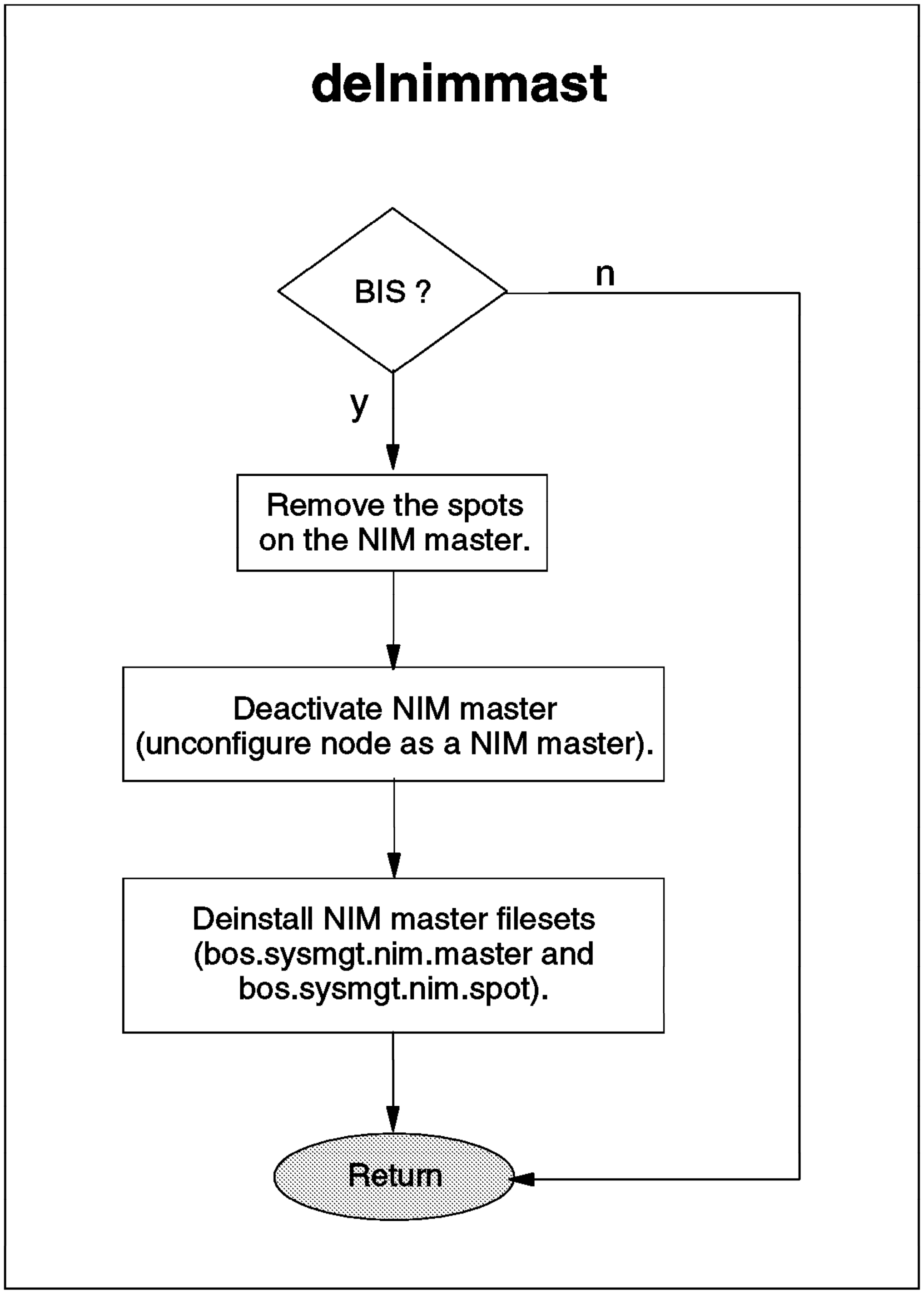

delnimmast
BIS ?
n
y
Remove the spots on the NIM master.
Deactivate NIM master (unconfigure node as a NIM master).
Deinstall NIM master filesets (bos.sysmgt.nim.master and bos.sysmgt.nim.spot).
Return

Figure 134. delnimmast Wrapper Flowchart

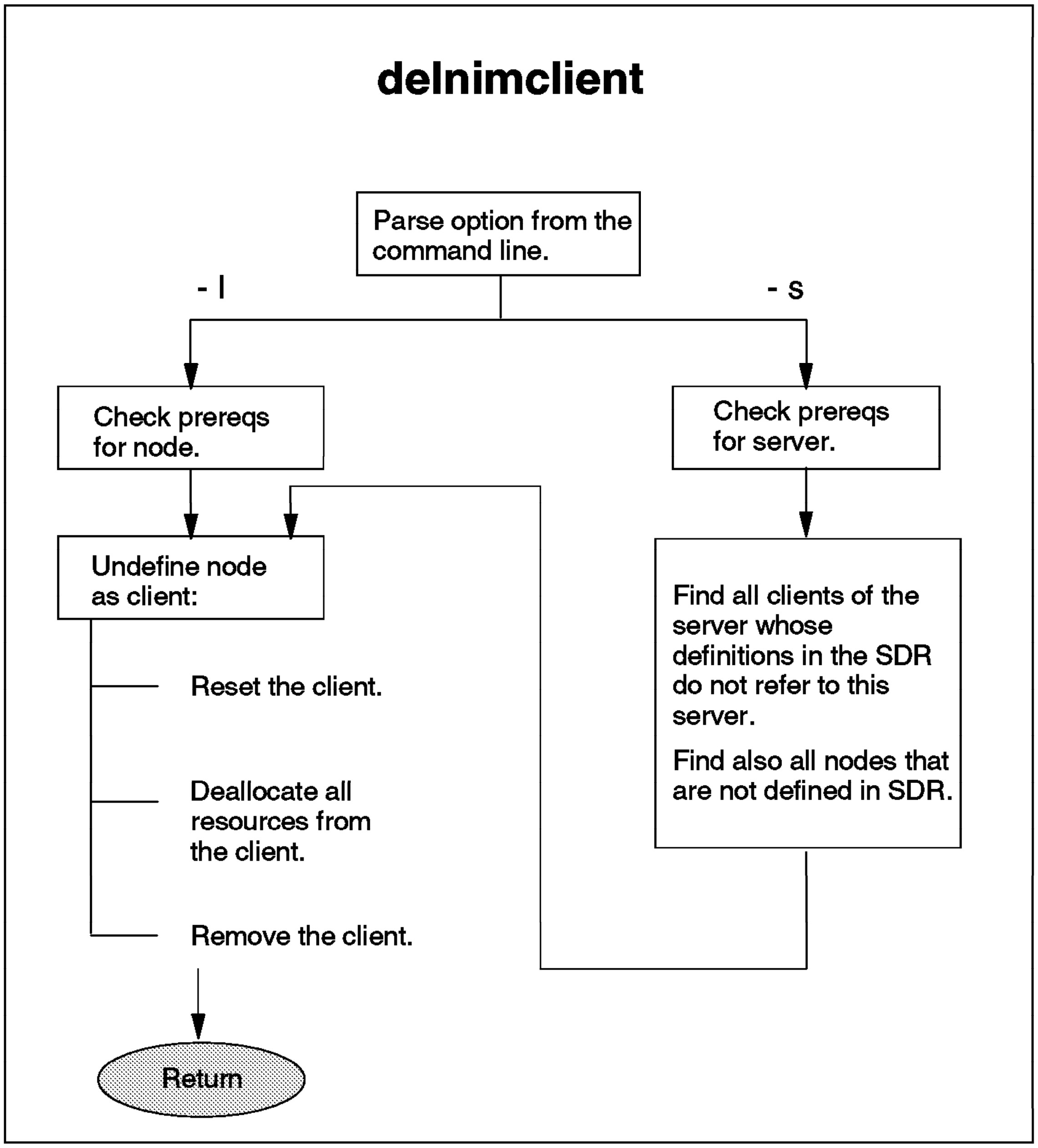

delnimclient
Parse option from the command line.
- l
- s
Check prereqs for node.
Check prereqs for server.
Undefine node as client:
Reset the client.
Deallocate all resources from the client.
Remove the client.
Find all clients of the server whose definitions in the SDR do not refer to this server.
Find also all nodes that are not defined in SDR.
Return

Figure 135. delnimclient Wrapper Flowchart

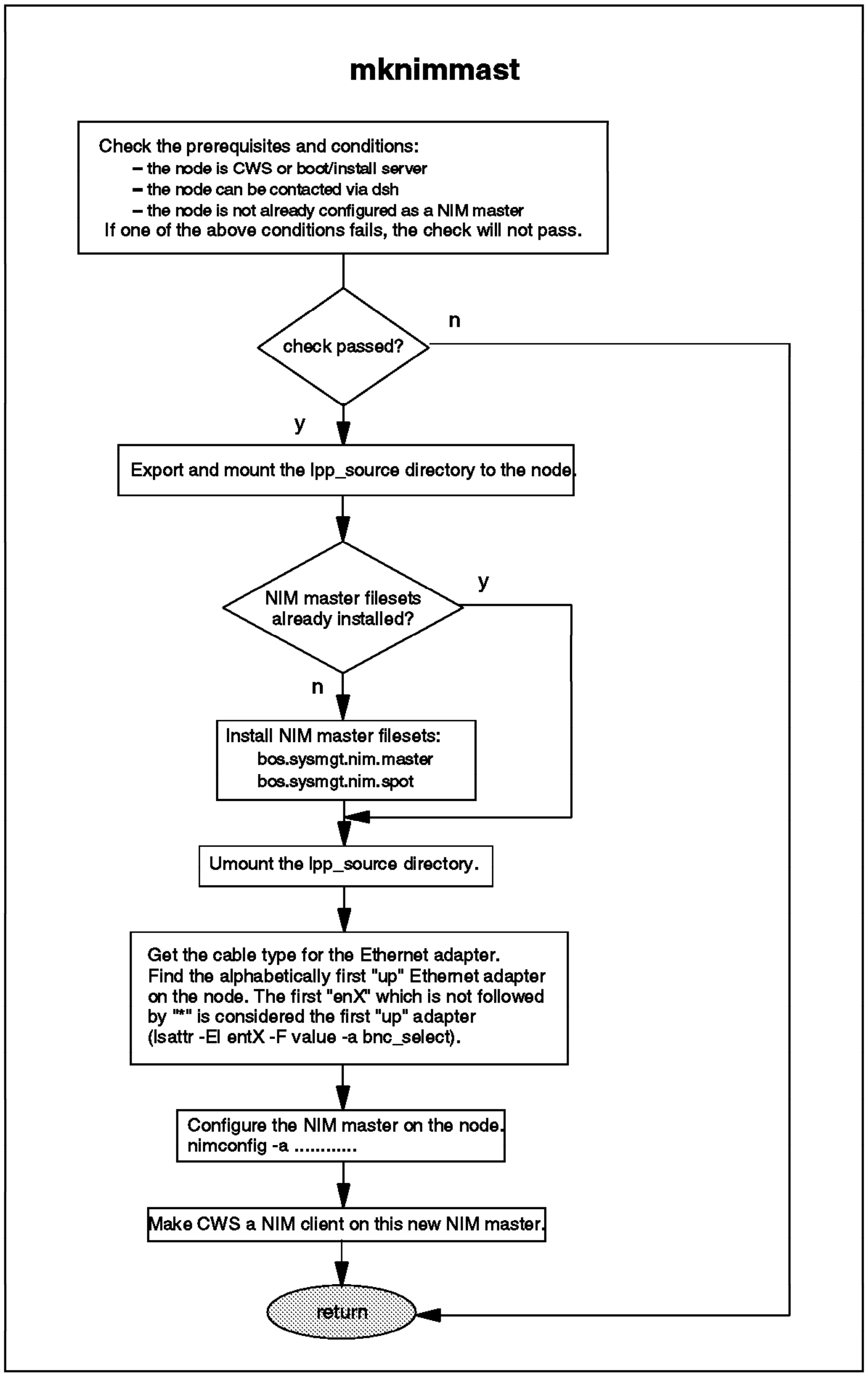

mknimmast
Check the prerequisites and conditions:
– the node is CWS or boot/install server
– the node can be contacted via dsh
– the node is not already configured as a NIM master
If one of the above conditions fails, the check will not pass.
check passed?
n
y
Export and mount the lpp_source directory to the node.
NIM master filesets already installed?
y
n
Install NIM master filesets:
bos.sysmgt.nim.master
bos.sysmgt.nim.spot
Umount the lpp_source directory.
Get the cable type for the Ethernet adapter.
Find the alphabetically first "up" Ethernet adapter on the node. The first "enX" which is not followed by "*" is considered the first "up" adapter (lsattr -El entX -F value -a bnc_select).
Configure the NIM master on the node.
nimconfig -a ....
Make CWS a NIM client on this new NIM master.
return

Figure 136. mknimmast Wrapper Flowchart

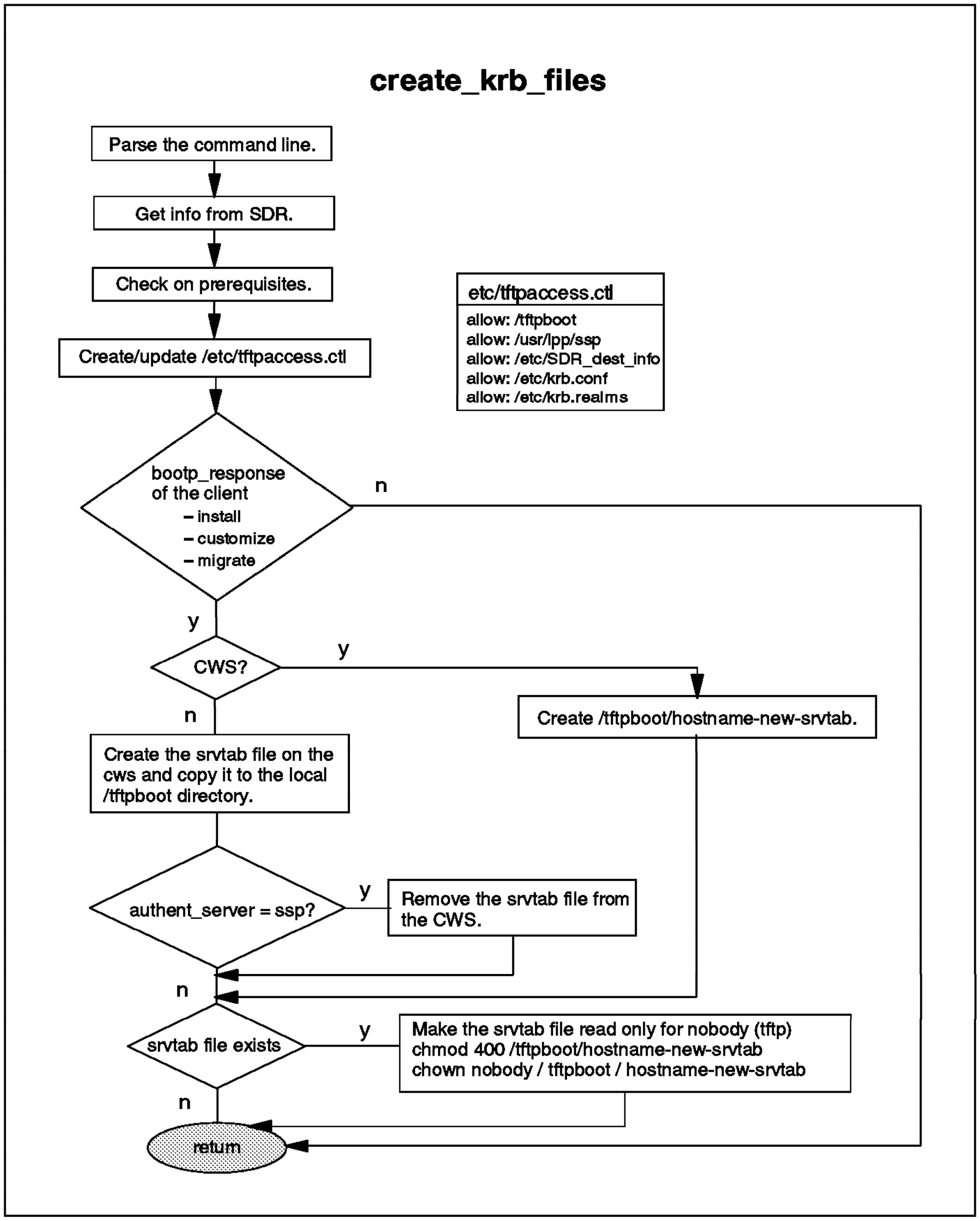

create_krb_files
Parse the command line.
Get info from SDR.
Check on prerequisites.
Create/update /etc/tftpaccess.ctl
etc/tftpaccess.ctl
allow: /tftpboot
allow: /usr/lpp/ssp
allow: /etc/SDR_dest_info
allow: /etc/krb.conf
allow: /etc/krb.realms
bootp_response of the client
– install
– customize
– migrate
n
y
CWS?
y
n
Create /tftpboot/hostname-new-srvtab.
Create the srvtab file on the cws and copy it to the local /tftpboot directory.
authent_server = ssp?
y
Remove the srvtab file from the CWS.
n
srvtab file exists
y
Make the srvtab file read only for nobody (tftp)
chmod 400 /tftpboot/hostname-new-srvtab
chown nobody / tftpboot / hostname-new-srvtab
n
return

Figure 137. create_krb_files Wrapper Flowchart

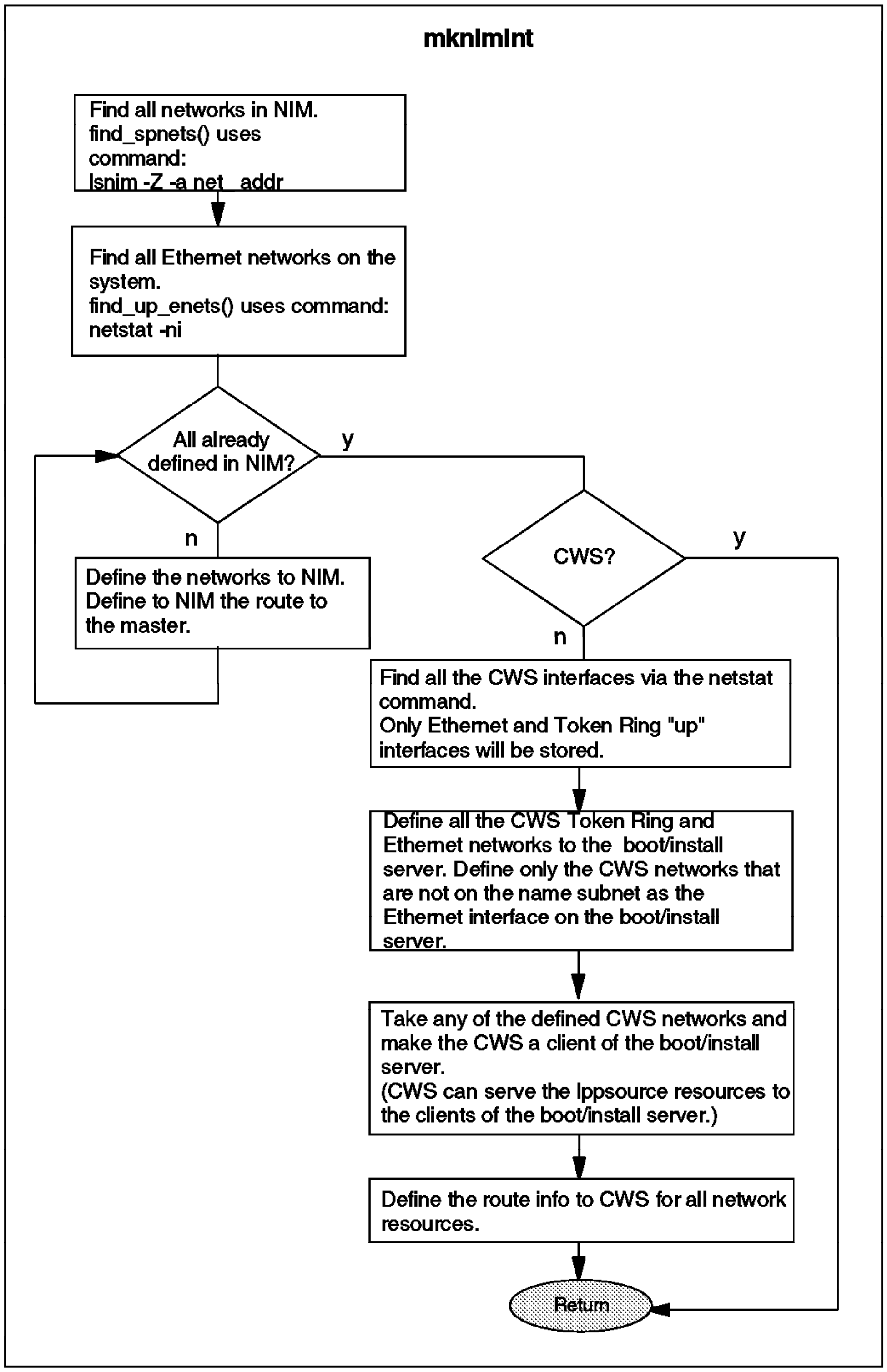

mknimint
Find all networks in NIM. find_spnets() uses command: lsnim -Z -a net_addr
Find all Ethernet networks on the system. find_up_enets() uses command: netstat -ni
All already defined in NIM?
y
n
Define the networks to NIM. Define to NIM the route to the master.
CWS?
y
n
Find all the CWS interfaces via the netstat command. Only Ethernet and Token Ring "up" interfaces will be stored.
Define all the CWS Token Ring and Ethernet networks to the boot/install server. Define only the CWS networks that are not on the name subnet as the Ethernet interface on the boot/install server.
Take any of the defined CWS networks and make the CWS a client of the boot/install server. (CWS can serve the lppsource resources to the clients of the boot/install server.)
Define the route info to CWS for all network resources.
Return

Figure 138. mknimint Wrapper Flowchart

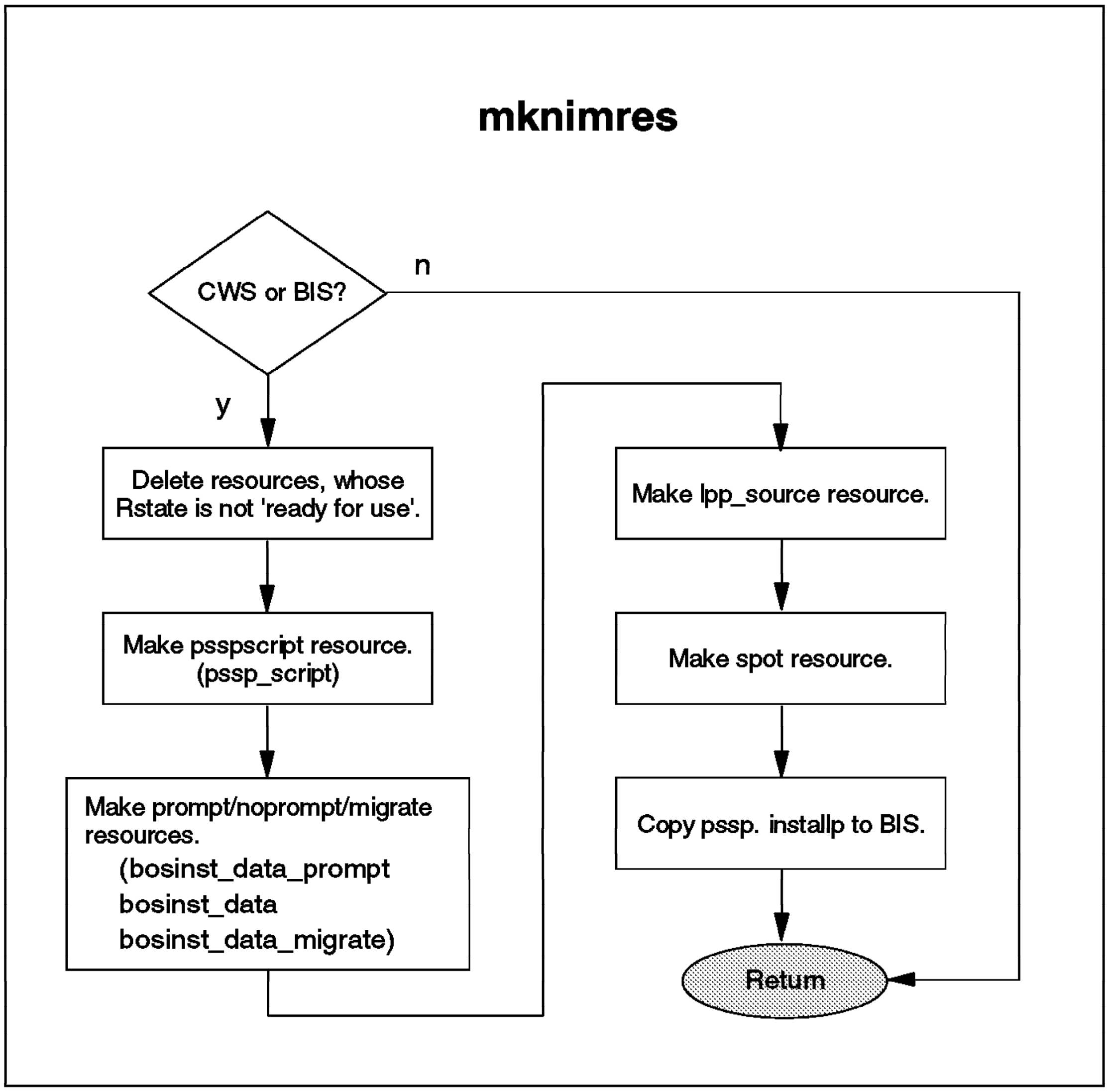

mknimres
CWS or BIS?
n
y
Delete resources, whose Rstate is not 'ready for use'.
Make psspscript resource. (pssp_script)
Make prompt/noprompt/migrate resources. (bosinst_data_prompt bosinst_data bosinst_data_migrate)
Make lpp_source resource.
Make spot resource.
Copy pssp. installp to BIS.
Return

Figure 139. mknimres Wrapper Flowchart

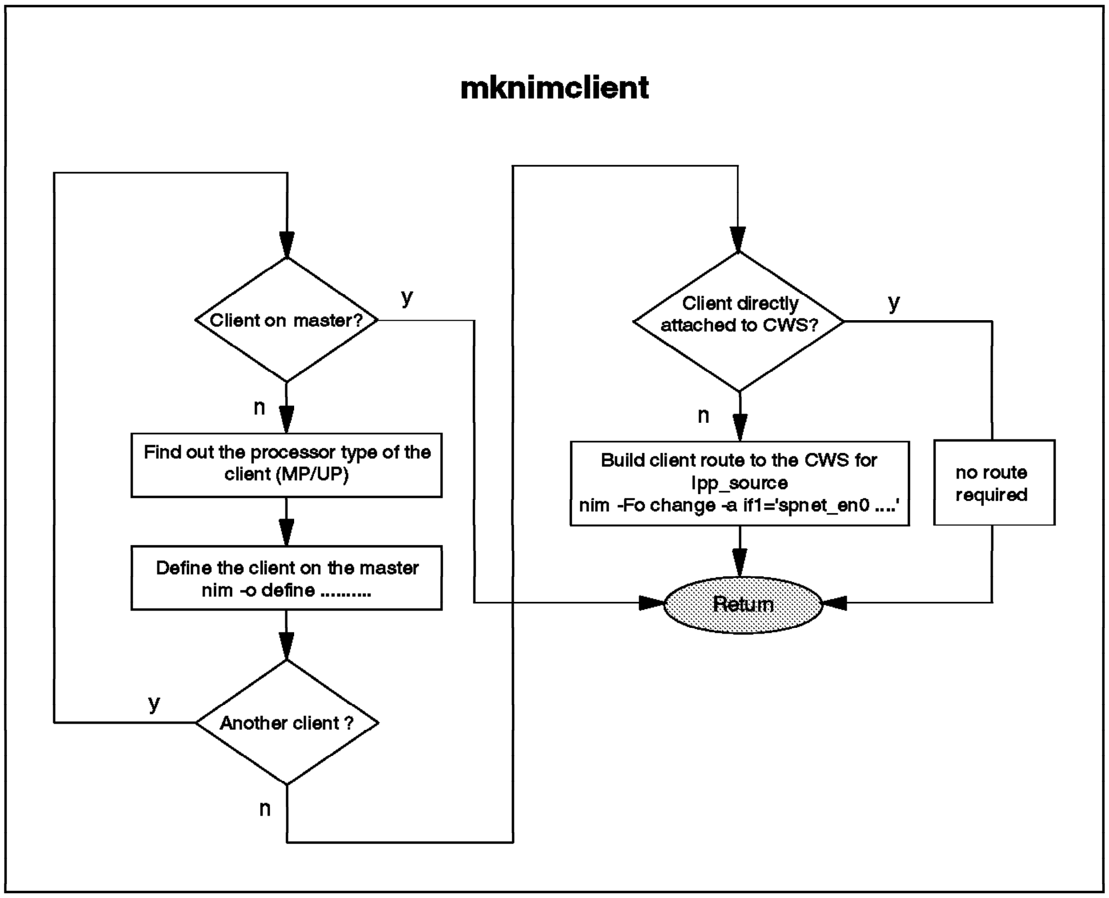

mknimclient
Client on master?
y
n
Find out the processor type of the client (MP/UP)
Define the client on the master nim -o define ....
Another client ?
y
n
Client directly attached to CWS?
y
n
Build client route to the CWS for lpp_source nim -Fo change -a if1='spnet_en0 ....'
no route required
Return

Figure 140. mknimclient Wrapper Flowchart

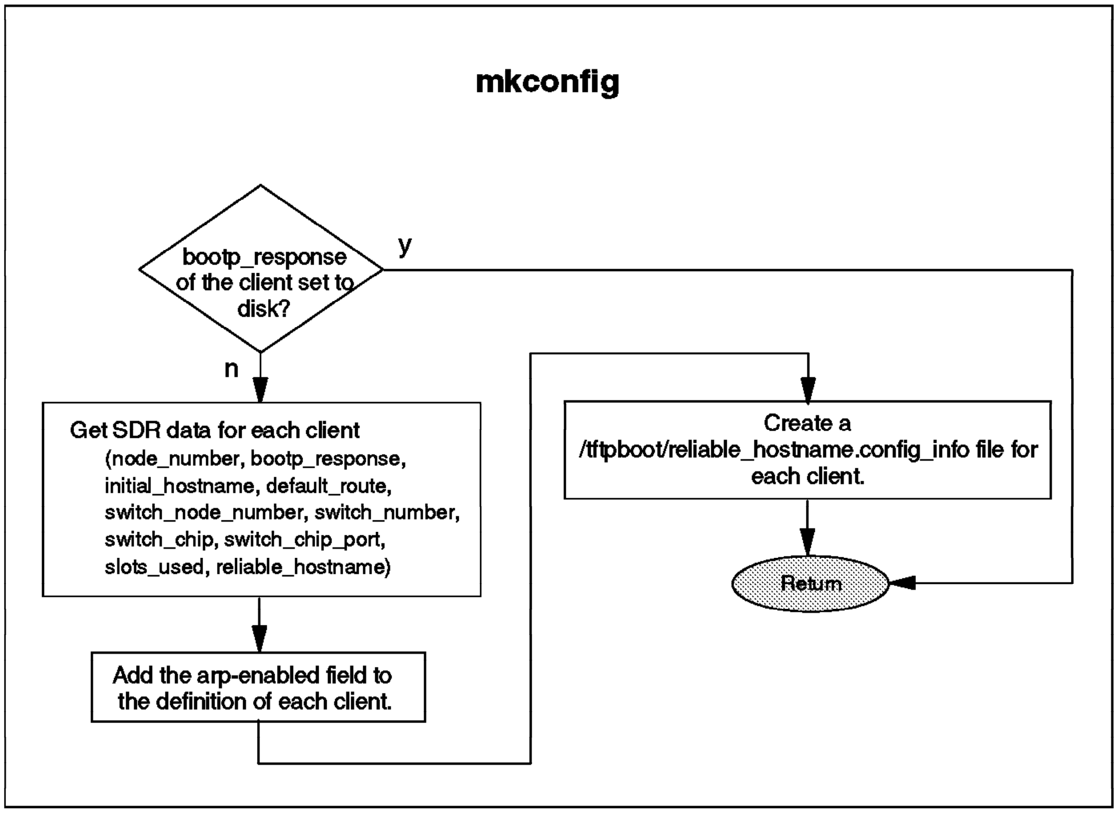

mkconfig
bootp_response of the client set to disk?
y
n
Get SDR data for each client (node_number, bootp_response, initial_hostname, default_route, switch_node_number, switch_number, switch_chip, switch_chip_port, slots_used, reliable_hostname)
Add the arp-enabled field to the definition of each client.
Create a /tftpboot/reliable_hostname.config_info file for each client.
Return

Figure 141. mkconfig Wrapper Flowchart

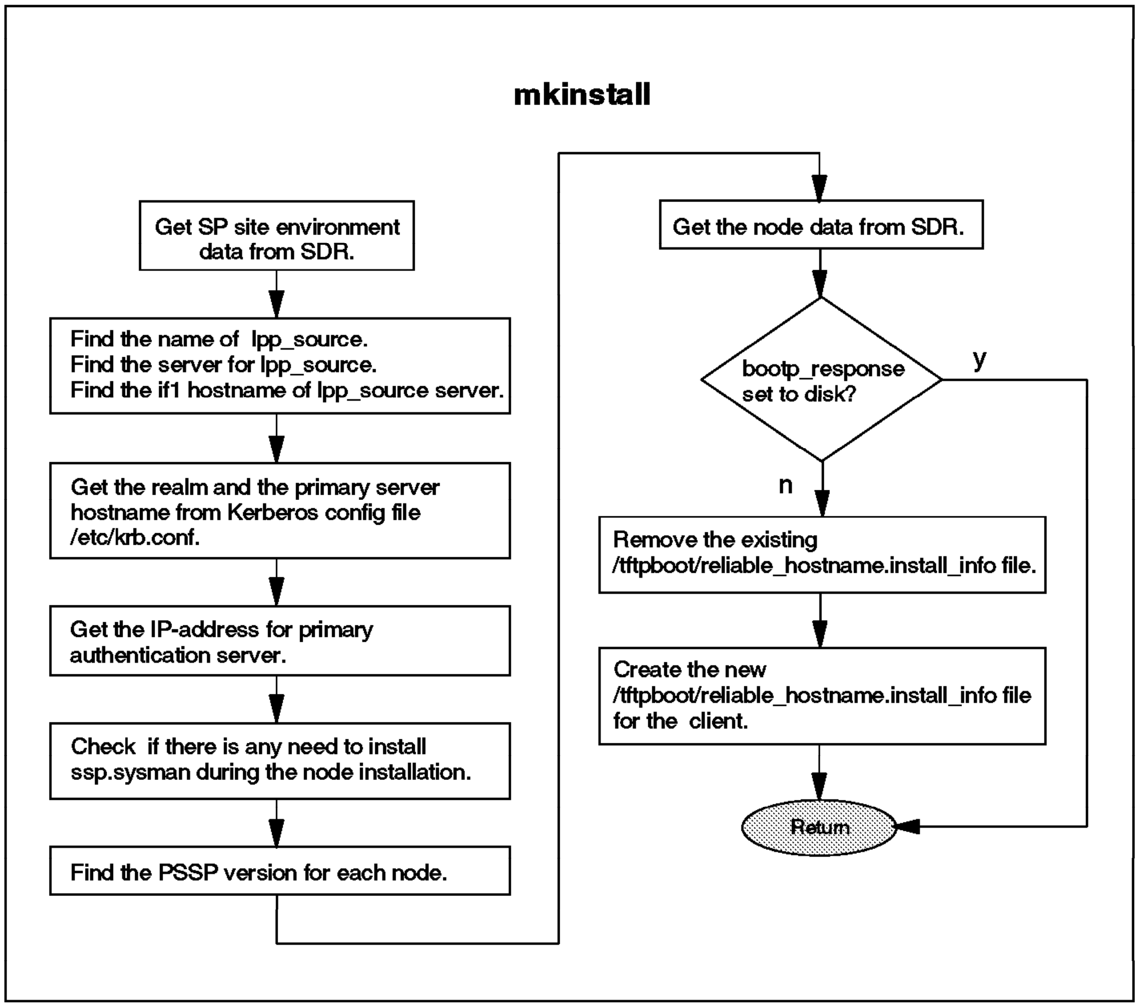

mkinstall
Get SP site environment data from SDR.
Find the name of lpp_source.
Find the server for lpp_source.
Find the if1 hostname of lpp_source server.
Get the realm and the primary server hostname from Kerberos config file /etc/krb.conf.
Get the IP-address for primary authentication server.
Check if there is any need to install ssp.sysman during the node installation.
Find the PSSP version for each node.
Get the node data from SDR.
bootp_response set to disk?
y
n
Remove the existing /tftpboot/reliable_hostname.install_info file.
Create the new /tftpboot/reliable_hostname.install_info file for the client.
Return

Figure 142. mkinstall Wrapper Flowchart

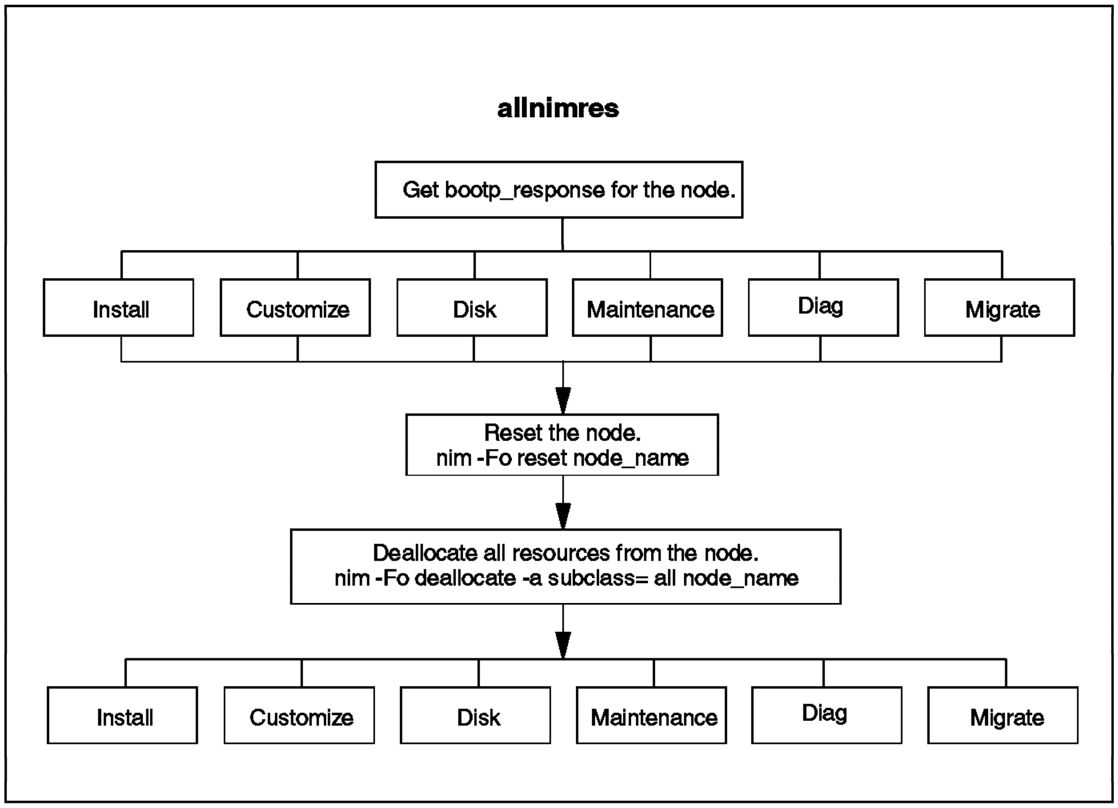


Figure 143. allnimres Wrapper (Part 1)

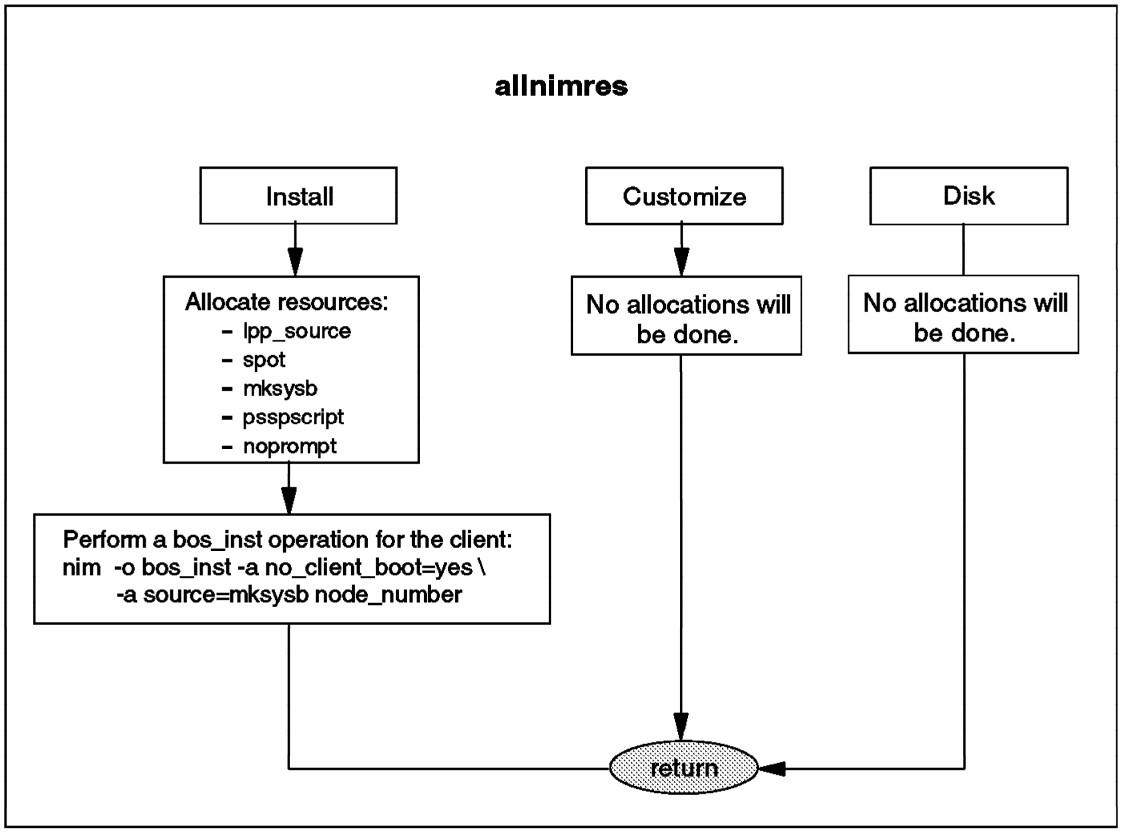

allnimres
Install
Customize
Disk
Allocate resources:
- lpp_source
- spot
- mksysb
- psspscript
- noprompt
No allocations will be done.
No allocations will be done.
Perform a bos_inst operation for the client:
nim -o bos_inst -a no_client_boot=yes \
-a source=mksysb node_number
return

Figure 144. allnimres Wrapper (Part 2)

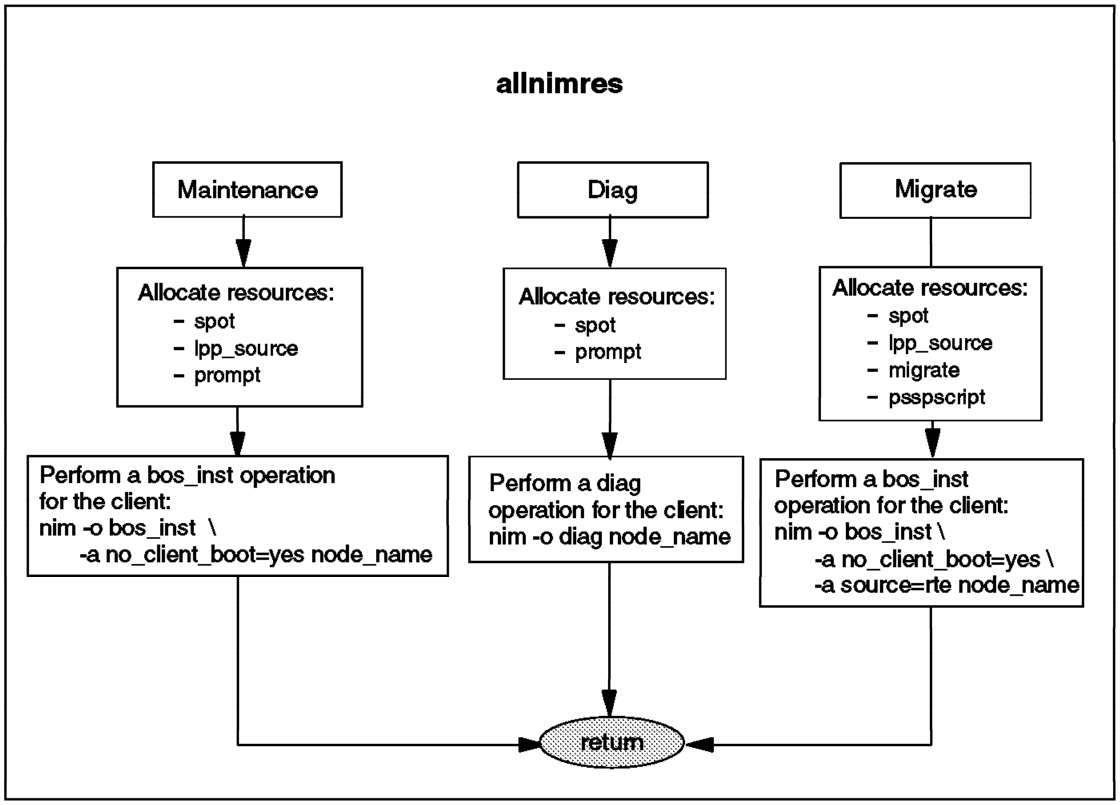


Figure 145. allnimres Wrapper (Part 3)

Appendix F. Updating the Microcode of Supervisor Cards

PSSP 2.2 can update supervisor card microcode, by command, for these three types of supervisor cards:

- Frame supervisor card
- Node supervisor card for high node
- SP Switch supervisor card

The “old” type of frame supervisor card, and node supervisor cards for wide and thin nodes, are not capable of downloading microcode through an RS-232C line.

The microcode source is included in the file set ssp.basic. After this LPP is installed, these files are located in the directory /spdata/sys1/ucode, as follows:

```
# ls /spdata/sys1/ucode
u_10.00.0604  u_10.1a.060c  u_10.3a.060c  u_80.01.0608  u_80.09.060b
u_10.00.0605  u_10.1a.060d  u_10.3a.060d  u_80.01.0609  u_80.11.060b
u_10.00.0606  u_10.1c.0606  u_10.3c.0606  u_80.01.060b  u_80.19.060b
u_10.00.0608  u_10.1c.0608  u_10.3c.0608  u_80.09.0608
u_10.00.060a  u_10.1c.060a  u_10.3c.060a  u_80.09.0609
```

Note that a PTF for ssp.basic may contain the latest microcode, so you should also download it to supervisor cards, as well as applying the PTF.

As described in step 29 of *PSSP 2.2 Installation and Migration Guide*, GC23-3898, checking and downloading microcode can be done by using the SMIT supervisor, as follows:

```
                              RS/6000 SP Supervisor Manager

Move cursor to desired item and press Enter.

  Check For Supervisors That Require Action (Single Message Issued)
  List Status of Supervisors (Report Form)
  List Status of Supervisors (Matrix Form)
  List Supervisors That Require Action (Report Form)
  List Supervisors That Require Action (Matrix Form)
  Update *ALL* Supervisors That Require Action (Use Most Current Level)
  Update Selectable Supervisors That Require Action (Use Most Current Lev










F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

This SMIT panel allows you to: check if any supervisor card requires action; list the current versions of microcodes installed on the supervisor cards and the versions of microcodes available in the/spdata/sys1/ucode directory; update microcodes to the latest version. In updating, nodes will be powered off, as in the case of getting the hardware address of the Ethernet adapter; therefore, you should shut down the node beforehand if it is running.

The command behind this SMIT panel is `spsvrmgr` and, as you may have noticed, in downloading microcodes it performs only a forward transition on the supervisor card′s microcode (that is, always to the most current level) and does not provide a way to reverse the transition. From the perspective of problem determination, it may help if you know how to download microcode levels that are not necessarily the latest levels (if, for example, files at /spdata/sys1/ucode are corrupted, or if the process of updating microcode is aborted prematurely).

In our case, however, we will simply download a current level of microcode.

For this purpose, we use `hmcmds` instead of `spsvrmgr`, and downloading of microcode will be done in two steps:

1. Bring supervisor card to its basecode level by entering:
   `# hmcmds -G -v basecode` *frame:slot*

   As described in the command reference, this command "performs a power off of the node and switches the active frame, node, or switch supervisor to basecode mode, causing the active supervisor to become nonactive and the basecode supervisor to become active." The option -v specifies verbose mode. The percentage of hardware components whose state matches the VFOP command is displayed at five-second intervals.

2. Perform a download of supervisor microcode to the frame, node, or switch by entering:
   `# hmcmds -G -v -u /spdata/sys1/ucode/`*filename* `microcode` *frame:slot*

The following describes what we experienced on our test SP system.

Our SP system had four high nodes on slots 1, 5, 9, and 13, and an SP Switch with a single partition. As soon as the PSSP 2.2 PTF Set 4 was available, we tried to download the latest microcodes included in ssp.basic.2.2.0.2.

First, we checked the status of supervisor cards by selecting **List Status of Supervisors (Report Form)** on the SMIT panel, as follows:

```
spsvrmgr: Frame  Slot  Supervisor  Media          Installed      Requ
                       State       Versions       Version        Action

          _____  ____  __________  _____________  _____________  __________
          1      0     Active      u_10.3c.0608   u_10.3c.060a   Upgrade
                                   u_10.3c.060a
                                   u_10.3c.060b

                 ____  __________  _____________  _____________  __________
                 1     Active      u_10.3a.060c   u_10.3a.0610   Upgrade
                                   u_10.3a.060d
                                   u_10.3a.0610
                                   u_10.3a.0612

                 ____  __________  _____________  _____________  __________
                 5     Active      u_10.3a.060c   u_10.3a.0610   Upgrade
                                   u_10.3a.060d
                                   u_10.3a.0610
                                   u_10.3a.0612

                 ____  __________  _____________  _____________  __________
                 9     Active      u_10.3a.060c   u_10.3a.0610   Upgrade
                                   u_10.3a.060d
                                   u_10.3a.0610
                                   u_10.3a.0612

                 ____  __________  _____________  _____________  __________
                 13    Active      u_10.3a.060c   u_10.3a.0610   Upgrade
                                   u_10.3a.060d
                                   u_10.3a.0610
                                   u_10.3a.0612

                 ____  __________  _____________  _____________  __________
                 17    Active      u_80.19.060b   u_80.19.060b   None
```

The frame supervisor card and SP Switch appear on slots 0 and 17, respectively, in the preceding output. It tells us to upgrade microcodes for the frame and four High nodes, which can be done by choosing **Update *ALL* Supervisors That Require Action (Use Most Current Level)**. As a result of doing this, we received error messages in the smit.log, as follows:

```
spsvrmgr: Dispatched "microcode" process [32558] for frame 1 slot 0.
          Process will take approximately 5 minutes to complete.
spsvrmgr: Process [32558] for frame 1 slot 0 completed successfully.

spsvrmgr: Dispatched "microcode" process [32612] for frame 1 slot 1.
          Process will take approximately 14 minutes to complete.

spsvrmgr: Dispatched "microcode" process [36968] for frame 1 slot 5.
          Process will take approximately 14 minutes to complete.

spsvrmgr: Dispatched "microcode" process [39020] for frame 1 slot 9.
          Process will take approximately 14 minutes to complete.

spsvrmgr: Dispatched "microcode" process [37232] for frame 1 slot 13.
          Process will take approximately 14 minutes to complete.

spsvrmgr: 0026-693 Process [36968] for frame 1 slot 5 failed, exit status(1).
          The contents of the generated log file follows:
hmcmds: Sent VFOP command "basecode" to 1 slots.


spsvrmgr: 0026-693 Process [37232] for frame 1 slot 13 failed, exit status(1).
          The contents of the generated log file follows:
hmcmds: Sent VFOP command "basecode" to 1 slots.


spsvrmgr: 0026-693 Process [39020] for frame 1 slot 9 failed, exit status(1).
          The contents of the generated log file follows:
hmcmds: Sent VFOP command "basecode" to 1 slots.


spsvrmgr: 0026-693 Process [32612] for frame 1 slot 1 failed, exit status(1).
          The contents of the generated log file follows:
hmcmds: Sent VFOP command "basecode" to 1 slots.
hmcmds: Application download begins to frame 1 node 1.
        Microcode file is "/spdata/sys1/ucode/u_10.3a.0612".
hmcmds: 0026-689 Frame supervisor not responding during microcode download.
hmcmds: The VFOP command "microcode" did not succeed for the following nodes.
        frame ID:   1  slot ID:  1
```

The failure in the microcode update seemed to disrupt RS-232C communications. The symptoms were the following:

- High nodes at slots 5, 9, and 13 disappeared.
- A ghost node appeared at slot 6, as follows:

```
[smpcws:/]:spmon -d
1.  Checking server process
    Process 18624 has accumulated 0 minutes and 26 seconds.
    Check ok

2.  Opening connection to server
    Connection opened
    Check ok

3.  Querying frame(s)
    1 frame(s)
    Check ok

4.  Checking frames

    This step was skipped because the -G flag was omitted.

5.  Checking nodes

--------------------------------- Frame 1 ------------------------------------
Frame  Node    Node         Host      Switch    Key     Env    Front Panel
Slot   Number  Type  Power  Responds  Responds  Switch  Fail      LEDs
------------------------------------------------------------------------------
                                                                ‾   ‾   ‾
  1       1    high   off     no        no      normal   no    |_| |_| |_|
                                                               |_| |_| |_|
                                                                ‾   ‾   ‾
  6       6    high   off     N/A       N/A     normal   no    |_| |_| |_|
                                                               |_| |_| |_|
[smpcws:/]:spsvrmgr -G -r status all

spsvrmgr: Frame  Slot  Supervisor  Media          Installed      Required
                       State       Versions       Version        Action
          _____  ____  __________  ____________   ____________   ____________
          1      0     Active      u_10.3c.0608   u_10.3c.060b   None
                                   u_10.3c.060a
                                   u_10.3c.060b
                 ____  __________  ____________   ____________   ____________
                 1     Active      u_10.3a.060c   u_10.3a.0612   None
                                   u_10.3a.060d
                                   u_10.3a.0610
                                   u_10.3a.0612
                 ____  __________  ____________   ____________   ____________
                 6     Inactive    None           u_10.11.5021   Update Media
                 ____  __________  ____________   ____________   ____________
                 17    Active      u_80.19.060b   u_80.19.060b   None
```

We tried the following corrective actions: stopping and starting hardmon, resetting the frame supervisor card by using `hmcmds -G runpost 1:0`, and re-issuing `spframe`, but none of these solved the problem.

At this point, we decided to go back to the previous microcode level, as shown:

```
[smpcws:/]:hmcmds -G -v basecode 1:0
hmcmds: Sent VFOP command "basecode" to 1 slots.
[smpcws:/]:hmcmds -G -v -u /spdata/sys1/ucode/u_10.3c.060a microcode 1:0
hmcmds: Application download begins to frame 1 node 0.
        Microcode file is "/spdata/sys1/ucode/u_10.3c.060a".
hmcmds: 25% complete. Elapsed time: 1 mins, 56 secs.
hmcmds: 50% complete. Elapsed time: 3 mins, 52 secs.
hmcmds: 75% complete. Elapsed time: 5 mins, 47 secs.
hmcmds: 100% complete. Elapsed time: 7 mins, 42 secs.
hmcmds: Sent VFOP command "clearq" to 1 slots.
hmcmds: Sent VFOP command "boot_supervisor" to 1 slots.
hmcmds: 0.00% complete.
hmcmds: 100.00% complete.
hmcmds: Number of slots expected to be in state "on": 1.
hmcmds: Number of slots currently in state "on": 1.
hmcmds: Sent VFOP command "setid" to 1 slots.
hmcmds: Sent VFOP command "microcode" to 1 slots.
[smpcws:/]:spmon -d
...

--------------------------------- Frame 1 ------------------------------------
Frame  Node    Node         Host      Switch    Key     Env    Front Panel
Slot   Number  Type  Power  Responds  Responds  Switch  Fail       LEDs
------------------------------------------------------------------------------

  1       1    high   off     no        no      normal   no    _   _   _


                                                                _   _   _
  5       5    high   off     no        no      normal   no   |_| |_| |_|
                                                              |_| |_| |_|
                                                                _   _   _
  6       6    high   off     N/A       N/A     normal   no   |_| |_| |_|
                                                              |_| |_| |_|
                                                                _   _   _
  9       9    high   off     no        no      normal   no   |_| |_| |_|
                                                              |_| |_| |_|
                                                                _   _   _
 13      13    high   off     no        no      normal   no   |_| |_| |_|
                                                              |_| |_| |_|
```

As you can see in the preceding output, the restoration of the frame supervisor card only resulted in the high node being recognized correctly via the RS-232C line, but a ghost node *still* remained at slot 6. Therefore, we continued to work on the nodes by first bringing them to basecode level, and then downloading microcode that had been running before.

```
[smpcws:/]:hmcmds -G -v basecode 1:1,5,9,13
hmcmds: Sent VFOP command "basecode" to 4 slots.
[smpcws:/]:spmon -d
1.  Checking server process
    Process 18586 has accumulated 0 minutes and 34 seconds.
    Check ok

2.  Opening connection to server
    Connection opened
    Check ok

3.  Querying frame(s)
    1 frame(s)
    Check ok

4.  Checking frames

    This step was skipped because the -G flag was omitted.

5.  Checking nodes

--------------------------------- Frame 1 ------------------------------------
Frame  Node         Clock Env
Slot   Type  Power  Input Fail
------------------------------------------------------------------------------
  1    high   off     0    no
  5    high   off     0    no
  6    swit    on     1   yes
  9    high   off     0    no
 13    high   off     0    no
```

Note that, at basecode level, the ghost node is recognized as a switch. We will discuss how to deal with this problem later in this section.

```
[smpcws:/]:hmcmds -G -v -u /spdata/sys1/ucode/u_10.3a.0610 microcode 1:1
hmcmds: Application download begins to frame 1 node 1.
        Microcode file is "/spdata/sys1/ucode/u_10.3a.0610".
hmcmds: 25% complete. Elapsed time: 4 mins, 37 secs.
hmcmds: 50% complete. Elapsed time: 9 mins, 12 secs.
hmcmds: 75% complete. Elapsed time: 13 mins, 50 secs.
hmcmds: 100% complete. Elapsed time: 18 mins, 29 secs.
hmcmds: Sent VFOP command "boot_supervisor" to 1 slots.
hmcmds: 0.00% complete.
hmcmds: 100.00% complete.
hmcmds: Number of slots expected to be in state "on": 1.
hmcmds: Number of slots currently in state "on": 1.
hmcmds: Sent VFOP command "microcode" to 1 slots.
[smpcws:/]:spmon -d
...

--------------------------------- Frame 1 ------------------------------------
Frame  Node         Clock Env
Slot   Type  Power  Input Fail
------------------------------------------------------------------------------
  1    high    on     0    no
  5    high   off     0    no
  6    swit    on     1   yes
  9    high   off     0    no
 13    high   off     0    no
[smpcws:/]:spsvrmgr -G -r status all

spsvrmgr: Frame  Slot  Supervisor  Media        Installed     Required
                       State       Versions     Version       Action
          _____  ____  __________  ____________ ____________  ____________
          1      0     Active      u_10.3c.0608 u_10.3c.060a  Upgrade
                                   u_10.3c.060a
                                   u_10.3c.060b
                 ____  __________  ____________ ____________  ____________
                 1     Active      u_10.3a.060c u_10.3a.0610  Upgrade
                                   u_10.3a.060d
                                   u_10.3a.0610
                                   u_10.3a.0612
                 ____  __________  ____________ ____________  ____________
                 5     Inactive    u_10.3a.060c u_10.3a.0610  Upgrade
                                   u_10.3a.060d
                                   u_10.3a.0610
                                   u_10.3a.0612
                 ____  __________  ____________ ____________  ____________
                 9     Inactive    u_10.3a.060c u_10.3a.0610  Upgrade
                                   u_10.3a.060d
                                   u_10.3a.0610
                                   u_10.3a.0612
                 ____  __________  ____________ ____________  ____________
                 13    Inactive    u_10.3a.060c u_10.3a.0610  Upgrade
                                   u_10.3a.060d
                                   u_10.3a.0610
                                   u_10.3a.0612
                 ____  __________  ____________ ____________  ____________
                 17    Active      u_80.19.060b u_80.19.060b  None
```

```
[smpcws:/]:hmcmds -G -v -u /spdata/sys1/ucode/u_10.3a.0610 microcode 1:5,9,13
hmcmds: Application download begins to frame 1 node 5.
        Microcode file is "/spdata/sys1/ucode/u_10.3a.0610".
...

[smpcws:/]:spmon -d
...

-------------------------------- Frame 1 ----------------------------------
Frame  Node    Node         Host      Switch    Key     Env    Front Panel
Slot   Number  Type  Power  Responds  Responds  Switch  Fail      LEDs
---------------------------------------------------------------------------
  1       1    high    on      yes       no     normal   no    LEDs are blank
  5       5    high    on      yes       no     normal   no    LEDs are blank
  9       9    high    on      yes       no     normal   no    LEDs are blank
 13      13    high    on      yes       no     normal   no    LEDs are blank
[smpcws:/]:spsvrmgr -G -r status all

spsvrmgr: Frame  Slot  Supervisor  Media          Installed      Required
                       State       Versions       Version        Action
          _____  ____  __________  ____________  ____________  ____________
          1      0     Active      u_10.3c.0608  u_10.3c.060a  Upgrade
                                   u_10.3c.060a
                                   u_10.3c.060b
                 ____  __________  ____________  ____________  ____________
                 1     Active      u_10.3a.060c  u_10.3a.0610  Upgrade
                                   u_10.3a.060d
                                   u_10.3a.0610
                                   u_10.3a.0612
                 ____  __________  ____________  ____________  ____________
                 5     Active      u_10.3a.060c  u_10.3a.0610  Upgrade
                                   u_10.3a.060d
                                   u_10.3a.0610
                                   u_10.3a.0612
                 ____  __________  ____________  ____________  ____________
                 9     Active      u_10.3a.060c  u_10.3a.0610  Upgrade
                                   u_10.3a.060d
                                   u_10.3a.0610
                                   u_10.3a.0612
                 ____  __________  ____________  ____________  ____________
                 13    Active      u_10.3a.060c  u_10.3a.0610  Upgrade
                                   u_10.3a.060d
                                   u_10.3a.0610
                                   u_10.3a.0612
                 ____  __________  ____________  ____________  ____________
                 17    Active      u_80.19.060b  u_80.19.060b  None
```

After all nodes were brought back to where they were previously, we checked to see if there was a problem with the switch, as follows:

```
[smpcws:/]:splstdata -s
                  List Node Switch Information

                        switch  switch  switch switch    switch
node# initial_hostname   node# protocol number   chip chip_port
----------------------------------------------------------------
    1 smp01e                 0       IP      1      5         3
    5 smp02e                 4       IP      1      5         1
    9 smp03e                 8       IP      1      4         3
   13 smp04e                12       IP      1      4         1

switch   frame    slot  switch_partition  switch  clock
number  number  number            number    type  input
----------------------------------------------------------------
     1       1      17                 1     129      0
  1001       1       6                 1      49      0

switch_part           topology           primary        arp  switch_node
     number           filename              name    enabled    nos._used
-------------------------------------------------------------------------
          1   expected.top.an  none                     yes           no
```

As expected, there is an erroneous entry (switch number=1001, slot number=6) in the switch information; this is the ghost node. We deleted this entry from the SDR, as follows:

```
[smpcws:/]:SDRGetObjects Switch
switch_number frame_number slot_number  switch_partition_number switch_type  cl
          1            1          17             1          129 0
        1001           1           6             1           49 0            "
[smpcws:/]:SDRDeleteObjects Switch switch_number==1001
[smpcws:/]:SDRGetObjects Switch
switch_number frame_number slot_number  switch_partition_number switch_type  cl
          1            1          17             1          129 0
[smpcws:/]:splstdata -s
                  List Node Switch Information

                        switch  switch  switch switch    switch
node# initial_hostname   node# protocol number   chip chip_port
----------------------------------------------------------------
    1 smp01e                 0       IP      1      5         3
    5 smp02e                 4       IP      1      5         1
    9 smp03e                 8       IP      1      4         3
   13 smp04e                12       IP      1      4         1

switch   frame    slot  switch_partition  switch  clock
number  number  number            number    type  input
----------------------------------------------------------------
     1       1      17                 1     129      0

switch_part           topology           primary        arp  switch_node
     number           filename              name    enabled    nos._used
-------------------------------------------------------------------------
          1   expected.top.an  none                     yes           no
```

The lesson we learned is that, indeed, there may be cases where we need to download microcode selectively, and that problems with microcodes may affect other parts of the system, such as the SDR.

Appendix G. Special Notices

This publication is intended to help IBM customers, Business Partners, IBM System Engineers and other RS/6000 SP specialists who are involved in POWERparallel System Support Programs Version 2.2 projects, including education of RS/6000 SP professionals responsible for installing, configuring, and administering PSSP Version 2 Release 2. The information in this publication is not intended as the specification of any programming interfaces that are provided by POWERparallel System Support Programs. See the PUBLICATIONS section of the IBM Programming Announcement for POWERparallel System Support Programs for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Appendix H. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

H.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see "How to Get ITSO Redbooks" on page 313.

- *RS/6000 SP PSSP 2.2 Technical Presentation*, SG24-4868
- *RS/6000 SP High Availability Infrastructure*, SG24-4838
- *RS/6000 SP Monitoring: Keeping it Alive*, SG24-4873
- *RS/6000 SP: Problem Determination Guide*, SG24-4778

H.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

H.3 Other Publications

These publications are also relevant as further information sources:

- *RS/6000 SP: Administration Guide*, GC23-3897
- *RS/6000 SP: Installation and Migration Guide*, GC23-3898
- *RS/6000 SP: Diagnosis and Messages Guide*, GC23-3899
- *RS/6000 SP: Command and Technical Reference*, GC23-3900
- *RS/6000 SP: System Planning Guide*, GC23-3902
- *RS/6000 SP: Event Management Programming Guide and Reference*, SC23-3996
- *RS/6000 SP: Group Services Programming Guide and Reference*, SC28-1675
- *AIX V4.1 Problem Solving Guide and Reference*, SC23-2606

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL `http://www.redbooks.ibm.com`.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type `GOPHER.WTSCPOK.ITSO.IBM.COM`
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

  `http://www.redbooks.ibm.com`

- **IBM Direct Publications Catalog on the World Wide Web**

  `http://www.elink.ibmlink.ibm.com/pbl/pbl`

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to `announce@webster.ibmlink.ibm.com` with the keyword `subscribe` in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
(+45) 48 14 2207 (long distance charge)	Outside North America

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com

- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number

- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

List of Abbreviations

ACL	Access Control List
AIX	Advanced Interactive Executive
AMG	Adapter Membership Group
ANS	Abstract Notation Syntax
APA	all points addressable
API	Application Programming Interface
BIS	Boot-Install Server
BSD	Berkeley Software Distribution
BUMP	Bring-Up Microprocessor
CP	Crown Prince
CPU	Central Processing Unit
CSS	Communication Subsystem
CWS	Control Workstation
EM	Event Management
EMAPI	Event Management Application Programming Interface
EMCDB	Event Management Configuration Database
EMD	Event Manager Daemon
EPROM	Erasable Programmable Read Only Memory
FIFO	First In - First Out
Gb	Gigabytes
GL	Group Leader
GS	Group Services
GSAPI	Group Services Application Programming Interface
hb	heart beat
HPS	High Performance Switch
hrd	host respond daemon
HSD	Hashed Shared Disk
IBM	International Business Machines Corporation
IP	Internet Protocol
ISB	Intermediate Switch Board
ISC	Intermediate Switch Chip
ITSO	International Technical Support Organization
JFS	Journal File System
LAN	Local Area Network
LCD	Liquid Crystal Display
LED	Light Emitter Diode
LRU	Least Recently Used
LSC	Link Switch Chip
LVM	Logical Volume Manager
Mb	megabytes
MIB	Management Information Base
MPI	Message Passing Interface
MPL	Message Passing Library
MPP	Massively Parallel Processors
NIM	Network Installation Manager
NSB	Node Switch Board
NSC	Node Switch Chip
OID	Object ID
ODM	Object Data Manager
PE	Parallel Environment
PID	Process ID
PROFS	Professional Office System
PSSP	Parallel System Support Program
PTC	Prepare to Commit
PTPE	Performance Toolbox Parallel Extensions
PTX/6000	Performance Toolbox/6000
RAM	Random Access Memory
RCP	Remote Copy Protocol
RM	Resource Monitor
RMAPI	Resource Monitor Application Programming Interface
RPQ	Request for Product Quotation
RSI	Remote Statistics Interface
RVSD	Recoverable Virtual Shared Disk
SBS	Structured Byte String
SDR	System Data Repository
SMP	Symmetric Multiprocessors
SNMP	System Network Management Protocol
SPDM	SP Data Manager

SPMI	System Performance Measurement Interface
SRC	System Resource Controller
SSI	Single System Image
TS	Topology Services
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
VSD	Virtual Shared Disk
VSM	Visual System Management

Index

ITSO Redbook Evaluation

RS/6000 SP PSSP 2.2 Survival Guide
SG24-4928-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____________

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

____

____

____

____

What other redbooks would you like to see published?

____

____

____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

____

____

____

____

____

Printed in U.S.A.

SG24-4928-00