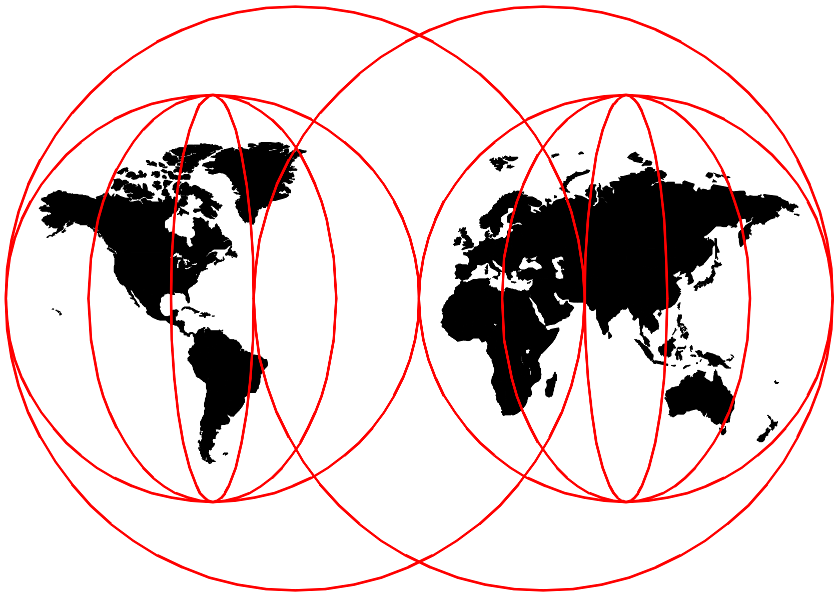
PSSP Version 3 Survival Guide

Marcelo Barrios, Paulo Aguiar, Shafkat Rabbi



International Technical Support Organization

www.redbooks.ibm.com

SG24-5344-00

SG24-5344-00

International Technical Support Organization

PSSP Version 3 Survival Guide

January 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix F, “Special notices” on page 441.

First Edition (January 2000)

This edition applies to PSSP Version 3, Release 1 (5765-D51) or later for use with the AIX Version 4, Release 3 Operating System.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Preface

Problem determination on the RS/6000 SP is part of the daily tasks a system administrator has to carry out. Although the RS/6000 SP is a very stable platform, the way stand-alone, distributed, and parallel environments are intermixed may cause certain problems that are not easy to solve for someone with only AIX experience.

This redbook gives a comprehensive explanation of certain RS/6000 SP components and provides the reader with tools and procedures that can be used for problem isolation and problem solving.

The book is oriented to RS/6000 SP professionals who install, configure, and administer SP systems. Several procedures are outlined and tested along explanation for the causes of common SP problems.

This guide for survival should be the perfect companion for the RS/6000 SP product manuals when it comes to identifying and solving system problems. The book does not guarantee success, but it certainly takes you one step closer.

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, Austin Center.

Marcelo R. Barrios is a project leader at the International Technical Support Organization, Poughkeepsie Center. He has been with IBM for five years working in different areas related to RS/6000. Currently, he focuses on RS/6000 SP technology by writing redbooks and teaching IBM classes worldwide.

Paulo Aguiar is an IT specialist at IBM Portugal. He joined IBM in 1990 and has been working with AIX ever since. He has been working with RS/6000 SP since 1994, supporting the large SP installations in Portugal. He has a master’s degree in mathematics from the University of Porto, Portugal.

Shafkat Rabbi is an IT specialist at IBM Canada. He joined the Metro Toronto Services Specialist team in the beginning of 1997. The Metro Services Specialist team is responsible for providing on-site support to large commercial customers in Toronto. He has been working with the RS/6000 SP since he joined the team. His current responsibility is to provide support for

SP to the third largest bank in Canada. He started his IBM career in IBM Bangladesh. He holds a bachelor's degree in electrical engineering.

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 "IBM Redbooks evaluation" on page 461 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Chapter 1. Problem determination

You are reading this book either because you have a problem you want to solve, or because you are interested in tools and techniques you can apply to your system.

If you are looking for solutions to particular problems, you may go directly to the section that discusses the component that is failing in your case. The chapters are basically self-contained so that you can use each independently.

Doing problem determination on the RS/6000 SP is not an easy task. This is not because the SP is so complex, but because the components involved in a problem may come from different farms, such as AIX, PSSP, RS/6000 firmware, or from a hardware malfunction. Identification is the key.

This chapter offers ideas about how to approach RS/6000 SP problems that, in fact, should apply to any computer-related problem. Later, we present some tools you may use to prevent most of the problems discussed in this book (80 percent of RS/6000 SP problems are caused by misconfigurations and/or lack of prevention).

Let's start with some concepts concerning problem determination.

1.1 Methodologies

The key to problem determination is to understand the symptom and identify the cause of that symptom.

First, do not trust the symptom. Different problems may have the "same" external effect. For example, think of when host responds is red. The symptom is the same, but the cause may have multiple variations (refer to Appendix D, "43 reasons why host responds is red" on page 429). The important thing is to approach the problem in a systematic way so that the time to isolate it to a single component is shortened as much as possible.

Second, you should consider if the problem may be reported. For example, you cannot solve the problem so it needs to be reported, but you have been trying to solve it for quite a while; therefore, the log files and state information is completely different now. You have contaminated the scene.

Third, check whether this problem has happened before. If yes, then the solution should be documented somewhere. Keep good documentation about the findings and solutions of your system problems.

Fourth, do some basic checking to roughly identify the cause. This may allow you to determine whether the problem is general to the entire system, or is isolated to a well-defined component. For example, you are having problems with host responds. Does it occur on all the nodes? Maybe only one node has the problem, so you have already isolated the problem to a single node. Easy, isn't it?

Unfortunately, real life is a little bit more complicated, so the problems you may face may be much more complex than the ones you see here. However, the techniques are always the same.

Let's review some tools, available on the RS/6000 SP, that can help you prevent many system problems.

1.2 Diagnostic tools

Information is everything. This is not new. However, it is particularly true when it comes to problem determination (and maybe the stock market). By having the right tools, your diagnosis can be more accurate. Fortunately, RS/6000 SP provides several tools that facilitate problem determination. We analyze some of these tools in this section.

What is a diagnostic tool? Is it a kind of device that will tell me what is wrong with my system? Well, essentially yes, because that is the purpose of the tool. However, the tool is not part of the solution, which makes the person behind the tool a critical resource.

Let's say that you have the host responds problem isolated to a single node. A tool could tell you that the Topology Services daemon is not running on that node. So you start it, or the tool may do it, and there you go, host responds is back to normal. You knew that Topology Services needs to be running in order to get host responds for that node. But what if Topology Services dies again? The tool will tell you that the daemon is not running. But now you know that even though that is the problem, it is certainly not the cause.

Having good diagnostic tools makes problem determination easier. The solutions, however, may be a bit more complicated.

Before discussing the tools, be aware that the RS/6000 SP team has put a lot of effort in developing an excellent diagnostic guide. Actually, there are two books that you should have with you (along with this one): the *PSSP: Diagnosis Guide*, GA22-7350, and the *PSSP: Messages Reference*, GA22-7352.

1.2.1 AIX service aids

Basically, every node (and the Control Workstation) is an AIX machine. This means that all the problem determination tools available for standard RS/6000 machines are also available for SP nodes and Control Workstations (CWS).

AIX provides facilities and tools for error logging, system trace, and system dumping (creation and analysis). Most of these facilities are included in the bos.rte fileset within AIX and, therefore, installed on every node and CWS automatically. However, some additional facilities and tools are included in an optionally installable package called bos.sysmgt.serv_aid, which should be installed on your nodes and CWS.

1.2.1.1 Error logging facility

The AIX error logging facility records hardware and software failures or informational messages in the error log. All of the AIX and PSSP subsystems will use this facility to log error messages or information about changes to state information.

By analyzing this log, you can get an idea of what went wrong, when, and possibly why. However, due to the way information is presented by the `errpt` command, it is difficult to correlate errors within a single machine, and much harder in the SP where errors could be caused by components on different machines. We will get back to this point later in this chapter.

The errdemon daemon keeps the log file updated based on information and errors logged by subsystems through the *errlog* facility, or through the *errsave* facility if they are running at kernel level. In any case, the errdemon daemon adds the entries in the error log on a first-come-first-served basis.

This error log facility also provides a mechanism through which you could create a notification object for specific log entries. You could instruct the errdemon daemon to send you an e-mail every time there is a hardware error. The section “Using the AIX Error Log Notification Facility” on page 72 of the *PSSP: Diagnosis Guide*, GA22-7350, provides excellent examples of setting up notification methods.

Log analysis is not bad. However, log monitoring is much better. You do not really want to check the error log on every node in your 128-node installation, right? Probably what you do is to create some notification objects in your nodes to instruct the errdemon daemon on those nodes to notify you in case any critical error gets logged into the error log.

PSSP provides facilities for log monitoring and error notification. This differs from AIX notification in the sense that, although it uses the AIX notification methods, it provides a global view of your system, so you could, for example, create a monitor for your AIX error log on all your nodes at once with a single command, or a few clicks.

1.2.1.2 Trace facility
A trace facility is available through AIX. However, it comes in an optional file set called bos.sysmgt.trace. Although the base system (bos.rte) includes minimal services for trace, you need to install this optional component if you want to activate the trace daemon and generate trace reports.

If you get to the point where trace is needed, it is probably because all the conventional methods have failed. Tracing is a serious business: It involves commitment and dedication to understand the trace report.

Tracing works basically in a 2-step mode: You turn on trace on selected subsystems and/or calls and then you analyze the trace file through report tools.

The events that can be included or excluded from the tracing facility are listed in the /usr/include/sys/trchkid.h header file. They are called hooks and subhooks. With these hooks, you can tell the tracing facility which specific event you want to trace. For example, you could generate a trace for all CREAT calls, which includes file creation.

To learn more about tracing, refer to Chapter 11, “Trace Facility” of the *AIX V4.3 Problem Solving Guide and Reference*, SC23-4123.

1.2.1.3 System dump facility
Yes, sometimes you have to use this. AIX generates a system dump when a severe error occurs. A system dump can also be user-initiated by users with root authority. It creates a picture of your system’s memory contents.

A system dump could save lives. Well, maybe not, but it certainly can help a lot in determining “who” took the machine out of order. A good system dump in the right hands can point to the guilty component.

Keep in mind that the system dump is a copy of selected areas of the kernel. These areas contain information about the processes and routines running at the moment of the crash. However, while it is easier for the operating system to keep this information in memory-address format, is not okay for a human. Therefore, for a good system dump analysis you need the table of symbols, which can be obtained from the operating system executable (/unix). Always save your system dumps along with the /unix corresponding to the operating system executable where the dump was produced. Support people will thank you.

For more information on AIX system dumps, refer to Chapter 12, “System Dump Facility” on page 81 of the *AIX V4.3 Problem Solving Guide and Reference*, SC23-4123.

1.2.2 PSSP service aids

The facilities that PSSP provides range from log files on every node and the CWS, to SP Perspectives, which utilizes the RS/6000 Cluster Technology discussed on Chapter 4, “RS/6000 Cluster Technology (RSCT)” on page 191.

1.2.2.1 SP log files

Besides errors and information being logged into the AIX error log, most of the PSSP subsystems write to their own log files where, usually, you will find the information you need for problem isolation and problem determination.

Since some components run only on the CWS (such as the SDR daemon, the host respond daemon, the switch admin daemon, and so on), others run only on nodes (such as the switch daemon). This needs to be taken into consideration in the search for logs. The *PSSP Diagnosis Guide*, GA22-7350 contains a complete list of PSSP log files and their location.

Unfortunately, there is no common rule for analyzing log files. They are very specific to each component and, in most cases, are created as internal debugging mechanisms and not for public consumption.

In this book, we cover some of these log files and explain how to read them. However, that information may be obsolete for the next release of PSSP. While the only “official” logging information is the AIX error log, nothing stops you from read those other log files. As a matter of fact, they are sometimes essential for problem determination.

All the PSSP log files are located in the /var/adm/SPlogs directory. All the RSCT log files are located in the /var/ha/log directory. So, considering that these two locations reside on the /var file system, make sure you have enough free space for holding all the logged information. Refer to the *IBM RS/6000 SP: Planning, Volume 2, Control Workstation and Software Environment*, GA22-7281, for details on disk space requirements.

1.2.2.2 Problem Management subsystem

The Problem Management (PMAN) subsystem is a facility, present on systems running PSSP v2.2 or later, for problem determination, problem notification, and problem solving. It uses the RSCT infrastructure for monitoring conditions for authorized users and then generates actions accordingly.

The PMAN subsystem consists of three components:

- **pmand** - This daemon interfaces directly with the Event Manager daemon (explained in detail in 4.3, “Event Management” on page 246) to register conditions and to receive notifications. This daemon runs on every node and the CWS and is partition-sensitive (the CWS may have more than one daemon running in case of multiple partitions).
- **pmanrmd** - This is a resource monitor provided by PMAN to “feed” Event Management with additional 16 user-defined variables. You can program this resource monitor to periodically run a command or execute a script to update one of these variables. Refer to “Monitoring a log file” on page 7 for an example on how to use this facility.
- **sp_configd** - With this daemon, PMAN can send simple network management protocol (SNMP) traps to SNMP managers to report predefined conditions.

You interface with PMAN through the command line interface. The command is called `pmandef`, and its use is restricted to authorized users. A user needs to have a Kerberos principal, and this principal needs to be listed in the /etc/sysctl.pman.acl file. The following is an example of this file:

```
#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 root.admin@PPD.POK.IBM.COM
#_PRINCIPAL joeuser@PPD.POK.IBM.COM
```

In this case, the Kerberos principal called `root.admin` at the `MSC.ITSO.IBM.COM` realm is authorized to use the PMAN subsystem on this node. Make sure you have the same on every node where you want to use this facility.

Make sure you stop and start the Sysctl SRC subsystem every time you modify this file.

The `pmandef` command has a very particular syntax, so if you want to give it a try, take a look at the *PSSP: Command and Technical Reference, Volume 1*, SA22-7351, page 350, for a complete definition of this command. Chapter 25, "Using the Problem Management Subsystem" in the *PSSP: Administration Guide*, SA22-7348, contains several examples and a complete explanation of how to use this facility, including the CWS.

Finally, the /usr/lpp/ssp/install/bin/pmandefaults script is an excellent starting point for using the PMAN subsystem. It has several examples of monitors for daemons, log files, file systems, and so forth.

Monitoring a log file

Now, we know that the PMAN subsystem provides 16 resource variables for user-defined events. In this section, we use one of these variables to monitor a specific condition where PSSP does not provide a resource variable.

Let's assume that you want to get a notification on the console screen each time there is an authentication failure for remote execution. We know that the remote shell daemon (rshd) logs these errors to the /var/adm/SPlogs/SPdaemon.log, so we can create a monitor for this specific error.

First, we need to identify the error that gets logged into this file every time somebody tries to execute a remote shell command without the proper credentials. Let's try this and observe the error log file:

```
Feb 27 14:30:16 sp3n01 rshd[17144]: Failed krb5_compat_recvauth
Feb 27 14:30:16 sp3n01 rshd[17144]: Authentication failed from
sp3en0.msc.itso.ibm.com: A connection is ended by software.
```

From this, we see that `Authentication failed` seems to be a nice string to look for. So, the idea here is to notify the operator (console) that there was a failed attempt to access this machine through the remote shell daemon.

Now there is a small problem to solve. If we are going to check this log file every few minutes, how do we know whether the log entry is new or was already reported? Fortunately, the way user-defined resource variables work is based on strings. The standard output of the script you associate with a user-defined resource variable is stored as the value of that variable. This means that if we print out the last Authentication failed entry every time, the variable value will change only when there is a new entry in the log file.

Let’s create the definition for a user-defined variable. To do this, PMAN needs a configuration file that has to be loaded to the SDR by using the `pmanrmdloadSDR` command.

PSSP provides a template for this configuration file. It is located in the /spdata/sys1/pman directory on the CWS. Let’s make a copy of this file and edit it:

```
TargetType=NODE_RANGE
Target=0-5
Rvar=IBM.PSSP.pm.User_state1
SampInt=60
Command=/usr/local/bin/Guard.pl
```

In this file, you can define all 16 user-defined variables (there must be one stanza per variable). In this case, we have defined the IBM.PSSP.pm.User_state1 resource variable. The resource monitor (pmanrmd) will update this variable every 60 seconds as specified in the sample interval (SampInt). The value of the variable will correspond to the standard output of the /usr/local/bin/Guard.pl script. Let’s see what the script does:

```
#!/usr/lpp/ssp/perl5/bin/perl

my $logfile="/var/adm/SPlogs/SPdaemon.log";
my $lastentry;


open (LOG,"cat $logfile|") ||
        die "Ops! Can't open $logfile: $!\n";

while (<LOG>) {
   if(/Authentication failed/) {
        $lastentry = $_;
        }
   }

print "$lastentry";
```

The script prints out the `Authentication failed` entry from the log file. If there is no new entry, the old value is the same as the new value, so we have to create a monitor that gets notified every time the value of this variable changes. Let’s take a look at the monitor’s definition:

```
[sp5en0:/]# /usr/lpp/ssp/bin/pmandef -s authfailed \
-e 'IBM.PSSP.pm.User_state1:NodeNum=0-5:X@0!=X@P0' \
-c "/usr/local/bin/SaySomething.pl" \
-n 0
```

This command defines a monitor, through PMAN, for the IBM.PSSP.pm.User_state1 resource variable. The expression `X@0!=X@P0` means that if the previous value (X@P0) is different from the current value (X@0), then the variable has changed. This special syntax is due to the fact that these user-defined variables are structured byte strings (SBSs), so to access the value of this variable you have to index this structure. However, these variables have only one field, so only index `0` is valid.

You can get a complete definition of this resource variable (and others) by executing the following command:

```
[sp5en0:/]# haemqvar "" IBM.PSSP.pm.User_state1 "*"|more
```

This gives you a good explanation along with examples of how to use it.

Now that we have subscribed our monitor, let's see what the /usr/local/bin/SaySomething.pl script does:

```
#!/usr/lpp/ssp/perl5/bin/perl

$cwsdisplay = "sp5en0:0";
$term="/usr/dt/bin/aixterm";
$cmd = "/usr/local/bin/SayItLoud.pl";
$title = qq/\"Warning on node $ENV{'PMAN_LOCATION'}\"/;
$msg = $ENV{'PMAN_RVFIELD0'};
$bg = "red";
$fg = "white";
$geo = "60x5+200+100";

$execute = qq/$term -display $cwsdisplay -T $title -geometry $geo -bg $bg
-fg $fg -e $cmd $msg/;

system($execute);
```

This script opens a warning window with a red background notifying the operator (it is run on node 0, the CWS) about the intruder.

The script /usr/local/bin/SayItLoud.pl displays the error log entry (the resource variable value) inside the warning window. Let's take a look at this script:

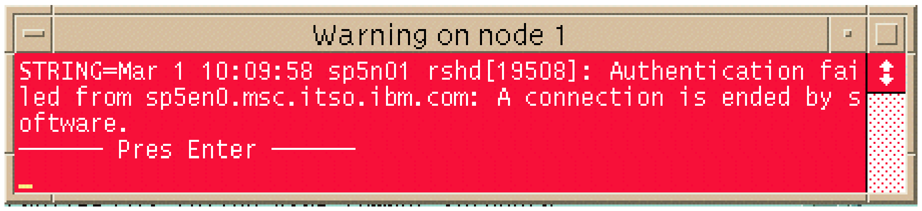
```
#!/usr/lpp/ssp/perl5/bin/perl

print "@ARGV\n";
print "------ Press Enter ------\n";
```

```
<STDIN>
```

Now that the monitor is active, let's try to access one of the nodes. We destroy our credentials (`kdestroy` command) and then try to execute a command on one of the nodes:

```
[sp5en0:/]# kdestroy
[sp5en0:/]# dsh -w sp5n01 date
sp5n01: spk4rsh: 0041-003 No tickets file found.  You need to run "k4init".
sp5n01: rshd: 0826-813 Permission is denied.
dsh:  5025-509 sp5n01 rsh had exit code 1
```

After a few seconds (a minute at most), we receive the warning window, shown in the warning message at the CWS illustrated in Figure 1.



```
STRING=Mar 1 10:09:58 sp5n01 rshd[19508]: Authentication fai
led from sp5en0.msc.itso.ibm.com: A connection is ended by s
oftware.
------- Pres Enter -------
```

Figure 1. User-defined resource variables—Warning window example

The example shown here is very simple. It is not intended to be complete, but instead, to illustrate the use of these user-defined resource variables.

1.2.2.3 SP Perspectives

The SP Perspectives is a set of applications, each with a graphical user interface (GUI), that enables you to perform monitoring and system management tasks for your SP system by directly manipulating icons that represent system objects.

Event Perspective is one of these applications. It provides a graphical interface to Event Management and the Problem Management subsystems.

Through this interface, you can create monitors for triggering events based on defined conditions and generate actions by using the Problem Management subsystem when any of these events are triggered.

1.2.3 Defining conditions

The procedure for setting up monitors in Event Perspective is very straightforward. A condition needs to be created prior to the definition of the monitor.

Conditions are based on resource variables, resource identifiers, and expressions, which the Event Manager daemon then evaluates.

To better illustrate this point, let's define a condition for a full file system. This condition will later be used in a monitor. Following are the steps required for creating a condition:

Step 1 In this step, you need to narrow down the condition you want to monitor. For example: I want to monitor free space in the /tmp file system. Then, you have to decide on the particular resource you want to monitor and the condition. You should also think about where in the SP system you want to monitor free space in /tmp. Let's decide on that later.

Step 2 Identify the resource variable. Once you have decided the condition you want to monitor, you need to find the variable that represents the particular resource associated with the condition. In our case, it is free space in a file system.

PSSP provides some facilities to determine the right variable. In releases previous to PSSP 3.1, the only way to get some information on resource variables is through the help facility on SP Perspectives. However, in PSSP 3.1 or later, there is a command that helps you find the right variable and provides you with information about how to use it. Let's use this command, called `haemqvar`.

We can use this command to list all the variables related to file systems, as follows:

```
[sp3en0:/]# haemqvar -d IBM.PSSP.aixos.FS "" "*"
IBM.PSSP.aixos.VG.free   Free space in volume group, MB.
IBM.PSSP.aixos.FS.%totused   Used space in percent.
IBM.PSSP.aixos.FS.%nodesused   Percent of file nodes that are used.
```

In this case, we have listed the variables within the IBM.PSSP.aixos.FS class. You may use the same format to list other classes.

In particular, we are interested in the IBM.PSSP.aixos.FS.%totused variable that represents exactly what we want to monitor.

Step 3 Define the expression. In order to define the expression we will use in our condition, we need to know how we use this variable. In other words, what are the resource identifiers for this variable. So, let's use the `haemqvar` command again; but this time, let's query the specific variable and get a full description as shown in Figure 2.

```
[sp3en0:/]# haemqvar  "IBM.PSSP.aixos.FS" IBM.PSSP.aixos.FS.%totused "*"
Variable Name:  IBM.PSSP.aixos.FS.%totused
Value Type:     Quantity
Data Type:      float
Initial Value:  0.000000
Class:          IBM.PSSP.aixos.FS
Locator:        NodeNum
Variable Description:
    Used space in percent.

    IBM.PSSP.aixos.FS.%totused represents the percent of space in a file
    system that is in use. The resource variable's resource ID specifies
    the names of the ldescriptogical volume (LV) and volume group (VG) of the file
    system, and the number of the node (NodeNum) on which the file system
    resides.
...lines not displayed...
The lsvg command can be used to list, and display information about
    the volume groups defined on a node. For example:

      # lsvg | lsvg -i -l
      spdata:
      LV NAME     TYPE     LPs   PPs   PVs  LV STATE      MOUNT POINT
      spdatalv    jfs      450   450   1    open/syncd    /spdata
      loglv00     jfslog   1     1     1    open/syncd    N/A
      rootvg:
      LV NAME     TYPE     LPs   PPs   PVs  LV STATE      MOUNT POINT
      hd6         paging   64    64    1    open/syncd    N/A
      hd5         boot     1     1     1    closed/syncd  N/A
      hd8         jfslog   1     1     1    open/syncd    N/A
      hd4         jfs      18    18    1    open/syncd    /
      hd2         jfs      148   148   1    open/syncd    /usr
      hd9var      jfs      13    13    1    open/syncd    /var
      hd3         jfs      32    32    1    open/syncd    /tmp
      hd1         jfs      1     1     1    open/syncd    /home

...lines not displayed...
    When enough files have been created to use all the available
    i-nodes, no more files can be created, even if the file system
    has free space. The "%nodesused" resource variable can be used
    to monitor the percent of file nodes which are in use.

    Example expression:

    To receive a notification that the file system mounted on /tmp on any
    node is more than 90% full, and also receive a notfication when the
    percentage has subsequently dropped below 80%, one could register
    for the following event using the HA_EM_CMD_REG2 command:

        Resource variable:   IBM.PSSP.aixos.FS.%totused
        Resource ID:         VG=rootvg;LV=hd3;NodeNum=*
        Expression:          X > 90
        Re-arm expression:   X < 80

....lines not displayed....
```

Figure 2. Resource variable query (Partial view)

This command gives us a complete description of the variable and also tells us how to use it in an expression. Our expression would be `X > 90`.

We could use a re-arm expression in our condition. A re-arm expression is optional. It defines a second condition that Event Manager will switch to when the main expression triggers. In our example, a re-arm expression would be `X < 60`, which means that after the file system is more than 90 percent used, Event Manager will send us a notification and then continue monitoring the file system; but now, it will send us a notification when the space used falls below 60 percent.

Step 4 Create the condition. To create the condition, let's move the focus to the conditions pane on Event Perspective, and then select **Actions -> Create...**, as shown in Figure 3.

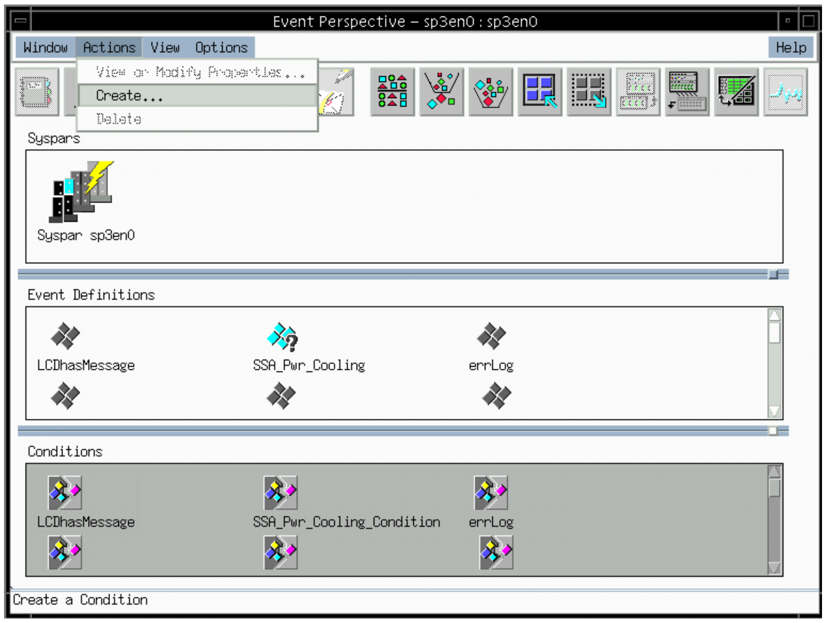


Figure 3. Create condition option from Event Perspective

When you click **Actions -> Create...** , you are presented with the Create Condition pane, as shown in Figure 4 on page 14.

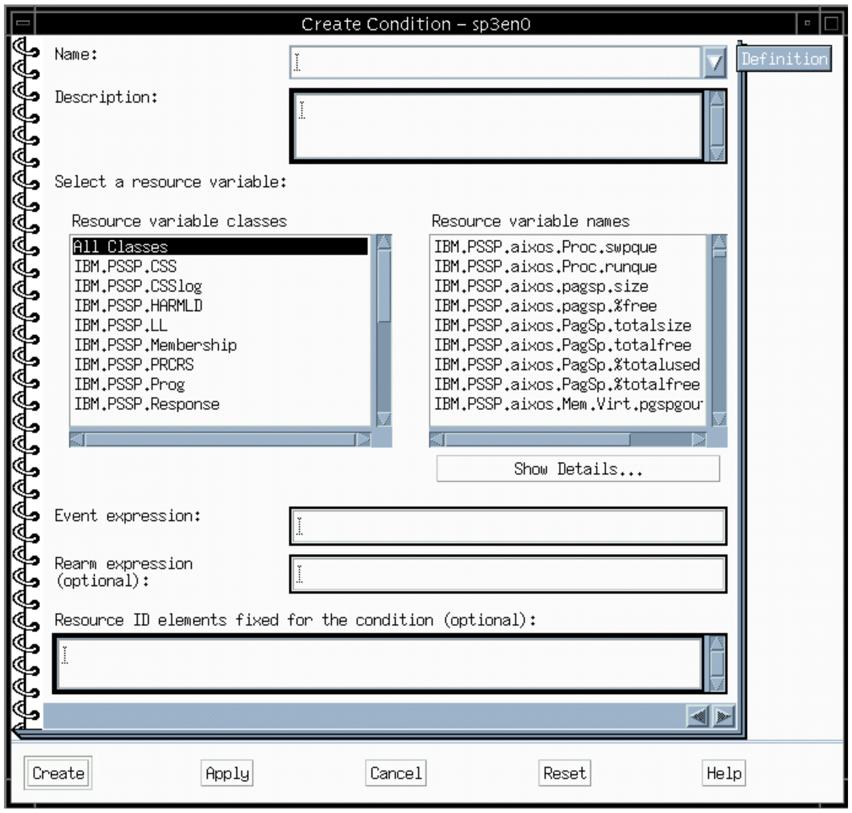
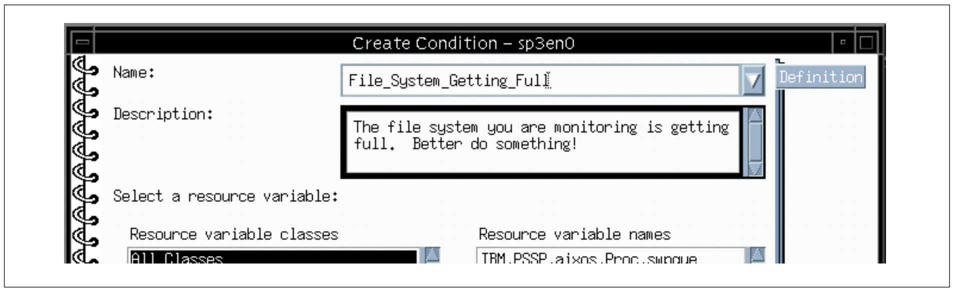

Create Condition – sp3en0
Definition
Name:
Description:
Select a resource variable:
Resource variable classes
All Classes
IBM.PSSP.CSS
IBM.PSSP.CSSlog
IBM.PSSP.HARMLD
IBM.PSSP.LL
IBM.PSSP.Membership
IBM.PSSP.PRCRS
IBM.PSSP.Prog
IBM.PSSP.Response
Resource variable names
IBM.PSSP.aixos.Proc.swpque
IBM.PSSP.aixos.Proc.runque
IBM.PSSP.aixos.pagsp.size
IBM.PSSP.aixos.pagsp.%free
IBM.PSSP.aixos.PagSp.totalsize
IBM.PSSP.aixos.PagSp.totalfree
IBM.PSSP.aixos.PagSp.%totalused
IBM.PSSP.aixos.PagSp.%totalfree
IBM.PSSP.aixos.Mem.Virt.pgspgou
Show Details...
Event expression:
Rearm expression (optional):
Resource ID elements fixed for the condition (optional):
Create
Apply
Cancel
Reset
Help

Figure 4. Create Condition pane

As you can see in the Create Condition pane, there are two initial input boxes for the name (Name) of the condition and the description (Description). For our example, let's name the condition `File_System_Getting_Full`, and add a brief description, such as `The file system you are monitoring is getting full. Better do something!`. This is shown in Figure 5 on page 15.


Create Condition – sp3en0

Name: File_System_Getting_Full

Description: The file system you are monitoring is getting full. Better do something!

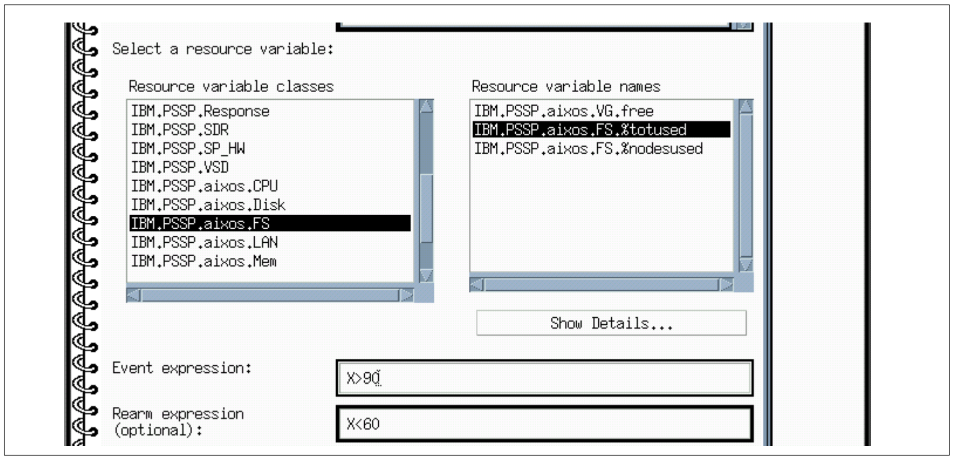
Select a resource variable:

Resource variable classes: All Classes

Resource variable names: IBM.PSSP.aixos.Proc.swpque

Figure 5. Defining name and description of a condition

Now, we select the resource variable class, **IBM.PSSP.aixos.FS**, and the resource variable, **IBM.PSSP.aixos.FS.%totused**, followed by the expression and re-arm expression we defined in the previous step. This is shown in Figure 6.


Select a resource variable:

Resource variable classes	Resource variable names
IBM.PSSP.Response	IBM.PSSP.aixos.VG.free
IBM.PSSP.SDR	IBM.PSSP.aixos.FS.%totused
IBM.PSSP.SP_HW	IBM.PSSP.aixos.FS.%nodesused
IBM.PSSP.VSD	
IBM.PSSP.aixos.CPU	
IBM.PSSP.aixos.Disk	
IBM.PSSP.aixos.FS	
IBM.PSSP.aixos.LAN	
IBM.PSSP.aixos.Mem	

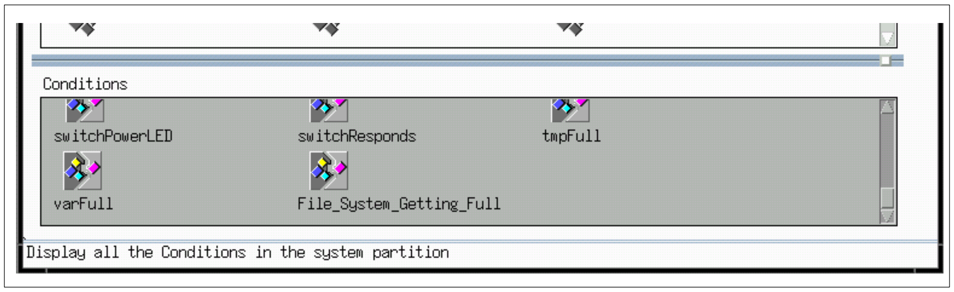
Event expression: X>90

Rearm expression (optional): X<60

Figure 6. Selecting resource variable and defining expression

If you select **Show Details...**, it will present the same output we obtained through the `haemqvar` command. We will leave the last input box empty, which represents the resources ID that you want to fix. For example, this resource variable (IBM.PSSP.aixos.FS.%totused) has two resource IDs. One is the volume group (VG) name, and the other is the logical volume (LV) name. By using the last input box, we could have fixed one or two resource IDs to a specific file system so that this condition could be applied to that particular file system only. However, leaving this input blank enables us to use this condition in any file system monitor.

Once the condition has been created, an icon appears in the Conditions pane, as shown in Figure 7.



Conditions

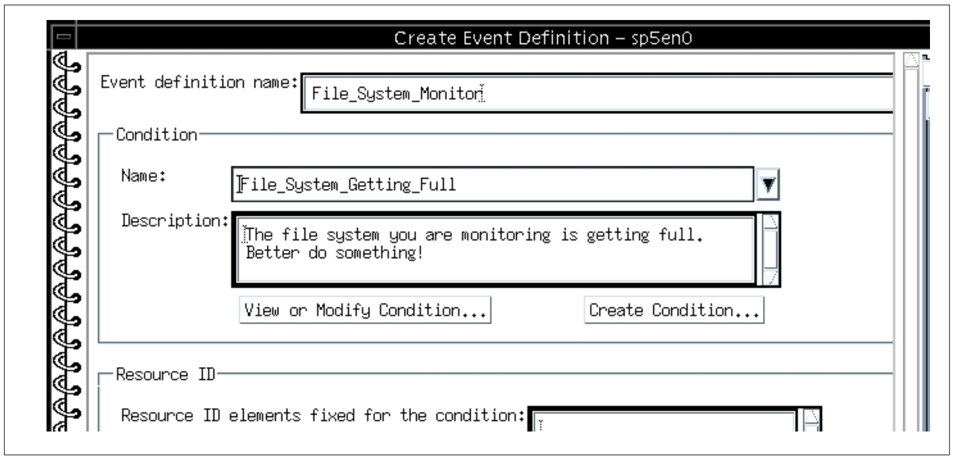
switchPowerLED switchResponds tmpFull

varFull File_System_Getting_Full

Display all the Conditions in the system partition

Figure 7. Conditions pane—New condition

Step 5 Once the condition has been defined, you need to define a monitor that will use this condition. To define a monitor, make the monitor pane your current pane and then click **Actions -> Create...**, and then you define the monitor as shown in Figure 8.



Create Event Definition – sp5en0

Event definition name: File_System_Monitor

Condition

Name: File_System_Getting_Full

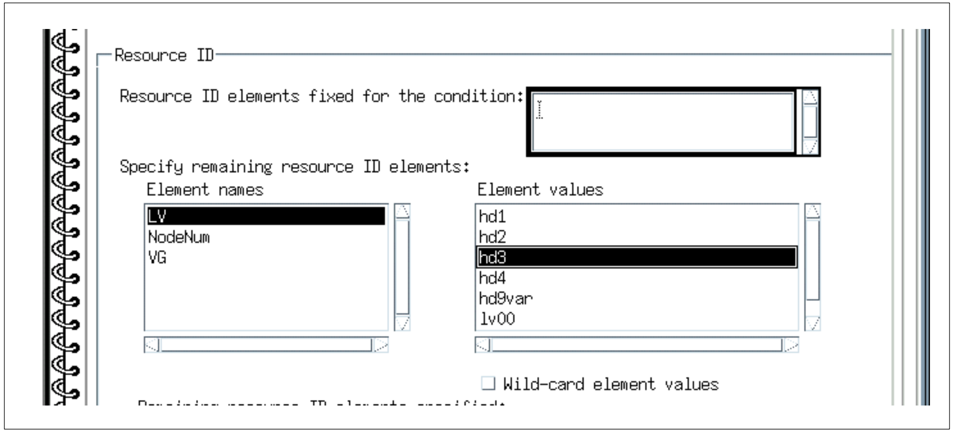
Description: The file system you are monitoring is getting full. Better do something!

Resource ID

Resource ID elements fixed for the condition:

Figure 8. Defining a new monitor

After selecting the previously-defined condition, you need to identify the file system you want to monitor. Let’s use /tmp as the file system we want to monitor, and let’s monitor /tmp on all nodes. To do that, first we need to identify /tmp as the file system, so we start with the logical volume as shown Figure 9 on page 17.



Resource ID

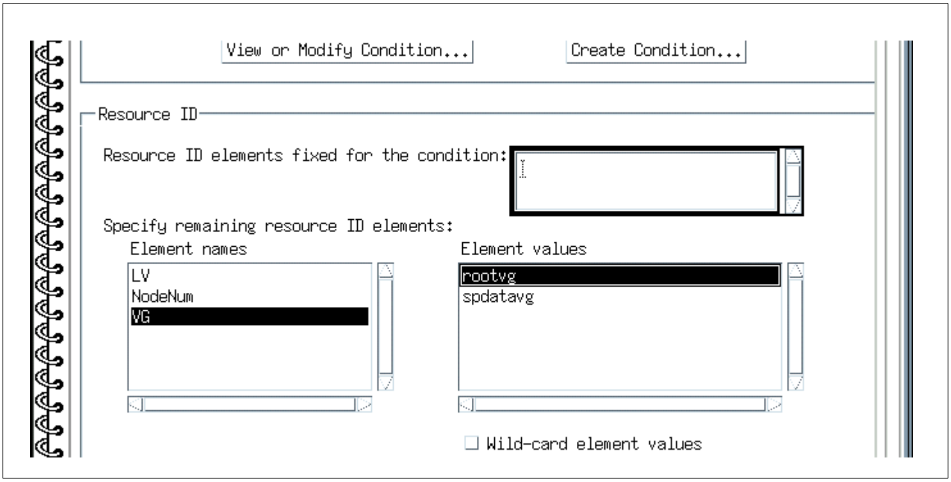
Resource ID elements fixed for the condition:

Specify remaining resource ID elements:

Element names	Element values
LV	hd1
NodeNum	hd2
VG	hd3
	hd4
	hd9var
	lv00

Figure 9. Selecting hd3 as the logical volume

Then, we select the volume group, as shown in Figure 10.



Resource ID

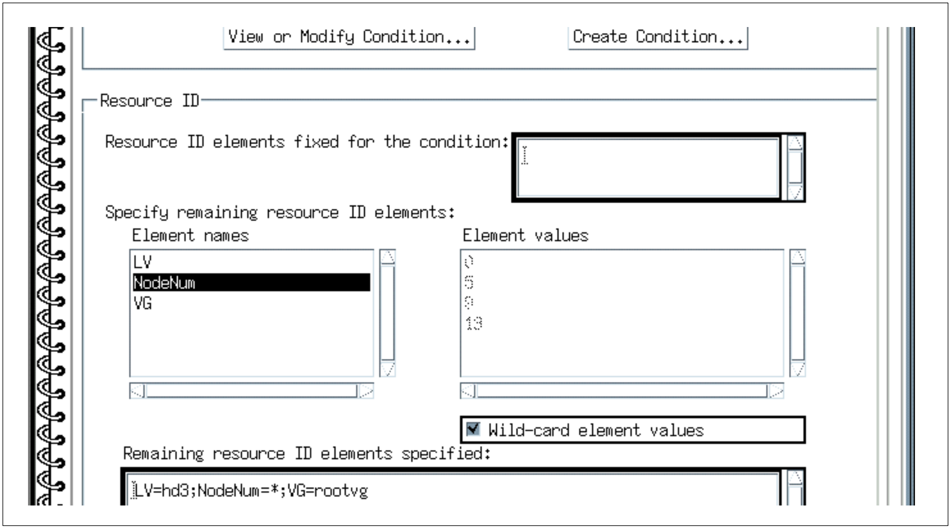
Resource ID elements fixed for the condition:

Specify remaining resource ID elements:

Element names	Element values
LV	rootvg
NodeNum	spdatavg
VG	

Figure 10. Selecting rootvg as the volume group

Finally, we select all nodes and the CWS as targets for this monitor. To do that, we use the wild-card option as shown in Figure 11 on page 18.



Resource ID elements fixed for the condition:

Specify remaining resource ID elements:

Element names	Element values
LV	0
NodeNum	5
VG	9
	13

☑ Wild-card element values

Remaining resource ID elements specified:

LV=hd3;NodeNum=*;VG=rootvg

Figure 11. Using the wild-card option

Once all the information as been filled in the input fields, you can create the monitor by clicking the **Create** button. For this example, we have not selected any special action, so only a window notification will be displayed when the condition evaluates true. However, you may add actions to your monitor by selecting the **Actions** tab on the right-hand side of the monitors pane.

The new monitor have been defined and is displayed in the Events pane as shown in Figure 12.

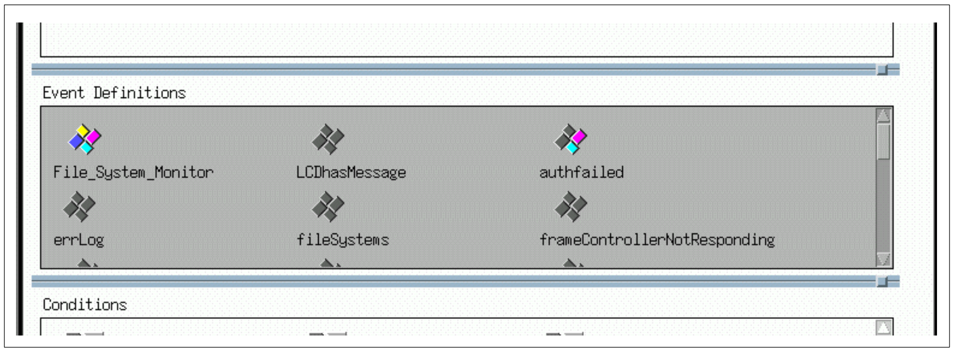
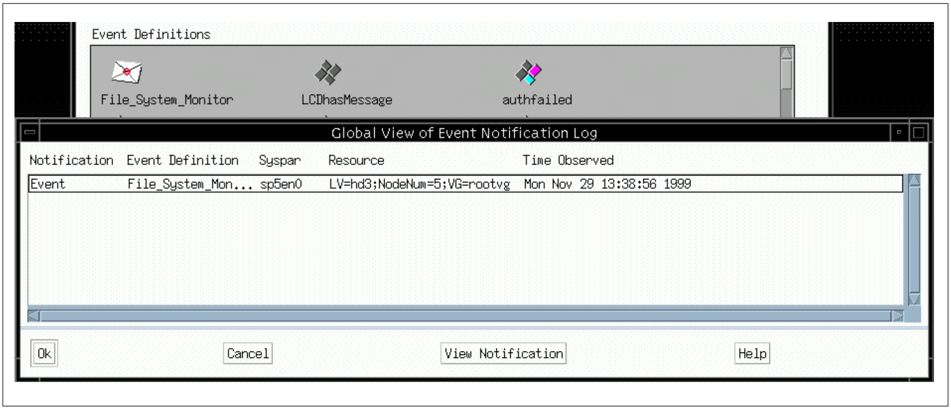


Figure 12. New monitor

To test the new monitor, we fill up /tmp in one of the nodes by using the `dd` command as follows:

```
dd if=/dev/zero of=/tmp/file
```

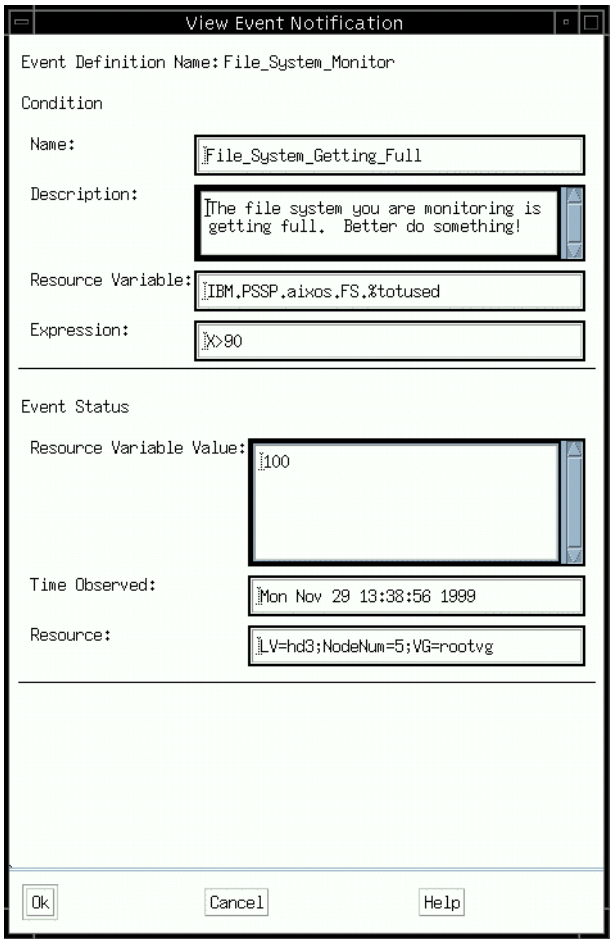
Once the file system is 100 percent full, we received notification at the Control Workstation's display as shown in Figure 13.



```
Notification  Event Definition  Syspar   Resource                     Time Observed
Event         File_System_Mon... sp5en0  LV=hd3;NodeNum=5;VG=rootvg   Mon Nov 29 13:38:56 1999
```

Figure 13. Event notification

To view the notification, we click the **View Notification** button and we get the window illustrated in Figure 14 on page 20.



Event Definition Name: File_System_Monitor

Condition

Name:	File_System_Getting_Full
Description:	The file system you are monitoring is getting full. Better do something!
Resource Variable:	IBM.PSSP.aixos.FS.%totused
Expression:	X>90

Event Status

Resource Variable Value:	100
Time Observed:	Mon Nov 29 13:38:56 1999
Resource:	LV=hd3;NodeNum=5;VG=rootvg

Figure 14. Event notification details

The re-arm events and notifications work in a similar way. To trigger the re-arm event (X < 60), we remove the file from the /tmp file system. The re-arm event notification and the notification windows are very similar to the ones we just saw. For a detailed discussion of Event Perspective and monitors, refer to *SP Perspectives: A New View of Your SP System*, SG24-5180.

1.3 Reporting problems

Sometimes you will not be able to solve a problem, so you will have to ask for help from your IBM Support Center. Keep this in mind and always exercise the methodology outlined in 1.1, “Methodologies” on page 1. For an effective use of the IBM Support Center, refer to “Making Effective Use of the IBM Support Center” on page 9 of the *PSSP: Diagnosis Guide*, GA22-7350. You will find very useful information in this reference about how to report problems to your IBM Support Center.

Chapter 2. RS/6000 SP installation

In this chapter, we explain the installation process for an RS/6000 SP and the problems that may arise.

The installation process is divided into three areas:

1. Environment preparation and SDR data entry.
2. Boot/install server configuration. After this, nodes are ready to be installed over the network.
3. Node installation.

2.1 Environment preparation and SDR data entry

In this section, we cover the Control Workstation (CWS) setup.

2.1.1 AIX preparation

First, we cover conditions that must be met before installing the PSSP code on the CWS.

Let us look at the requirements that must be met:

- Refer to the "READ THIS FIRST" document that accompanies the PSSP installation media for the latest information for supported AIX releases and PTF levels. You can download this document from the following URL:

  `http://www.rs6000.ibm.com/resource/aix_resource/sp_books/pssp/index.html`

- The bos.net fileset is installed (TCP/IP and NFS client and server).
- The perfagent.tools fileset is installed.
- All names and IP addresses for all the SP nodes are resolved (through /etc/hosts, DNS, or NIS).
- The network and TTY connections are created and verified.
- The environment variables are set up properly.
- The volume group, logical volumes, and file systems have been created, and the AIX lppsource, mksysb images, and PSSP filesets have been placed in the correct location. See *PSSP: Installation and Migration Guide*, GA22-7347.

2.1.2 PSSP installation and configuration

We now discuss the System Data Repository (SDR) populating phase.

The first step is the installation of PSSP software on the CWS. For information on the fileset descriptions, consult *IBM RS/6000 SP: Planning, Volume 2, Control Workstation and Software Environment,* GA22-7281.

After the installation of PSSP has completed, invoke `setup_authent` to set up Kerberos. Kerberos problem determination is covered in Chapter 3, “Security environment” on page 167.

The next step is invoking the `install_cw` command, which performs the following actions:

1. Configures the CWS.
2. Installs PSSP SMIT panels.
3. Starts SP daemons.
4. Configures the SDR.
5. Updates /etc/inittab and /etc/services.
6. Sets the node_number for the CWS to zero in the ODM database.
7. Configures the system as one partition.

After this step, verify the success of the `install_cw` command by issuing the following commands:

```
SDR_test
spmon_itest
```

These commands create log files in the /var/adm/SPlogs directory that must be verified.

After this, you start populating the SDR database, so the following steps must be performed:

1. Enter site environment information. Here, we must define the services that will be available on the RS/6000 SP. To perform this step, invoke `smitty site_env_dialog` or the `spsitenv` command.
2. Enter frame information and reinitialize the SDR. This step creates Frame objects in the SDR for each frame on the system. This also creates Node objects in the SDR. To enter data, invoke `smitty sp_frame_dialog` or the `spframe` command.

   The problems that may arise with these commands are related to hardmon and Kerberos. Therefore, if you have a problem, verify the serial connections and invoke `spmon_itest` to determine if there is a problem. Kerberos problem determination is covered in Chapter 3, “Security environment” on page 167.

3. Enter non-SP frame information. This step is necessary if you have SP-attached server nodes. Invoke `smitty nonsp_frame_dialog` or the `spframe` command to introduce data into the SDR database.
4. At this point, `spmon_ctest` should be invoked and the /var/adm/SPlog/spmon_ctest.log log file should be verified.
5. In this step, you should update the supervisor card microcode if needed. The following command must be invoked to determine if there are supervisor cards that need to have their microcode updated:

   ```
   spsvrmgr -G -r status all
   ```

   If the last level of microcode is not on the supervisor cards, you need to update them by executing the following command:

   ```
   spsvrmgr -G -u all
   ```

6. In this step, we enter node information according to the planning worksheets. To perform this step, invoke `smitty sp_eth_dialog` or the `spethernt` command.
7. Acquire the hardware Ethernet address by invoking `smitty hrdwrad_dialog` or the `sphrdwrad` command.

   The problems that are associated with this command are the same as those for the network boot of the node. So, if you have a problem, consult 2.3.3, “How to diagnose boot/install problems” on page 109 for more information.

8. Configure additional adapters for nodes. In this step, you must enter adapter information according to the planning worksheets by invoking `smitty add_adapt_dialog` or the `spadaptrs` command.
9. Configure the initial hostname for nodes by invoking `smitty hostname_dialog` or the `sphostnam` command.

   In this step, you configure the initial hostname of the node. The reliable hostname is always associated with the en0 interface.

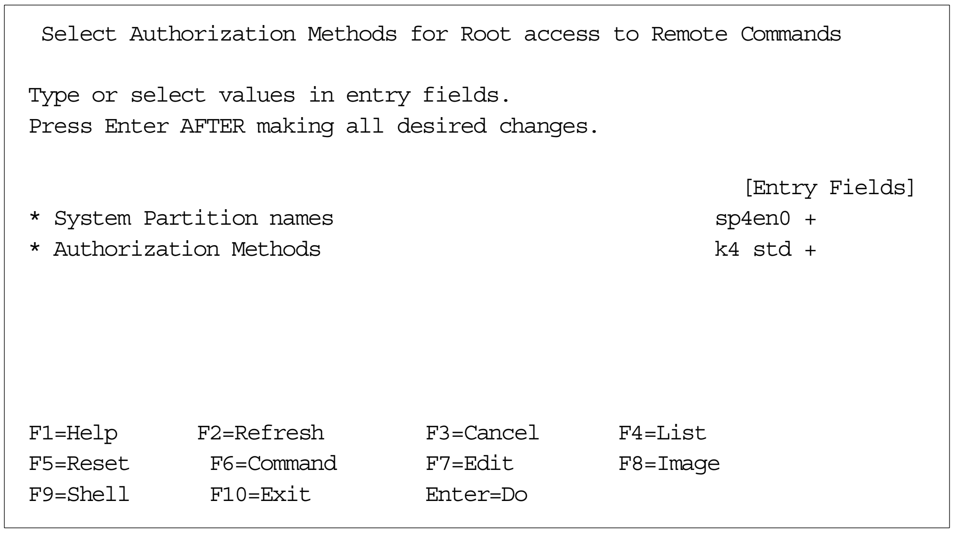
10. Create authorization files by invoking `smitty spauth_rcmd` or the `spsetauth` command.

    In this step, you create the appropriate authorization methods so that you can use root remote commands. The possible values are:

    - k5 for Kerberos V5 (DCE is needed. The .klogin5 file is not created by PSSP.)

- k4 for Kerberos V4 (.klogin)
- std for standard authorization (.rhosts)

A sample SMIT panel is shown in Figure 15.



```
 Select Authorization Methods for Root access to Remote Commands

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

                                                             [Entry Fields]
* System Partition names                                   sp4en0 +
* Authorization Methods                                    k4 std +






F1=Help       F2=Refresh         F3=Cancel       F4=List
F5=Reset       F6=Command        F7=Edit         F8=Image
F9=Shell       F10=Exit          Enter=Do
```

Figure 15. SMIT panel—Select authorization methods

The k4 authentication is a must for the RS/6000 SP.

If you do not have DCE installed, and std is not set, you will not be able to perform root remote commands, such as ftp, telnet, or rlogin.

11. Start the system partition subsystem by issuing:

```
syspar_ctrl -A
```

This command adds and starts partition-sensitive subsystems that are listed in the /usr/lpp/ssp/config/cmi/syspar_subsystems file.

To verify that the subsystems have been started, execute:

```
syspar_ctrl -E
```

If PTPE has not been installed on your system, and a message is received complaining about this, you can ignore it.

For problem determination on RSCT subsystems consult Chapter 4, “RS/6000 Cluster Technology (RSCT)” on page 191.

12. In this step, you create or modify Volume_Group objects in the SDR. To create volume group objects, invoke `smitty createvg_dialog` or the

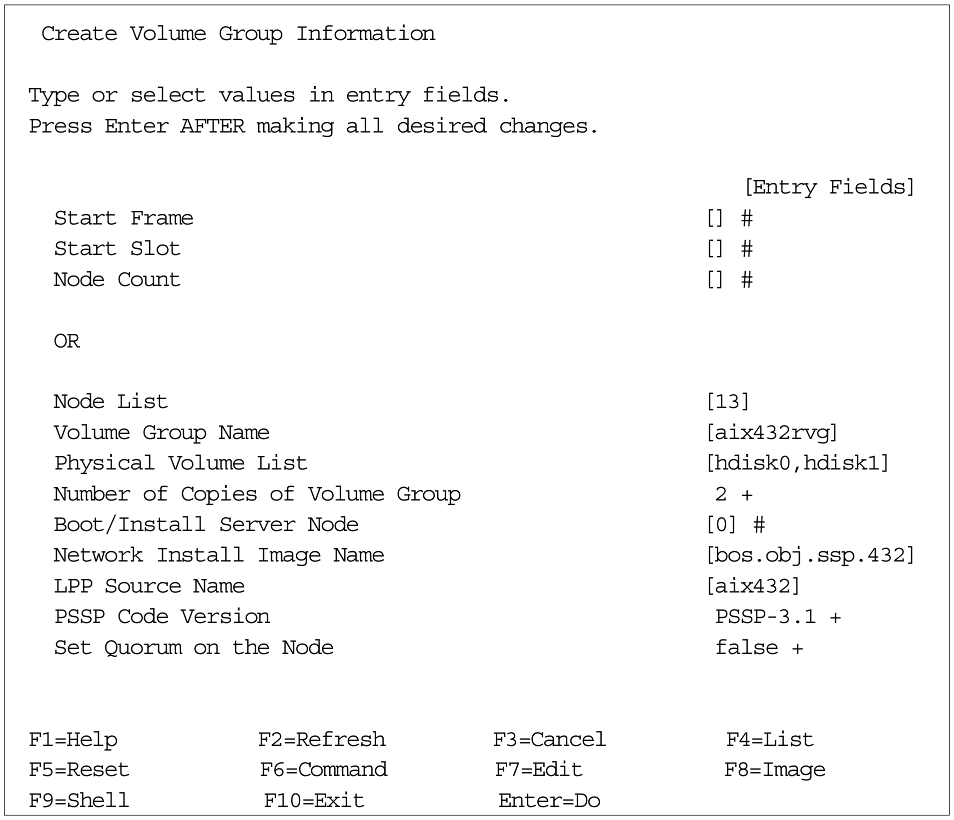
`spmkvgobj` command. To change objects, invoke `smitty changevg_dialog` or the `spchvgobj` command.

The attributes that can be set are:

- node number
- volume group name
- physical volume list
- quorum
- copies
- install image (mksysb image)
- PSSP code version
- lppsource name
- boot/install server

The Node object will point to one Volume_Group SDR object through the selected_vg and node_number attributes.

A sample of the SMIT panel is shown in Figure 16 on page 28.



```
 Create Volume Group Information

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

                                                          [Entry Fields]
  Start Frame                                           [] #
  Start Slot                                            [] #
  Node Count                                            [] #

  OR

  Node List                                             [13]
  Volume Group Name                                     [aix432rvg]
  Physical Volume List                                  [hdisk0,hdisk1]
  Number of Copies of Volume Group                       2 +
  Boot/Install Server Node                              [0] #
  Network Install Image Name                            [bos.obj.ssp.432]
  LPP Source Name                                       [aix432]
  PSSP Code Version                                      PSSP-3.1 +
  Set Quorum on the Node                                 false +


F1=Help           F2=Refresh        F3=Cancel         F4=List
F5=Reset          F6=Command        F7=Edit           F8=Image
F9=Shell          F10=Exit          Enter=Do
```

Figure 16. SMIT panel—Create volume group information

In the physical volume list, you can define the hdisk names separated by commas, or the disk locations separated by colons.

We strongly recommend you use disk locations instead of hdisk names.

An example of a disk location is as follows:

For SSA disks:

ssar//000xxxxxxxxx00D:ssar//000yyyyyyyyy00D

Where xxxxxxxxx, yyyyyyyyy are the SSA serial disk numbers.

For SCSI disks:

00-00-00-0,0:00-00-00-0,1

13. The last step is the selection of the boot response and volume group for the nodes.

The problems associated with these steps are usually SDR problems, which can be:

- The `SDR` daemon is not running.
- The /etc/SDR_dest_info file is incorrect or corrupted.
- The SDR database is corrupted.

For SDR daemon problems, start the daemon, if it is not running, by issuing:

```
startsrc -g sdr
```

Then investigate if there is any problem by looking in the /var/adm/SPlogs/sdr log directory.

The SDR_dest_info file on your nodes must be similar to the sample given in Figure 17. The entries in the SDR_dest_info file should point to the single IP address you have set so far (the Control Workstation's IP address).

```
default:192.168.4.130
primary:192.168.4.130
nameofdefault:sp4en0
nameofprimary:sp4en0
```

Figure 17. Sample SDR_dest_info file

For corrupted SDR files, there is no easy way of debugging the problem. The best way is to invoke the `SDRGetObjects` command for the SDR classes that are suspected to be corrupted.

2.2 Boot/install server configuration

In this section, we cover the boot/install server setup. It is organized into the following subsections:

- Basic checking: This covers some basic checking that you must do before setting up the boot/install server.
- setup_server overview: This is an overview of the setup_server script.
- setup_server wrappers: This covers the setup_server subcommands called wrappers.
- Some useful NIM commands.
- Problem determination methodology.
- Case studies.

The CWS is considered a boot/install server (the first BIS available on your system) for the discussions in this chapter.

2.2.1 Basic checking

First, we discuss the basic checking you must perform before running the `setup_server` command:

- The correct version of AIX has been installed on the boot/install server.
- The PSSP code has been successfully installed on the boot/install servers.
- We should take into account the following space requirements on the boot/install server (for more information, refer to *IBM RS/6000 SP: Planning, Volume 2, Control Workstation and Software Environment,* GA22-7281):
  - The /tftpboot directory holds the boot images of the boot/install clients and some additional files (srv-tab, intall_info, config_info, and niminfo files for each of the boot/install clients). The boot images that will be created depend on the following: the AIX version, the processor type, the platform type of the boot/install clients, and networks used for install.
  - The lppsource directory contains the filesets needed to create the SPOT and the filesets that are needed in the installation/migration process. The lppsource only exists on the CWS.
  - The SPOT space requirements on boot/install servers depend on:
    - The lppsource versions that are needed by the boot/install clients
    - The number of filesets in the lppsource directory
  - The images directory holds the mksysb images needed by the boot/install clients.
  - The pssplpp directory holds the PSSP code needed by the boot/install clients.
- Verify that the environment variables of this boot/install server have been set properly.
- Verify that the network interfaces have been set properly with correct IP address and netmask.
- Verify that the hosts IP resolution is working properly (through /etc/hosts, DNS, or NIS), and IP addresses for all the network interfaces on the nodes are resolved there.

- The filesets needed for each lppsource have been placed in the correct location with the correct permissions.

Note

Invoke `inutoc` on each lppsource directory after placing or adding filesets to it.

Note

Put all the necessary filesets in the lppsource directory to create the SPOT. If there are additional device support media, also put it in the lppsource directory.

The necessary filesets for creating the SPOT are defined in the /usr/lpp/bos.sysmgr/nim/methods/c_sh_lib file.

The variable DEFAULT_SPOT_OPTIONS (or DEFAULT_SPOT_<version> if the lppsource version is the same as <version>) defines the minimum filesets that must be on the lppsource to be able to create the SPOT.

The filesets defined by REQUIRED_SIMAGES<version> (if the lppsource version is not in the file, choose REQUIRED_SIMAGES) must also be placed in the lppsource directory.

- The mksysb images needed by the boot/install clients are placed in the /spdata/sys1/install/images directory on the CWS and have correct permissions.
- The PSSP fileset versions needed by boot/install clients are placed in the /spdata/sys1/install/pssplpp/PSSP-<ver.rel> directory on the CWS and have the correct permissions.

2.2.2 setup_server overview

The setup_server script is used to set up a boot/install server. Figure 18 on page 32 shows a flow diagram of setup_server.

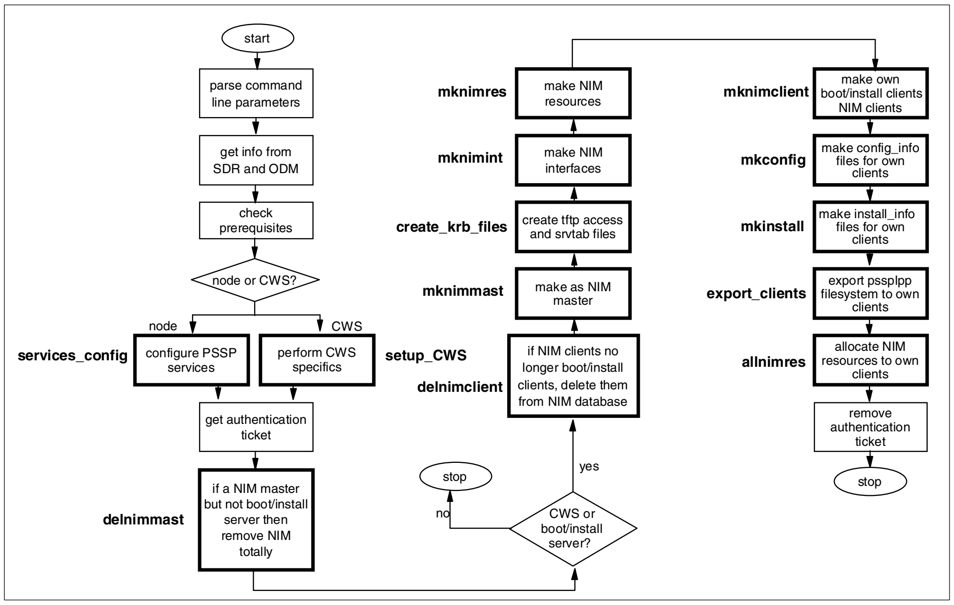

start
parse command line parameters
get info from SDR and ODM
check prerequisites
node or CWS?
node
CWS
services_config
configure PSSP services
perform CWS specifics
setup_CWS
get authentication ticket
delnimmast
if a NIM master but not boot/install server then remove NIM totally
CWS or boot/install server?
no
stop
yes
delnimclient
if NIM clients no longer boot/install clients, delete them from NIM database
mknimmast
make as NIM master
create_krb_files
create tftp access and srvtab files
mknimint
make NIM interfaces
mknimres
make NIM resources
mknimclient
make own boot/install clients NIM clients
mkconfig
make config_info files for own clients
mkinstall
make install_info files for own clients
export_clients
export pssplpp filesystem to own clients
allnimres
allocate NIM resources to own clients
remove authentication ticket
stop

Figure 18. setup_server flow diagram

setup_server is divided into modular sections called wrappers. Wrappers are functions in Perl that can run independently.

Let us now look at the flow of this command:

1. Parse command line parameters.
2. Retrieve information from SDR and ODM (node_number).
3. Check prerequisites.
4. Configure the PSSP services for the node. The `services_config` wrapper is invoked to perform this function.
5. If you are running setup_server on the CWS, invoke the `setup_CWS` wrapper.
6. Get an authentication ticket.
7. If the node in which you are running setup_server is configured as NIM master, but the node does not have any NIM client (running `splstdata -b`, there are no nodes pointing to this one as boot/install server), invoke the following wrapper:

   ```
   delnimmast -l <node_number>
   ```

   `<node_number>` is the node number where setup_server is running.

This wrapper deletes the NIM master configuration. It also un-installs the NIM master filesets.

8. If this is not a boot/install server, do not do anything else (that is, there are no other nodes referring to this node as boot/install server in the SDR).
9. Invoke the `delnimclient` wrapper:

   ```
   delnimclient -s <node_number>
   ```

   `<node_number>` is the node number where setup_server is running.

   This wrapper deletes from the NIM master database all the nodes that are no longer boot/install clients.
10. Invoke the `mknimmast` wrapper:

    ```
    mknimmast -l <node_number>
    ```

    `<node_number>` is the node number where setup_server is running.

    This wrapper installs NIM master filesets and configures the node as NIM master, if the node is not an NIM master already.
11. Invoke the `create_krb_files` wrapper with no arguments.

    This wrapper creates/updates the /etc/tftpaccess.ctl file on the BIS node. It creates the /tftpboot/<hostname>-new-srvtab file for each boot/install client that has a bootp_reponse equal to install, migrate, or customize.
12. Invoke the `mknimint` wrapper:

    ```
    mknimint -l <node_number>
    ```

    `<node_number>` is the node number where setup_server is running.

    This wrapper creates network objects on the NIM master to serve NIM clients. If setup_server was not invoked from the CWS, the network objects of the CWS are added.
13. Invoke the `mknimres` wrapper:

    ```
    mknimres -l <node_number>
    ```

    `<node_number>` is the node number where setup_server is running.

    This wrapper creates NIM resources that are needed in order to install client nodes. The resources created on this boot/install server depend on the bootp_response attribute that the boot/install clients have in the SDR. If the resource is already available, it will not be created.

Table 1 shows the resources that are created by the `mknimres` wrapper.

Table 1. Resources created

bootp_response	**Resource needed**
install	bosinst_data, lpp_source, mksysb, spot, pssplpp, psspscript
migrate	bosinst_data, lpp_source, spot, pssplpp, psspscript
customize	pssplpp, psspscript
maintenance	bosinst_data, lpp_source, spot
diag	bosinst_data, spot
disk	None

The term resource here does not necessarily mean NIM resource (pssplpp is not a NIM resource).

14. Invoke the `mknimclient` wrapper:

```
mknimclient -l <node_list>
```

Where `<node_list>` is a list of nodes that have this node as boo/install server.

This wrapper creates NIM client definitions on the node. It also defines routing if the boot/install server is not the CWS and the node is on a different subnet than the CWS.

- Invoke the `mkconfig` wrapper with no arguments.

  This wrapper creates the /tftpboot/<reliable_hostname>.config_info file for each client node that has the bootp_response attribute not set to disk.

- Invoke the `mkinstall` wrapper with no arguments.

  This wrapper creates the /tftpboot/<reliable_hostname>.install_info file for each boot/install/client that doesn't have the bootp_response attribute set to disk.

- Invoke the `export_clients` wrapper with no arguments.

  This wrapper exports all necessary file systems on the node to the boot/install clients.

- Call the `allnimres` wrapper:

  ```
  allnimres -l <node_list>
  ```

  Where `<node_list>` is a list of nodes that have this node as boot/install server.

This wrapper allocates all necessary NIM resources on the NIM master for the boot/install clients.

- Remove authentication tickets.

2.2.3 NIM operations

In this section, we cover some useful NIM commands.

2.2.3.1 Resetting a NIM object

When a machine object is not in the “ready for a NIM operation” state (Cstate attribute), and you want to put it in that state so that you can perform operations on the machine object, execute:

```
nim -Fo reset <machine_object>
```

For example, you want to deallocate the SPOT for the machine object sp5n13 to be able to perform software updates to it, but the system complains with the following messages:

```
0042-001 nim: processing error encountered on "master":
warning:
0042-302 m_dealloc_spot: the state of "sp5n13" prevents this operation
from succeeding. Use the "reset" operation to correct its state then
retry the intended operation.
```

Let us see the state of sp5n13 by executing:

```
lsnim -l sp5n13
```

The output is:

```
sp5n13:
    class          = machines
    type           = standalone
    platform       = rs6k
    netboot_kernel = mp
    if1            = spnet_en1 sp5n13 02608CE8FCB3 ent
    cable_type1    = bnc
    Cstate         = BOS installation has been enabled
    prev_state     = ready for a NIM operation
    Mstate         = not running
    boot           = boot
    bosinst_data   = 13_noprompt
    lpp_source     = lppsource_aix432
    mksysb         = mksysb_3
    nim_script     = nim_script
    script         = psspscript
    spot           = spot_aix432
```

```
        cpuid          = 00085138A400
        control        = master
        err_info       = 0042-302 m_dealloc_spot: the state of "sp5n13"
    prevents this operation from succeeding. Use the "reset" operation to
    correct its state then retry the intended operation.
```

The `Cstate` attribute is not `ready for a NIM operation`, which is why you cannot perform NIM operations. Let us also look at the err_info attribute, which shows the error from the last unsuccessful NIM operation on this object.

Now, let us reset the NIM machine object:

```
nim -Fo reset sp5n13
```

The output of the `lsnim -l sp5n13` command is:

```
sp5n13:
     class          = machines
     type           = standalone
     platform       = rs6k
     netboot_kernel= mp
     if1            = spnet_en1 sp5n13 02608CE8FCB3 ent
     cable_type1    = bnc
     Cstate         = ready for a NIM operation
     prev_state     = BOS installation has been enabled
     Mstate         = not running
     bosinst_data   = 13_noprompt
     lpp_source     = lppsource_aix432
     mksysb         = mksysb_3
     script         = psspscript
     spot           = spot_aix432
     cpuid          = 00085138A400
     control        = master
     Cstate_result = reset
```

The `Cstate` attribute is now `ready for a NIM operation`, so you can perform operations on this machine object

2.2.3.2 Deallocating NIM resources

When you want to deallocate one resource from a machine object, do the following:

To deallocate one resource object from a machine object, execute:

```
nim -o deallocate -a <resource_name>=<resource_object> <machine_object>
```

For example, to deallocate mksysb_3 resource object sp5n13, execute:

```
nim -o deallocate -a mksysb=mksysb_3 sp5n13
```

Execute the command:

```
lsnim -l sp5n13
```

The output now shows that the resource object is no longer allocated:

```
sp5n13:
     class          = machines
     type           = standalone
     platform       = rs6k
     netboot_kernel = mp
     if1            = spnet_en1 sp5n13 02608CE8FCB3 ent
     cable_type1    = bnc
     Cstate         = ready for a NIM operation
     prev_state     = BOS installation has been enabled
     Mstate         = not running
     bosinst_data   = 13_noprompt
     lpp_source     = lppsource_aix432
     script         = psspscript
     spot           = spot_aix432
     cpuid          = 00085138A400
     control        = master
     Cstate_result  = reset
```

To deallocate all NIM resources from a machine object, execute:

```
nim -o deallocate -a subclass=all <machine_object>
```

For example, you want to deallocate all the resource objects from node sp5n13. To perform this, execute:

```
nim -o deallocate -a subclass=all sp5n13
```

And if you execute:

```
lsnim -l sp5n13
```

The output shows that the resource objects are not longer allocated:

```
sp5n13:
     class          = machines
     type           = standalone
     platform       = rs6k
     netboot_kernel = mp
     if1            = spnet_en1 sp5n13 02608CE8FCB3 ent
     cable_type1    = bnc
     Cstate         = ready for a NIM operation
     prev_state     = BOS installation has been enabled
     Mstate         = not running
     cpuid          = 00085138A400
     Cstate_result  = reset
```

2.2.3.3 Allocating NIM resources

To allocate NIM resources to a machine object, issue the command:

```
nim -o allocate -a <resource_name>=<resource_object> <machine_object>
```

For example, you want to allocate the lppsource_aix432 resource object to node sp5n13, so execute:

```
nim -o allocate -a lpp_source=lppsource_aix432 sp5n13
```

After issuing the command:

```
lsnim -l sp5n13
```

The output shows that the resource object lppsource_aix432 is allocated:

```
sp5n13:
     class          = machines
     type           = standalone
     platform       = rs6k
     netboot_kernel  = mp
     if1            = spnet_en1 sp5n13 02608CE8FCB3 ent
     cable_type1    = bnc
     Cstate         = ready for a NIM operation
     prev_state     = BOS installation has been enabled
     Mstate         = not running
     lpp_source     = lppsource_aix432
     cpuid          = 00085138A400
     control        = master
     Cstate_result  = reset
```

2.2.3.4 Installing additional filesets on the SPOT

To install additional software on the SPOT, do the following:

```
nim -o cust -a lpp_source=<lppsource> -a installp_flags=<instp_flgs> -a
filesets=<filesets>|ALL <spot_name>
```

For example, to install the fileset bos.terminfo on spot_aix432, execute:

```
nim -o cust -a lpp_source=lppsource_aix432 -a installp_flags=acXg -a
filesets=bos.terminfo spot_aix432
```

After applying software to SPOT, verify the operation by reviewing the log:

```
nim -o showlog <spot_name>
```

For example, to see the log of the last operation, execute:

```
nim -o showlog spot_aix432
```

2.2.3.5 Updating the SPOT

To update all the filesets on the SPOT, do the following:

```
nim -o cust -a lpp_source=<lppsource> -a fixes=update_all <spot_name>
```

For example, to update all filesets on the SPOT with all the fixes that are on the lppsource_aix432 resource object, issue:

```
nim -o cust -a lppsource=lppsource_aix432 -a fixes=update_all
spot_aix432
```

2.2.3.6 Other useful SPOT operations

To uninstall software from the SPOT, execute:

```
nim -o maint -a installp_flags=u -a filesets=<filesets> <spot_name>
```

To clean up installation failures on the SPOT, execute:

```
nim -o maint -a installp_flags=C <spot_name>
```

To commit applied software on the SPOT, execute:

```
nim -o maint -a installp_flags=c -a filesets=<fileset>| ALL <spot_name>
```

To reject applied software on the SPOT, execute:

```
nim -o maint -a installp_flags=r -a filesets=<fileset>| ALL <spot_name>
```

To list the software installed on the SPOT, execute:

```
nim -o lslpp <spot_name>
```

To see if a fileset is installed, execute

```
nim -o lslpp -a filesets=<fileset_name> <spot_name>
```

2.2.3.7 Perform software checking

To perform installation-checking software on the SPOT, issue:

```
nim -o lppchk <spot>
```

For example:

```
nim -o lppchk spot_aix432
```

To validate that all necessary SIMAGES are on lpp_source, issue:

```
nim -o check <lppsource>
```

To see if the attribute simages is set to `yes` for the <lppsource>, execute:

```
lsnim -l <lppsource>
```

2.2.3.8 Checking the SPOT and recreating new boot images

To check the SPOT and create boot images, issue:

```
nim -Fo check <spot_name>
```

For example, to check the SPOT and re-create the boot images, execute:

```
nim -Fo check spot_aix432
```

2.2.3.9 Analyzing resource allocation

To analyze attributes for all resources, execute:

```
lsnim -a <attribute_name>
```

To analyze an object, execute:

```
lsnim -l <object_name>
```

To analyze a class object, execute:

```
lsnim -c <class_name>
```

2.2.3.10 Recreating the SPOT

To re-create the SPOT, perform the following steps:

1. First, confirm that the resource object is not allocated, so execute:

   ```
   lsnim -a alloc_count <spot_object>
   ```

2. If alloc_count is not 0, there are machine objects allocating this resource object. Execute the following command to list them:

   ```
   lsnim -a spot
   ```

3. See which machine objects have the SPOT object allocated, and for each one execute:

   ```
   nim -Fo reset <machine_object>
   nim -o deallocate -a spot=<spot_object> <machine_object>
   ```

To confirm that alloc_count is 0, again execute:

```
lsnim -a alloc_count <spot_object>
```

4. Remove the resource object with the following command:

   ```
   nim -o remove <spot_object>
   ```

5. Now, re-create the SPOT object by issuing:

   ```
   mknimres -l <node_number>
   ```

2.2.3.11 Recreating NIM master

To re-create the NIM master, issue:

```
delnimmast -l <node_number>
setup_server
```

2.2.4 setup_server problem determination

The problems with setup_server are usually indicated by messages that appear while it is running. To isolate the problem, examine the messages and identify which operation within setup_server failed. Figure 19 shows a sample output of setup_server. The figure shows that the setup_server operation was incomplete; it could not complete the `mknimmast` wrapper because it could not find the hostname for a node. This failure also caused `mknimres` to fail.

```
setup_server command results from sp4cw0
-------------------------------------------------------
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.
/etc/auto/startauto: The automount daemon is already running on this
system.
setup_CWS: Control Workstation setup complete.
mknimmast: Node 0 (sp4en0) already configured as a NIM master.
create_krb_files: 0016-428: Cannot create the client srvtab file for
node number . No hostname information was found in the SDR.
create_krb_files: tftpaccess.ctl file and client srvtab files
created/updated on server node 0.
/spdata/sys1/install/pssp/bosinst_data_migrate.
mknimres: 0016-369 Error creating custom bosinst_data file for node 0 on
sp4en0.
setup_server: 0016-279 Failure of internally called command:
/usr/lpp/ssp/bin/mknimres; rc= 2.
setup_server: Processing incomplete (rc= 2).
```

Figure 19. Sample output of a failed setup_server operation

The most common causes of `setup_server` failures are:

- Kerberos problems
- SDR problems
- NIM problems

Refer to Chapter 3, “Security environment” on page 167 for resolving Kerberos-related problems.

Here, we describe how to identify SDR and NIM problems. If NIM problems are detected, they can be fixed using NIM commands for those of you comfortable with NIM operations, or with higher-level SP commands for the rest of us.

2.2.4.1 SDR problem determination

A common problem with the SDR is that the information in it is not correct. But you should also check the /etc/SDR_dest_info file and see if it is pointing to the correct partition IP address. If setup_server complains about the SDR, first make sure that all the information is correct. To do this, the following commands are useful:

- `splstdata -b`
- `splstdata -a`
- `splstdata -n`
- `splstdata -s`
- `splstdata -e`

These commands extract information from the SDR and display it on the screen. Figure 20., “Sample output of the splstdata command” on page 43 shows the output of the `splstdata -n` command. Check whether the IP addresses of the nodes are correct and whether the reliable and initial hostnames are correct. The IP addresses, or the hostnames, must have corresponding entries in the /etc/hosts file for proper name resolution. If other mechanisms, such as DNS are used, they should be able to do the name resolution using the initial or reliable hostnames. The output of these commands should not have phantom entries for nonexistent nodes.

```
node# frame# slot# slots  initial_hostname  reliable_hostname
dcehostname default_route   processor_type processors_installed
description
----------------------------------------------------------------------
   5    1    5    1 sp4n05.msc.itso.  sp4n05.msc.itso.  ""
       192.168.4.130          UP                  1 66_Mhz_PWR2_Thin
   6    1    6    1 sp4n06.msc.itso.  sp4n06.msc.itso.  ""
       192.168.4.130          UP                  1 66_Mhz_PWR2_Thin
   7    1    7    1 sp4n07.msc.itso.  sp4n07.msc.itso.  ""
       192.168.4.130          UP                  1 66_Mhz_PWR2_Thin
   8    1    8    1 sp4n08.msc.itso.  sp4n08.msc.itso.  ""
       192.168.4.130          UP                  1 66_Mhz_PWR2_Thin
   9    1    9    1 sp4n09.msc.itso.  sp4n09.msc.itso.  ""
       192.168.4.130          MP                  4 332_MHz_SMP_Thin
  10    1   10    1 sp4n10.msc.itso.  sp4n10.msc.itso.  ""
       192.168.4.130          MP                  4 332_MHz_SMP_Thin
  11    1   11    2 sp4n11.msc.itso.  sp4n11.msc.itso.  ""
       192.168.4.130          UP                  1 66_MHz_PWR2_Wide
  13    1   13    2 sp4n13.msc.itso.  sp4n13.msc.itso.  ""
       192.168.4.130          UP                  1 66_MHz_PWR2_Wide
  15    1   15    2 sp4n15.msc.itso.  sp4n15.msc.itso.  ""
       192.168.4.130          UP                  1 66_MHz_PWR2_Wide
```

Figure 20. Sample output of the splstdata command

The Syspar_map SDR class is very important for setup_server. To retrieve the information from the Syspar_map SDR class, execute:

```
SDRGetObjects Syspar_map
```

Figure 21., “Sample output of Syspar_map class” on page 44 shows the Syspar_map file of our 10 SP nodes. The fifth column indicates if a node is present on that slot (that is, if there is a serial connection between the frame supervisor and the node). The third column is the node number and the first and second columns show the system partition name and system partition IP address information. If an entry is missing, setup_server will complain about the SDR.

```
syspar_name syspar_addr node_number switch_node_number used node_type
1=sp4en0 2=192.168.4.130 3=1 4=0 5=1 6=standard
1=sp4en0 2=192.168.4.130 3=2 4=1 5=0 6=standard
1=sp4en0 2=192.168.4.130 3=3 4=2 5=0 6=standard
1=sp4en0 2=192.168.4.130 3=4 4=3 5=0 6=standard
1=sp4en0 2=192.168.4.130 3=5 4=4 5=1 6=standard
1=sp4en0 2=192.168.4.130 3=6 4=5 5=1 6=standard
1=sp4en0 2=192.168.4.130 3=7 4=6 5=1 6=standard
1=sp4en0 2=192.168.4.130 3=8 4=7 5=1 6=standard
1=sp4en0 2=192.168.4.130 3=9 4=8 5=1 6=standard
1=sp4en0 2=192.168.4.130 3=10 4=9 5=1 6=standard
1=sp4en0 2=192.168.4.130 3=11 4=10 5=1 6=standard
1=sp4en0 2=192.168.4.130 3=12 4=11 5=0 6=standard
1=sp4en0 2=192.168.4.130 3=13 4=12 5=1 6=standard
1=sp4en0 2=192.168.4.130 3=14 4=13 5=0 6=standard
1=sp4en0 2=192.168.4.130 3=15 4=14 5=1 6=standard
1=sp4en0 2=192.168.4.130 3=16 4=15 5=0 6=standard
```

Figure 21. Sample output of Syspar_map class

If nothing obvious is found by checking the SDR information, the easiest thing to do is to isolate the failing wrapper. Then, find out which SDR command is executed in the wrapper. Identify the command in the wrapper that is trying to get information from the SDR. If you can locate the command, your search has ended. The SDR command parameter tells you which object class of the SDR is used. Now, manually check the SDR object class for missing or wrong information. Use the `SDRGetObjects` command to retrieve and check the information from the SDR. In our case studies, we explain this procedure.

2.2.4.2 NIM problems

Most setup_server problems are due to NIM. The most common NIM problems are related to SPOT and the lppsource, which are NIM resources. Other NIM resources are mksysb, prompt, noprompt, migrate, and psspscript.

Based on the bootp_response selected, setup_server creates these resources and allocates them. The general type of failure is the inability to create or allocate the resources.

For NIM problem determination, it is necessary to understand which resources are not created or failed to be allocated. The setup_server messages clearly indicate the failing item most of the time.

Common reasons for failure to create the NIM resources are:

- Missing filesets on lppsource.
- Not enough space in the /tftpboot or /var file systems.
- Not enough space on the /spdata/sys1/install/<lppsource> directory.
- Permissions not set up properly.
- Resources that are needed for resource creation are not in the correct state.
- Operations are needed to be performed on a resource, but the resource is allocated.
- The /etc/niminfo file is incorrect or corrupted.

The allocation problem occurs if the resource Rstate is not in *ready for use* state. Therefore, if the messages indicate a resource allocation problem, all the resources for the Rstate should be checked.

There is a log file called spot.out.<pid> in the /tmp file system. Each time SPOT is created, this file also gets created. This log is useful for problem determination for SPOT failures.

Let us now look at each wrapper and see what should be expected from each and what the common problems are.

The following problems are common to almost all wrappers.

- Kerberos problems
- Communication problems
- SDR corruption
- A corrupted or missing /etc/niminfo file
- A corrupted NIM master database
- The Cstate attribute of the NIM master machine object not *ready for a NIM operation*

delnimclient
Success:

- The NIM machine objects defined in the wrapper are deleted from the NIM master database.
- The NIM custom bosinst_data resource objects that were created for this machine object are removed from the NIM master database. The resource objects are:
  - <node_number>_noprompt

    - <node_number>_migrate
- The following files are deleted:
    - <node_number>.noprompt
    - <node_number>.migrate

Specific problems of this wrapper:

- The <node_number>_noprompt resource object is allocated by a machine object. This problem is very unlikely to occur. It will only arise if you are performing NIM operations outside setup_server and wrappers.
- The <node_number>_migrate resource object is allocated by a machine object. This problem is very unlikely to occur. It will only arise if you are performing NIM operations outside setup_server and wrappers.

mknimmast
Success:

1. The following filesets are installed:
    - bos.sysmgt.nim.spot
    - bos.sysmgt.nim.master
2. The nimd and nimesis subsystems are added to the SRC.
3. The nimesis subsystem is started.
4. An entry for the nimesis subsystem is added to the /etc/inittab file.
5. The /etc/niminfo file is created.
6. The following objects are created:
    - The master machine object
    - The nim_script resource object (type is script)
    - The boot resource object (type is boot)
    - The spnet_en0 network object
7. The interface attribute for en0 is added to the master machine object.

Specific problems of this wrapper:

If the master being defined is the CWS:

- One of the following filesets or prerequisites is not on the lppsource directory (/spdata/sys1/install/<aixlvl>/lppsource):
    - bos.sysmgt.nim.master
    - bos.sysmgt.nim.spot

- The .toc file on lppsource is corrupted or not consistent with the filesets on lppsource.
- File system full problems.

  If the master being defined is not the CWS:
- One of the following filesets or prerequisites is not on the lppsource directory (/spdata/sys1/install/<aixlvl>/lppsource):
- bos.sysmgt.nim.master
- bos.sysmgt.nim.spot
- The .toc file on lppsource is corrupted or not consistent with the filesets on lppsource.
- File system full problems.
- The .toc file permission is not correct (should be rw-r--r--).
- The fileset permissions are not correct (should be rw-r--r--).
- The lppsource directory is not exported from the CWS.
- There is an NFS server problem on the CWS.
- There is an NFS client problem on the node being defined as NIM master.

mknimint
Success:

- If it is being run on the CWS and the CWS has only one Ethernet adapter:
  - No operation is invoked (the network object was already added by `mknimmast`).
- If it is being run on the CWS and the CWS has more than one Ethernet adapter:
  - The network objects are added to the NIM master database of the CWS (one for each additional Ethernet adapter).
  - The interface attributes are added to the master machine object in the NIM master database of the CWS (one for each additional Ethernet adapter).
- If it is not being run on the CWS, the boot/install server has only one Ethernet, and the CWS has only one Ethernet and no token-ring adapter:
  - The CWS machine object is added to the NIM master database.
- If it is not being run on the CWS, the boot/install server has only one Ethernet, and the CWS has more than one Ethernet or token-ring adapter:

    - All the CWS networks (token-ring, Ethernet) are added to the NIM master database.
    - The CWS machine object is added to the NIM master database.
    - The interface attributes are added to the CWS machine object on the NIM master database.
- If it is not being run on the CWS, the boot/install server has more than one Ethernet, and the CWS has only one Ethernet and no token-ring adapter:
    - The network objects are added to the NIM master database of the boot/install server (one for each additional Ethernet adapter on the boot/install server).
    - The interface attributes are added to the master machine object in the NIM master database (one for each additional Ethernet adapter on the boot/install server).
    - The CWS machine object is added to the NIM master database.
- If it is not being run on the CWS, the boot/install server has more than one Ethernet, and the CWS has more than one Ethernet or token-ring adapter:
    - The network objects are added to the NIM master database of the boot/install server (one for each additional Ethernet adapter).
    - The interface attributes are added to the master machine object in the NIM master database (one for each additional Ethernet adapter).
    - All the CWS networks (token-ring, Ethernet) are added the NIM master database.
    - The CWS machine object is added to the NIM master database.
    - The interface attributes are added to the CWS machine object on the NIM master database.

Specific problems of this wrapper:

If the `mknimint` wrapper being run is for the CWS:

- Make sure all the Ethernet network interfaces have been configured properly.
- Make sure the IP and subnet masks are correctly configured in the Adapter SDR class.

If the `mknimint` being run is not for the CWS:

- Make sure all the Ethernet network interfaces on the boot/install server have been configured.

- Make sure the IP and subnet masks are correctly configured in the Adapter SDR class.
- Make sure hostname resolves to the same IP address on the CWS and on the boot/install server.
- The Cstate attribute of the CWS machine object on the NIM master is not ready for a NIM operation.
- The Mstate attribute of the CWS machine object on the NIM master is not running.
- Make sure all the Ethernet network interfaces and token-ring network interfaces on the CWS have been configured and their IP addresses can be resolved on the BIS node.

mknimres
Success:

1. The pssp_script file is created in the /spdata/sys1/install/pssp directory from the original file in the /usr/lpp/ssp/install/bin directory.
2. The psspscript resource object of type script is created in the NIM master database.
3. The bosinst_data file is created in the /spdata/sys1/install/pssp directory from the template file in the /usr/lpp/ssp/install/config directory.
4. The bosinst_data_noprompt file is created in the /spdata/sys1/install/pssp directory from the template file in the /usr/lpp/ssp/install/config directory.
5. The bosinst_data_migrate file is created in the /spdata/sys1/install/pssp directory from the template file in the /usr/lpp/ssp/install/config directory.
6. The prompt resource object of type bosinst_data is created in the NIM master database.
7. The noprompt resource object of type bosinst_data is created in the NIM master database.
8. The migrate resource object of type bosinst_data is created in the NIM master database.
9. If there are nodes with bootp_response equal to install or migrate, and the install disk is not the hdisk0 alone:
    - The <node_number>.noprompt file is created in the /spdata/sys1/install/pssp directory based on the template file in the /usr/lpp/ssp/install/config directory.

- The <node_number>.migrate file is created in the /spdata/sys1/install/pssp directory based on the template file in the /usr/lpp/ssp/install/config directory.
- The <node_number>_noprompt resource object of type bosinst_data is created in the NIM master database.
- The <node_number>_migrate resource object of type bosinst_data is created in the NIM master database.

10. For each lppsource needed by the nodes:

    If the node performing the action is the CWS:

    - Create the lppsource_<aixlvl> resource of type lpp_source in the NIM master database and define master as the server of this resource.
    - NFS exports the /spdata/sys1/install/<aixlvl>/lppsource directory with read-only permissions to everybody.

    If the node performing the action is not the CWS:

    - Create the lppsource_<aixlvl> resource of type lpp_source on the NIM master database, and define the CWS as the server of this resource.

11. For each mksysb needed by the nodes:

    If the node performing the action is the CWS:

    - Create the mksysb_<seq> resource of type mksysb in the NIM master database.

    If the node performing the action is not the CWS:

    - Create/expand the /spdata/sys1/install/images file system.
    - Copy the mksysb image file from the CWS.
    - Create the mksysb_<seq> resource of type mksysb in the NIM master database.

12. For each SPOT needed by the nodes:

    If the node performing the action is the CWS:

    - Create the spot_<aixlvl> resource of type SPOT on the NIM master database from the lppsource_<aixlvl> resource.
    - Create boot images in the /tftpboot directory for NIM clients.

    If the node performing the action is not the CWS:

    - Create the /spdata/sys1/install/spot_<aixlvl> file system.
    - Create the spot_<aixlvl> resource of type SPOT in the NIM master database from the lppsource_<aixlvl> resource.

8. Create boot images in the /tftpboot directory for NIM clients.

13. For each PSSP code level needed by the nodes:

    If the node performing the action is not the CWS:

    - Create/expand the /spdata/sys1/install/pssplpp file system.
    - Copy the pssp.installp, rsct.basic and rsct.clients from the CWS.

Specific problems of this wrapper:

The first thing this wrapper does is remove all the resources (except boot resources) that have the Rstate attribute not in "ready to use" state. The problems that can arise from here are the following:

1. The Rstate attribute for the nim_script resource is not "ready to use." Since the nim_script resource cannot be removed, we have two choices: to re-create the NIM master, or to use the following command:

   ```
   /usr/lpp/bos.sysmgt/nim/methods/m_chattr -a Rstate=available
   nim_script
   ```

2. The Cstate attribute of the master machine objects is not "ready for a NIM operation." In this case, execute:

   ```
   nim -Fo reset master
   ```

3. The Rstate attribute for a resource is not "ready to use," and the resource is allocated. In this case we cannot remove the resource because it is allocated, and we cannot deallocate the resource because it is not *ready to use*. So, we must perform the following actions:

   - Execute:

     ```
     /usr/lpp/bos.sysmgt/nim/methods/m_chattr -a Rstate=available
     <resource_name>
     ```

   - Deallocate this resource from all machine objects.
   - For lpp_source and SPOT resource objects, perform a check on the resource. Execute:

     ```
     nim -o check <resource_name>
     ```

   - If, after this, the objects are not on the correct Rstate attribute, remove them by executing:

     ```
     nim -o remove <resource_name>
     ```

4. The lppsource_<aixlvl> resource object is locked and the Rstate is not "ready for use." If this is true, another operation is being performed on the lppsource_<aixlvl> objects, so wait for the PID identified by the lock

attribute to finish. If the process that locked the lppsource object is not running, but the resource is shown as locked, you must perform the following steps:

```
stopsrc -s nimesis
startsrc -s nimesis
nim -o check lppsource<aixlvl>
```

5. For a psspscript resource creation failure, the causes can be:
   - The pssp_script file does not exist on the /usr/lpp/ssp/config/bin directory.
   - The file system that holds the directory /spdata/sys1/install/pssp does not have enough free space.
6. Creation of the resources of the type bosinst_data can fail if:
   - The bosinst_data template files do not exist on the /usr/lpp/ssp/config/config directory.
   - The file system that holds the directory /spdata/sys1/install/pssp does not have enough free space.
7. Creation of the lpp_source resource can fail for different reasons, depending on whether this wrapper is being invoked for the CWS or for a boot/install server other than the CWS.

If the `mknimres` wrapper being performed is for the CWS, creation of the resource can fail if the following statements are true:

- The file system on which the lppsource_<aixlvl> directory resides is not a jfs file system.
- The lppsource_name attribute in Volume_Group for the select volume group is not correct. The lppsource_name attribute is used to define the location of the resource object (/spdata/sys1/install/<lppsource_name>/lppsource) and is used in resource object names (lppsource_<lppsource_name>, spot_<lppsource_name>), so use only alphabetic characters in the lppsource_name attribute.
- The lppsource directory does not contain the required filesets. Verify that the /spdata/sys1/install/<lppsource_name>/lppsource directory has REQUIRED_SIMAGES_<ver_rel> defined in the /usr/lpp/bos.sysmgt/nim/methods/c_sh_lib file as shown in Figure 22 on page 53 (REQUIRED_SIMAGES should be used for AIX 4.3 lppsource).

```
REQUIRED_SIMAGES="\
        bos \
        bos.64bit \
        bos.up \
        bos.mp \
        bos.net \
        bos.diag \
        bos.sysmgt \
        bos.terminfo \
        bos.terminfo.all.data \
        devices.base.all \
        devices.buc.all \
        devices.common.all \
        devices.graphics.all \
        devices.mca.all \
        devices.rs6ksmp.base \
        devices.scsi.all \
        devices.sio.all \
        devices.sys.all \
        devices.tty.all \
        xlC.rte"
```

Figure 22. Required images in the lppsource directory

The lpp_source resource object will be removed after creation if the simages attribute of this resource is not set to yes.

We recommend that you put all device filesets from the AIX media in the lppsource directory, especially those supporting the architecture of your nodes. Device filesets missing from the lppsource directory will be missing from the SPOT, and you will not be able to install, migrate, or network boot your nodes. For instance:

- If devices.rs6ksmp.base is missing, you will not be able to perform allocation operations to High nodes.
- If boot adapter support is missing, you will not be able to network boot nodes.
- If devices.sio.sa fileset is missing, you will be stopped in LED C45 during the network boot of a node.

If the mknimres wrapper being run is not on the CWS, the creation of the resource can fail if the following statements are true (in addition to the reasons for on the CWS):

- Read permissions on the filesets in the lppsource directory are not granted to others on the CWS.
- Read permissions for the .toc file in the lppsource directory are not granted to others on the CWS.
- The lppsource is not being NFS-exported on the CWS.

- There is an NFS server problem on the CWS.
- There is an NFS client problem on the boot/install server.
- The Cstate attribute of the CWS machine object is not "ready for a NIM operation".

  You should be aware that if you use symbolic links to define the lppsource directory, for instance:

  ```
  ln -s /spdata/sys1/install/aix432 /spdata/sys1/install/default
  ```

  And define <lppsource_name> as <default>, and then change the link to somewhere else, you will certainly be faced with a nightmare, because once the resources are created, they are not changed by the changes you make on the system.

  After placing additional filesets or PTFs in the lppsource directory on the CWS, change the permissions to `rw-r--r--` and run `inutoc` to re-create the .toc file. You should execute `nim -Fo check <lppsource_object>` and verify the simages attribute on the resource object. After this, update the SPOT resource to be at the same PTF level as lppsource. See 2.2.3.5, "Updating the SPOT" on page 39 for details on how to update the SPOT.

  The creation of the SPOT resource can fail for different reasons, depending on whether this wrapper is being invoked for the CWS or for a boot/install server that other than the CWS.

  To be sure that there are no prerequisite failures before starting the creation of the SPOT resource, issue

  ```
  installp -ap -d /spdata/sys1/install/<lppsource_name>/lppsource ALL
  ```

  If the mknimres being performed is for the CWS, the creation of the resource can fail if the following statements are true:

- The file system on which the SPOT directory resides is not a jfs file system.
- The lppsource directory does not contain the required filesets. Verify that the /spdata/sys1/install/<lppsource_name>/lppsource directory on the CWS has the DEFAULT_SPOT_OPTIONS_<ver_rel> defined in the /usr/lpp/bos.sysmgt/nim/methods/c_sh_lib file as shown on Figure 23 on page 55.

```
DEFAULT_SPOT_OPTIONS="\
        bos.sysmgt.nim.client \
        bos.sysmgt.nim.spot \
        bos.64bit \
        bos.up \
        bos.mp \
        bos.net.nfs.client \
        bos.net.tcp.client \
        bos.net.tcp.smit \
        bos.diag \
        bos.sysmgt.sysbr \
        bos.sysmgt.smit \
        bos.terminfo \
        devices.all"
```

Figure 23. Required images on lppsource for the SPOT

The SPOT resource object will be removed after creation if the Rstate attribute of this resource is not "ready for use".

- The file system that holds the SPOT directory (/spdata/sys1/install/<lppsource_name>/spot) does not have enough space to create the SPOT.
- The file system that holds the /tftpboot directory does not have enough space for the boot image files.
- There is a problem updating the SPOT. After the SPOT is created, the AIX PTFs on lppsource are applied to the SPOT, so if there is an error updating the SPOT so that the Rstate is not "ready to use", the SPOT will be removed. Look in /tmp/spot.update.out.<pid> or by using the `nim -o showlog` command for problems in updating the SPOT.

  In addition to the problems listed for the CWS, the mknimres wrapper may have the following problems when run on a BIS other than the CWS:

- There is not enough space on the volume group to create the SPOT logical volume.
- Read permissions on the filesets on the lppsource directory are not granted to others on the CWS.
- Read permissions on the .toc file on the lppsource directory are not granted to others on the CWS.
- The lppsource is not being NFS-exported on the CWS.
- There is an NFS server problem on the CWS.
- There is an NFS client problem on the boot/install server.
- The Cstate attribute of the CWS machine object is not "ready for a NIM operation".

When the SPOT is created, a log file (/tmp/spot.out.<pid>, where <pid> is the process ID of this wrapper) is also created. This log file contains the output NIM command with verbose set to 5, so this is a good place to start looking for the cause of problems.

If, after doing this, you have not found a way to solve the problem, you can create the SPOT manually and investigate the problem by executing the following command:

```
nim -o define -t spot -a server=master -a source=lppsource_<aixlvl>
-a location=/spdata/sys1/install/<aixlvl>/spot spot_<aixlvl>
```

In this way, the SPOT resource will not be removed if the Rstate attribute is not "ready for use."

After placing additional filesets or PTFs on nodes, put them on the lppsource directory on the CWS, change the permissions to `rw-r--r--`, and run `inutoc` to re-create the .toc file. Then issue:

```
nim -Fo check <lppsource_object>
```

Verify the simages attribute on the resource object. Then update the SPOT resource to be at the same PTF level as lppsource and the nodes.

The creation of the mksysb resource can fail for different reasons, depending on whether this wrapper is being invoked for the CWS or for a boot/install server that is not the CWS.

If this wrapper is being invoked for the CWS, and any of the following conditions are true, this wrapper will fail:

- If the image_install attribute for the selected volume group in the Volume_Group SDR class does not exist
- Verify that /spdata/sys1/install/images/<image_install> exists on the CWS.
- If /spdata/sys1/install/images/<image_install> is not a mksysb image (not in backup format)

  Verify that it is in fact an mksysb image and that the image.data file exists inside that image by executing the following command:

  ```
  restore -xvf <image_file> ./image.data
  ```

  Then, examine the image.data file created in the current directory.

In addition to these reasons, if this wrapper is not being invoked for the CWS, and any of the following conditions are true, this wrapper will fail:

- The conditions stated for invocation for the CWS
- If there is not enough space on the rootvg volume group to create/extend the /spdata/sys1/install/images file system to hold the mksysb images

- The mksysb images are copied from the CWS.
- If the mksysb resource exists on the boot/install server, and the size of this image is not the same as on the CWS, and the mksysb resource is allocated

    To solve this problem, deallocate the mksysb resource from all the machine objects that are allocating it.

  Be aware that if you use the same mksysb image name for different versions, release, or maintenance level, you could have serious problems. For instance, suppose you initially put a mksysb image of AIX 4.3.1 on the image directory and created the mksysb resource. lppsource and SPOT are also AIX 4.3.1. After this, you put an AIX 4.3.2 image with the same name in the images directory, did not make any change to the SDR, and ran setup_server. The setup_server script does not create any new resource because the resource exists. During the allocation process, setup_server does not find any problem since this resource has the same attributes as SPOT with respect to version, release, and maintenance level (both are AIX 4.3.1). Finally, when installing the node, you get C52 LED and conflict symbols between the libc.a (in the SPOT) and the executable (from the mksysb image).

14. The mknimres wrapper could also fail if PSSP code is missing from the /spdata/sys1/install/pssplpp/PSSP-<ver>.<rel> directory. The pssp.installp fileset has to be in the directory. If the PSSP version is 3.1 or later, then rsct.basic and rsct.clients also have to be in the directory.

    This wrapper can fail if it is not being invoked for the CWS and there is not enough space on the rootvg volume group to create/expand the /spdata/sys1/install/pssplpp file system.

mknimclient

Success:

- The NIM machine object is created in the NIM master database.
- A routing attribute is added to the machine object if the boot/install server is not the CWS and the CWS is not on the same subnet as the node.

Specific problems of this wrapper:

- The NIM client is not on the same net as the boot/install server.
- The subnet masks are not correct on the Adapter SDR class and on the interface.
- The host name resolution is not given the same IP address on both the CWS and the boot/install server.

- There are interfaces not correctly configured on the boot/install server.

mkconfig
Success:

The <reliable_hostname>.config_info is created for each node that has a bootp_response set to customize, install, or migrate.

Specific problems of this wrapper:

There is not enough space on the file system that holds the /tftpboot directory.

mkinstall
Success:

The <reliable_hostname>.install_info is created for each node that has a bootp_response set to customize, install, or migrate.

Specific problems of this wrapper:

- There is not enough space on the file system that holds the /tftpboot directory.
- The nodes are not defined as NIM clients.

export_clients
Success:

- The directory /spdata/sys1/install/pssplpp is added to the NFS export file.
- The nodes with bootp_response set to customize, install, or migrate are added to the NFS export node list of the pssplpp directory.

create_krb_files
Success:

- The Kerberos service key files are created on /tftpboot for each client that has a bootp_reponse set to install, customize, or migrate.
- The /etc/tftpaccess.ctl file is created/updated.

Specific problems of this wrapper:

There is not enough space on the file system that holds the /tftpboot directory.

allnimres
Success:

- The resources are allocated based on the bootp_response attribute from the Node SDR class. The file and file systems associated with the location attribute from the resource object are exported.
- The /etc/inetd.conf is updated.
- The bos_inst operation is performed on the machine object class. This means that entries for the machine objects are placed in the /etc/bootptab file, the node boot image file (/tftpboot/<reliable_hostname>) is linked to the generic boot image file, and the niminfo file is created (/tftpboot/<reliable_hostname>.info).

Specific problems of this wrapper:

- The information from the SDR does not match any resource objects. For instance, the install_image attribute from the SDR is bos.obj.ssp.432 and there is no resource object of type mksysb that has the location of object attribute set to /spdata/sys1/install/images/bos.obj.ssp.432.
- The file pointed to by the location of object attribute from the resource object does not exist.
- The Cstate attribute of a machine object is not "ready for a NIM operation". In this case, reset the machine object with the command:
- `nim -Fo reset <node_name>`
- There are inconsistencies between the resources that are being allocated. The version, release, and mod attributes from the SPOT resource object are not the same as the mksysb resource. In this case, the information from the Volume_Group SDR class has to be corrected.
- A resource that is needed does not have the Rstate attribute set to "ready to run". First be sure that the resource is not "ready for use" because of some operation that is being performed. If so, wait for the operation to stop execution. If no operation is running, then do the following:
  1. If the resource is locked, confirm that the process locking the resource is not running by executing:

     ```
     stopsrc -s nimesis; startsrc -s nimesis
     ```

  2. Change the state of the object to `available` by executing:

     ```
     /usr/lpp/bos.sysmgt/nim/methods/m_chattr -a Rstate=available
     <resource_name>
     ```

  3. Deallocate this resource from all machine objects.
  4. For lpp_source and SPOT resource objects, perform a check on the resource. Execute:

     ```
     nim -o check <resource_name>
     ```

5. If, after this, the objects do not have the correct value for the Rstate attribute, remove them by executing `nim -o remove <resource_name>`, and add them back by invoking `mknimres -l <boot_install_server>`.

- The SPOT does not support the platform and kernel of the machine object. In the case of High nodes, you must install devices.rs6ksmp.base and adapter support filesets if they are missing on the SPOT. In the case of a chrp machine object (332 MHz, POWER3 based nodes, or S70 familiy of attached servers), make sure devices.*chrp* filesets are installed on the SPOT, and pci adapter support filesets are also installed.
- The boot image file is missing from /tftpboot/spot_<aixlvl>.<platform>.[um]p.ent, where <platform> is rs6k or chrp. In this case, device filesets are missing from the SPOT. So, copy all devices.* filesets from the AIX CD-ROM to the lppsource directory and update the SPOT by issuing:

  ```
  nim -o cust -a lpp_source=lppsource_<aixlvl> -a installp_flags=acXg
  -a filesets=ALL spot_<aixlvl>
  ```

  Then, check the log file or issue a `nim -o showlog` command to verify a correct execution.
- The lpp_source directory is not accessible to the client. This means that a route is missing. Therefore, there is a problem in the mknimclient wrapper and you should investigate the problem there.
- The lpp_source resource does not have the simages attribute set to yes. This means filesets are missing from lppsource.
- The file system that holds the /tftpboot directory is full.

2.2.5 Case studies

In this section, we provide case studies for setup_server problems.

2.2.5.1 SDR problem with setup_server

We configure one node to customize (Figure 24., "Case study—SDR problem with setup_server" on page 61 shows the topology of the RS/6000 SP), and while running setup_server, we get the output that is described in Figure 25 on page 62.

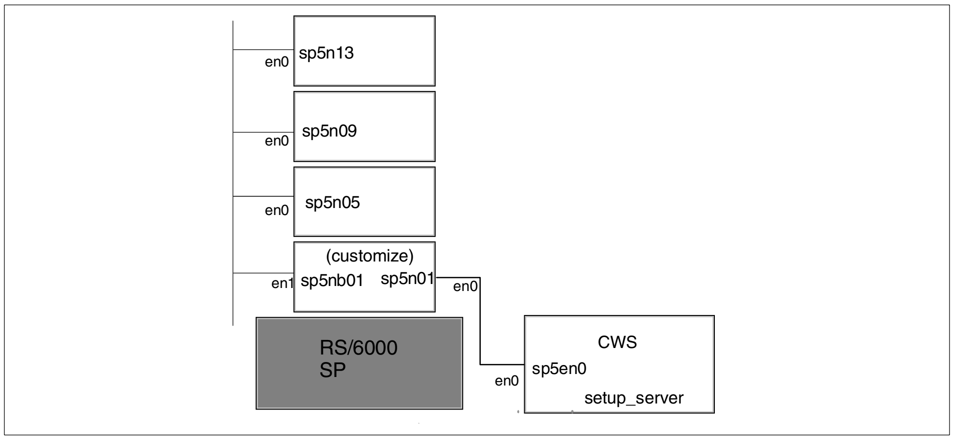


Figure 24. Case study—SDR problem with setup_server

```
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.
/etc/auto/startauto: The automount daemon is already running on this
system.
setup_CWS: Control Workstation setup complete.
mknimmast: Node 0 (sp5en0) already configured as a NIM master.
create_krb_files: 0016-428: Cannot create the client srvtab file for
node number . No hostname information was found in the SDR.
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/pssp/pssp_script.
mknimres: Copying /usr/lpp/ssp/install/config/bosinst_data.template to
/spdata/sys1/install/pssp/bosinst_data.
mknimres: Copying
/usr/lpp/ssp/install/config/bosinst_data_prompt.template to
/spdata/sys1/install/pssp/bosinst_data_prompt.
mknimres: Copying
/usr/lpp/ssp/install/config/bosinst_data_migrate.template to
/spdata/sys1/install/pssp/bosinst_data_migrate.
mknimres: 0016-369 Error creating custom bosinst_data file for node 0 on
sp5en0.
setup_server: 0016-279 Failure of internally called command:
/usr/lpp/ssp/bin/mknimres; rc= 2.
setup_server: Processing incomplete (rc= 2).
```

Figure 25. setup_server output

In this output, we see two errors. The first one is:

```
create_krb_files: 0016-428: Cannot create the client srvtab file for
node number. No hostname information was found in the SDR.
```

The second one is:

```
mknimres: 0016-369 Error creating custom bosinst_data file for node 0 on
sp5en0.
setup_server: 0016-279 Failure of internally called command:
/usr/lpp/ssp/bin/mknimres; rc= 2.
setup_server: Processing incomplete (rc= 2).
```

We must always start with the first error; so, let us analyze it.

There is enough space on the / and /tftpboot file systems, so there is no space problem, which seems to indicate a Kerberos or SDR problem.

We issue `k4list`. The output of this command is shown in Figure 26.

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

  Issued           Expires          Principal
Oct 26 08:02:53  Nov 25 08:02:53 krbtgt.MSC.ITSO.IBM.COM@MSC.ITSO.IBM.
COM
Oct 26 09:53:41  Nov 25 09:53:41  rcmd.sp5n01@MSC.ITSO.IBM.COM
Oct 26 10:45:11  Nov 25 10:45:11  hardmon.sp5en0@MSC.ITSO.IBM.COM
Oct 26 13:02:29  Nov 25 13:02:29  rcmd.sp5n05@MSC.ITSO.IBM.COM
Oct 26 13:02:29  Nov 25 13:02:29  rcmd.sp5n09@MSC.ITSO.IBM.COM
Oct 26 13:02:29  Nov 25 13:02:29  rcmd.sp5n13@MSC.ITSO.IBM.COM
```

Figure 26. k4list output

We now have a valid ticket-granting ticket.

Now let us run the following command:

```
rsh sp5en0 date
```

This command was successful, so we do not have a Kerberos problem on the CWS. Therefore, this must be an SDR problem.

Let us first see if the sdrd is running by issuing `lssrc -g sdr`. We get the output shown in Figure 27.

```
Subsystem         Group            PID     Status
 sdr.sp5en0       sdr              11358   active
```

Figure 27. lssrc output

The SDR daemon (sdrd) is running, so the next thing to do is to look at the /etc/SDR_dest_info file and see if it has the correct information. Refer to Figure 28., “SDR_dest_info file” on page 64.

```
default:192.168.5.150
primary:192.168.5.150
nameofdefault:sp5en0
nameofprimary:sp5en0
```

Figure 28. SDR_dest_info file

This file has the correct information, so the problem has to be associated with an SDR class. But what class should we look at?

We could debug the wrapper code. This is not a bad option, but we do not cover debugging code in this chapter. The other option is the one we now discuss.

First, we must get the process ID for the SDR daemon, in this case, 11358. Then, we open another aixterm window and change to directory /var/adm/SPlogs/sdr. The SDR log file has the following format:

sdrdlog.<partition-ip-addr>.<pid>

In our case, the log file is sdrdlog.192.168.1.150.11358, so in this aixterm window, we issue:

```
tail -f sdrdlog.192.168.1.150.11358
```

In the other aixterm window we do the following:

1. Turn sdrd debug on. The pid of the SDR daemon is 11358, so we issue:
   ```
   kill -HUP 11358
   ```
2. Re-execute the wrapper that failed by issuing:
   ```
   create_krb_file
   ```
3. Turn sdrd debug off. The pid of the sdrd is 11358, so execute:
   ```
   kill -HUP 11358
   ```

After completing these steps, the output on the aixterm window that is executing the tail command is shown in Figure 29., “sdrd log file” on page 65.

```
Received SIGHUP: turning debug logging ON: Wed Oct 21 13:54:55 1998
Sending part of class SP to TurboGetObjects client from address
192.168.5.150
Sending part of class Node to TurboGetObjects client from address
192.168.5.150
Sending part of class Adapter to TurboGetObjects client from address
192.168.5.150
Received SIGHUP: turning debug logging OFF: Wed Oct 21 13:54:57 1998
```

Figure 29. sdrd log file

The SP, Node, and Adapter SDR class were read during the execution of the `create_krb_files` wrapper, allowing us to restrict the search to these three classes.

Let us now read these three SDR classes by doing the following:

1. Issue:

   `SDRGetObjects SP`

   The output of this command is shown in Figure 30.

```
control_workstation cw_ipaddrs   install_image remove_image
primary_node ntp_config   ntp_server   ntp_version  amd_config
print_config print_id     usermgmt_config passwd_file  passwd_file_loc
homedir_server homedir_path filecoll_config supman_uid
supfilesrv_port spacct_enable spacct_actnode_thresh
spacct_excluse_enable acct_master  cw_has_usr_clients code_version
layout_dir   authent_server backup_cw    ipaddrs_bucw active_cw
sec_master   cds_server   cell_name    cw_lppsource_name cw_dcehostname
sp5en0     9.12.1.150:192.168.5.150: bos.obj.ssp.432 false     1
consensus    ""          3           true        false       ""
true       /etc/passwd  sp5en0      sp5en0      /home/sp5en0 true
102         8431        false               80 false
0 false       PSSP-3.1    ""          ssp         ""          ""
""            ""            ""            ""           aix432       ""
```

Figure 30. SP SDR class

- The CWS hostname is `sp5en0`.
- The IP addresses of the interfaces on the CWS are `9.12.1.150` and `192.168.5.150`.
- The default installation image is `bos.obj.ssp.432`.

- The NTP information is correct.
- The automount information is correct.
- The print management information is correct.
- The user management information is correct.
- The file collection information is also correct.
- Accounting is not configured.
- The CWS PSSP level is `3.1`.
- The lppsource on the CWS is `aix432`.
- DCE is not installed.

All information in this class looks correct, so we need to check the next class.

2. Issue:

   `SDRGetObjects Node node_number`

   The output of this command is shown in Figure 31.

```
node_number
1
5
9
13
""
```

Figure 31. Extract from the Node class

There is something wrong in this class: We have an object that does not contain any information for the node_number.

So how can we now solve this problem, since the object has no attributes? SDRDeleteObjects will not be able to delete the object.

We must do the following:

1. Issue `SDRArchive` to make a backup copy of the SDR.
2. Delete the empty line at the end of the file /spdata/sys1/sdr/partitions/<partition-ip-addr>/classes/Node.
3. Issue `stopsrc -g sdr`.
4. Issue `startsrc -g sdr`.

We run setup_server again and the problem is solved (both errors).

2.2.5.2 SPOT problem while running setup_server

We configure one node to install, and while running setup_server on the boot/install server (Figure 32 shows the topology of the RS/6000 SP), we get the output that is shown in Figure 33.

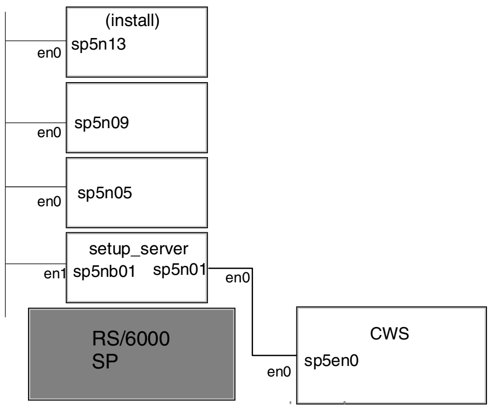


Figure 32. Case study 2—SPOT problem while running setup_server

```
mknimres: Copying /usr/lpp/ssp/install/bin/pssp_script to
/spdata/sys1/install/pssp/pssp_script.
mknimres: Copying /usr/lpp/ssp/install/config/bosinst_data.template to
/spdata/sys1/install/pssp/bosinst_data.
mknimres: Copying
/usr/lpp/ssp/install/config/bosinst_data_prompt.template to
/spdata/sys1/install/pssp/bosinst_data_prompt.
mknimres: Copying
/usr/lpp/ssp/install/config/bosinst_data_migrate.template to
/spdata/sys1/install/pssp/bosinst_data_migrate.
mknimres: Creating the spot resource spot_aix432.
mknimres: 0016-400 Creation of the spot_aix432 resource apparently
failed, since the Rstate
 is not 'ready for use'.  Check previous messages for possible cause.
mknimres: 0016-407 Refer to /tmp/spot.out.36674 for more debug
information.
```

Figure 33. setup_server output messages

The problem is in the mknimres wrapper. So, let us first see the Rstate of the resources by executing

```
lsnim -a Rstate
```

The output of this command is shown in Figure 34.

```
boot:
   Rstate = ready for use
nim_script:
   Rstate = ready for use
psspscript:
   Rstate = ready for use
prompt:
   Rstate = ready for use
noprompt:
   Rstate = ready for use
migrate:
   Rstate = ready for use
13_noprompt:
   Rstate = ready for use
13_migrate:
   Rstate = ready for use
lppsource_aix432:
   Rstate = ready for use
mksysb_1:
   Rstate = ready for use
```

Figure 34. Resource state

We see that the Rstate attribute of all the resources is `ready for use`. We also see that the following resources were created:

- boot
- nim_script
- psspscript
- prompt
- noprompt
- migrate
- 13_noprompt
- 13_migrate
- lppsource_aix432
- mksysb_1

Therefore, the problem is with the creation of the SPOT resource. Let us do basic checking on this resource:

1. There is enough space on the rootvg volume group to create the SPOT. We issue `lsvg rootvg` on the boot/install server, and then, on the CWS, execute:

   ```
   du -ks /spdata/sys1/install/aix432/spot
   ```

   There are enough physical partitions on the rootvg, so that is not why the mknimres wrapper failed.

2. So, let us see if the lpp_source resource has the required filesets.

   The filesets that are needed to create the SPOT are defined in the DEFAULT_SPOT_OPTIONS variable in the /usr/lpp/bos.sysmgt/nim/methods/c_sh_lib file. They are shown in Figure 35 for a system running AIX 4.3.2.

```
DEFAULT_SPOT_OPTIONS="\
        bos.sysmgt.nim.client \
        bos.sysmgt.nim.spot \
        bos.64bit \
        bos.up \
        bos.mp \
        bos.net.nfs.client \
        bos.net.tcp.client \
        bos.net.tcp.smit \
        bos.diag \
        bos.sysmgt.sysbr \
        bos.sysmgt.smit \
        bos.terminfo \
        devices.all"
```

Figure 35. Required SPOT filesets

3. To see which filesets are in the lppsource, we issue:

   ```
   nim -o lslpp lppsource_aix432
   ```

   We find that bos.sysmgt.smit is missing.

But how can this be? If the bos.sysmgt.smit fileset is not in the lppsource directory, the lppsource resource should fail also. It is possible because the lppsource resource already exists when we run setup_server. Someone copied updates to the lppsource directory, and one of the updates was copied over bos.sysmgt.smit. Since this file is not an install fileset, we get the error.

An important lesson to learn from this error is not to always believe that all the files in the lppsource directory are being considered in the lppsource NIM object. If the lppsource resource is already created, use NIM commands to see what filesets are in the lppsource directory. If the lppsource resource is not already created, run the `inutoc` command and check the .toc file.

2.2.5.3 Resource locked

We ran setup_server on the CWS (Figure 36 shows the topology of the RS/6000 SP), and we got the error described in Figure 37.

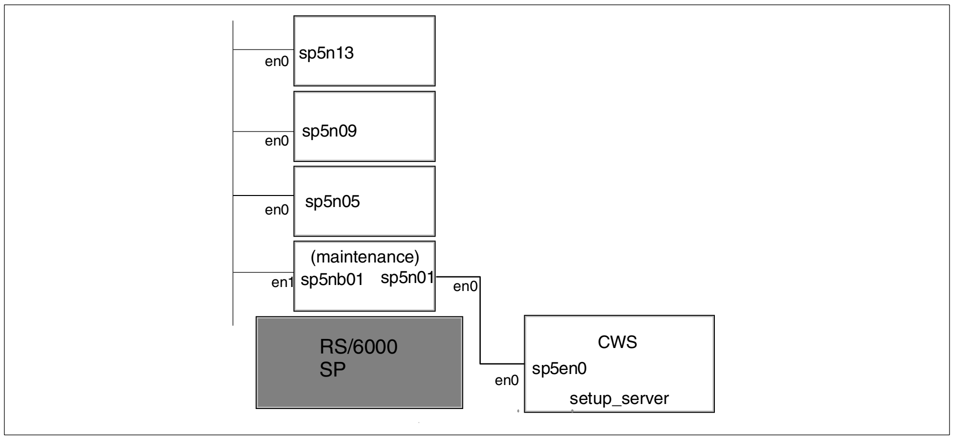


Figure 36. Case study 3—Resource locked

```
0042-001 nim: processing error encountered on "master":
   0042-001 m_alloc_boot: processing error encountered on "master":
   0042-157 c_alloc_boot: unable to access the
"/tftpboot/spot_aix432.rs6k.mp.ent" file


allnimres: 0016-261: Failure of bosinst operation (install) from server
node 1 (sp5n01.msc.itso.ibm.com) to client node 9
(sp5n09.msc.itso.ibm.com)
(nim -o bosinst; rc=1).
```

Figure 37. setup_server error

The message is clear: The boot image file does not exist (someone probably removed it).

How should we solve this problem? We simply have to re-create the boot images from the SPOT by issuing:

```
nim -o check -a debug=no spot_aix432
```

But this time someone kills this process. So, how can we solve this problem now?

By running:

```
lsnim -l spot_aix432
```

We get the output shown in Figure 38.

```
spot_aix432:
   class         = resources
   type          = spot
   locked        = 20492
   Rstate        = verification is being performed
   prev_state    = ready for use
   location      = /spdata/sys1/install/aix432/spot/spot_aix432/usr
   version       = 4
   release       = 3
   mod           = 2
   alloc_count   = 0
   server        = master
   if_supported  = rs6k.mp ent
   if_supported  = rs6k.up ent
   if_supported  = rs6k.up tok
   Rstate_result = failure
   plat_defined  = chrp
   plat_defined  = rs6k
   plat_defined  = rspc
```

Figure 38. lsnim output—Resource locked

The resource is locked by process ID `20492`. But we do not see any process running that has this ID, so we stop the NIM daemon and start it again to remove the lock by executing:

```
stopsrc -s nimesis
startsrc -s nimesis
```

If we now look at the output of the same NIM command, we get the output shown in Figure 39 on page 72.

```
spot_aix432:
   class         = resources
   type          = spot
   Rstate        = verification is being performed
   prev_state    = ready for use
   location      = /spdata/sys1/install/aix432/spot/spot_aix432/usr
   version       = 4
   release       = 3
   mod           = 2
   alloc_count   = 0
   server        = master
   if_supported  = rs6k.mp ent
   if_supported  = rs6k.up ent
   if_supported  = rs6k.up tok
   Rstate_result = failure
   plat_defined  = chrp
   plat_defined  = rs6k
   plat_defined  = rspc
```

Figure 39. lsnim output—Resource released

Now, we do not have the lock anymore, but the Rstate attribute is not `ready for use`. In this case, we can use the `m_chattr` command. Invoke:

```
/usr/lpp/bos.sysmgt/nim/methods/m_chattr -a Rstate=available
spot_aix432
```

Next, execute the following command to confirm that the SPOT does not have any problems:

```
nim -o check spot
```

Then issue `lsnim -l spot_aix432` again to confirm that everything is correct.

2.2.5.4 mksysb resource failure on the boot/install server

We configured one node to install, and after running setup_server on the boot/install server (Figure 40 on page 73 shows the topology of the RS/6000 SP), we noticed that the image was not the one we wanted to install. So, we copied the image that we wanted to the CWS and ran setup_server. We got the output shown in Figure 41 on page 73.

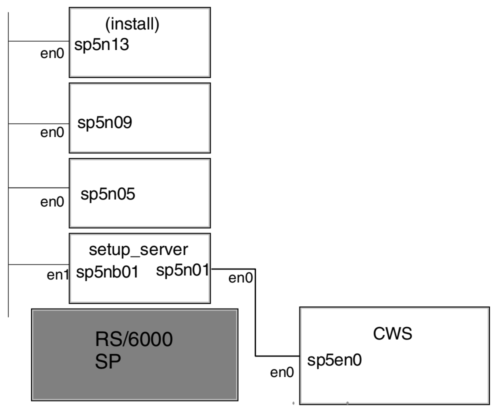


Figure 40. Case study 4—mksysb resource failure

```
mknimres: Copying /usr/lpp/ssp/install/bin/pssp_script to
/spdata/sys1/install/pssp/pssp_script.
mknimres: Copying /usr/lpp/ssp/install/config/bosinst_data.template to
/spdata/sys1/install/pssp/bosinst_data.
mknimres: Copying
/usr/lpp/ssp/install/config/bosinst_data_prompt.template to
/spdata/sys1/install/pssp/bosinst_data_prompt.
mknimres: Copying
/usr/lpp/ssp/install/config/bosinst_data_migrate.template to
/spdata/sys1/install/pssp/bosinst_data_migrate.
mknimres: 0016-467 The size of image bos.obj.ssp.432 on the cws and on
node sp5n01.msc.itso.ibm.com, differs.  Unable to proceed.
```

Figure 41. setup_server output

Let us look at the resource state on the boot/install server by issuing:

```
lsnim -a Rstate
```

The output of this command is shown in Figure 42 on page 74.

```
boot:
   Rstate = ready for use
nim_script:
   Rstate = ready for use
psspscript:
   Rstate = ready for use
prompt:
   Rstate = ready for use
noprompt:
   Rstate = ready for use
migrate:
   Rstate = ready for use
13_noprompt:
   Rstate = ready for use
13_migrate:
   Rstate = ready for use
lppsource_aix432:
   Rstate = ready for use
mksysb_1:
   Rstate = ready for use
spot_aix432:
   Rstate = ready for use
```

Figure 42. Rstate attribute

The Rstate attributes of all resource objects are `ready for use`, so there is no problem with resource objects. Then why are we getting this error?

Let us look at the output of the following command

```
lsnim -l mksysb_1
```

The output is shown in Figure 43.

```
mksysb_1:
   class       = resources
   type        = mksysb
   Rstate      = ready for use
   prev_state  = unavailable for use
   location    = /spdata/sys1/install/images/bos.obj.ssp.432
   version     = 4
   release     = 3
   mod         = 2
   alloc_count = 1
   server      = master
```

Figure 43. mksysb resource output

The allocation count of this resource is `1`, which means that a machine object has this resource allocated.

The problem is that every time a boot/install server that is not the CWS has an mksysb image of different size than the one from the CWS with the same file name, it tries to remove the mksysb resource object, remove the file, get the new one from the CWS, and create the resource object.

In this case, the system cannot remove the resource, since it is allocated. To solve this problem, we have to deallocate the resource from the machine object to see which machine objects are allocating this resource, we execute

```
lsnim -a mksysb
```

The output is shown in Figure 44.

```
sp5n09:
   mksysb = mksysb_1
```

Figure 44. lsnim output—Resource allocated

sp5n09 is the machine object that has the resource allocated. We execute the following commands to deallocate it:

```
nim -Fo reset sp5n09
nim -o deallocate -a mksysb=mksysb_1 sp5n09
```

The problem is solved by executing setup_server again.

The lesson of this example is do not copy mksysb image files to image files that already exist. There are other problems that can arise from doing this. If you put another image on the CWS with the same name, the resource is not re-created, so the information in the resource object about version, release, and maintenance level can be incorrect.

The correct procedure is to copy the mksysb image with another name and introduce the correct information in the Volume_Group SDR class. Another approach is to remove the mksysb resource object.

2.2.5.5 Problem allocating resources

We ran setup_server on the boot/install server (Figure 45 on page 76 shows the topology of the RS/6000 SP), and we got the error shown in Figure 44 on page 75.

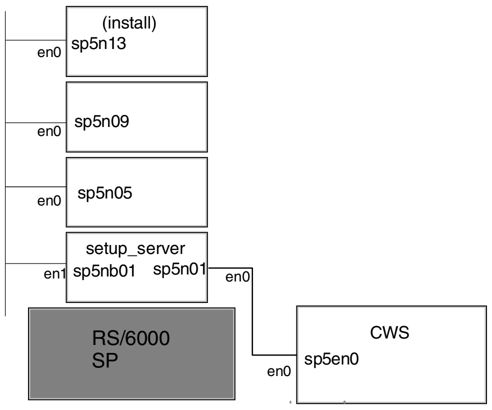


Figure 45. Case study 5—Problem allocating resources

```
0042-001 nim: processing error encountered on "master":
   0042-027 m_bos_inst: "simages" is required to complete the
        definition of the "lppsource_aix432" object.  This attribute is
either
         currently missing or the current operation would cause
         it to become missing.

allnimres: 0016-261: Failure of bosinst operation (install) from server
node 1 (sp5n01.msc.itso.ibm.com) to client node 9
(sp5n09.msc.itso.ibm.com)
(nim -o bosinst; rc=1).
```

Figure 46. setup_server error allocating lppsource

Let us look at the first error message:

```
042-001 nim: processing error encountered on "master":
0042-027 m_bos_inst: "simages" is required to complete the
definition of the "lppsource_aix432" object. This attribute is either
currently missing or the current operation would cause
it to become missing.
```

The problem has to be with the lppsource resource object. Therefore, we issue:

```
lsnim -l lppsource_aix432
```

We get the output shown in Figure 47 on page 77.

```
lppsource_aix432:
   class       = resources
   type        = lpp_source
   Rstate      = ready for use
   prev_state  = ready for use
   location    = /spdata/sys1/install/aix432/lppsource
   alloc_count = 1
   server      = sp5en0
```

Figure 47. lppsource resource output

From this output , we confirm that the simages attribute is missing from the resource object.

We get the fileset that is described in the REQUIRED_SIMAGES variable defined in the /usr/lpp/bos.sysmgt/nim/methods/c_sh_lib file as shown in Figure 48.

```
REQUIRED_SIMAGES="\
        bos \
        bos.64bit \
        bos.up \
        bos.mp \
        bos.net \
        bos.diag \
        bos.sysmgt \
        bos.terminfo \
        bos.terminfo.all.data \
        devices.base.all \
        devices.buc.all \
        devices.common.all \
        devices.graphics.all \
        devices.mca.all \
        devices.rs6ksmp.base \
        devices.scsi.all \
        devices.sio.all \
        devices.sys.all \
        devices.tty.all \
        xlC.rte"
```

Figure 48. Required lppsource images

We compare this with the filesets obtained from the execution of the following command:

```
nim -o lslpp lppsource_aix432
```

We notice that the bos.sysmgt.smit fileset is missing from the lppsource (someone deleted it).

So now, we copy the fileset back from the AIX CDROM and perform the following command:

```
nim -o check lppsource_aix432
```

If we now execute:

```
lsnim -l lppsource_aix432
```

We get the output shown in Figure 39 on page 72, which indicates that we have the simages attribute.

```
lppsource_aix432:
   class       = resources
   type        = lpp_source
   Rstate      = ready for use
   prev_state  = verification is being performed
   location    = /spdata/sys1/install/aix432/lppsource
   simages     = yes
   alloc_count = 1
   server      = sp5en0
```

Figure 49. lsnim output—simages attribute

Because we added filesets to lppsource, we should also apply these filesets to the SPOT by executing:

```
nim -o cust -a lpp_source=lppsource_aix432 -a installp_flags=acXg -a
filesets=ALL spot_aix432
```

2.2.5.6 Problem allocating the SPOT

We ran setup_server on the boot/install server (Figure 50 on page 79 shows the topology of the RS/6000 SP), and we got the error shown in Figure 51 on page 79.

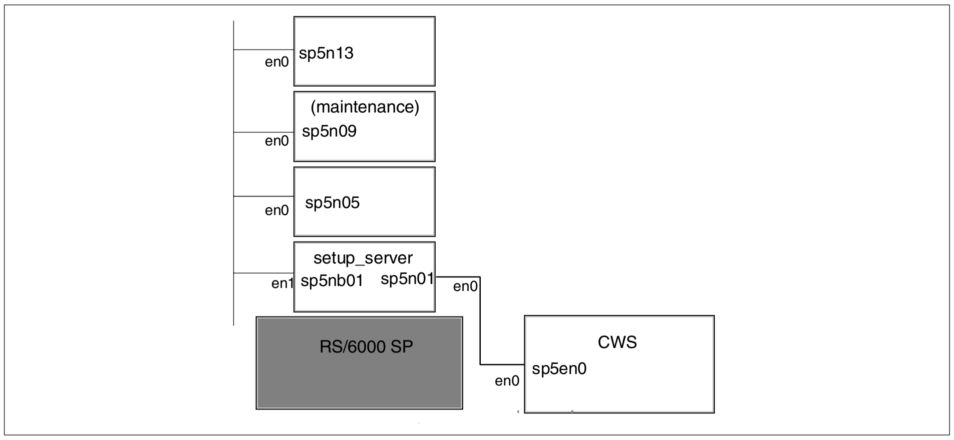


Figure 50. Case study 6—Problem allocating the SPOT

```
0042-001 nim: processing error encountered on "master":
   0042-058 m_alloc_spot: unable to allocate "spot_aix432" to "sp5n09"
        because it does not support the network interface type
        of that client

allnimres: 0016-251: Failure to allocate spot resource spot_aix432 from
server node 1
(sp5n01.msc.itso.ibm.com) to node 9 (sp5n09.msc.itso.ibm.com) (nim -o
allocate; rc=1).
```

Figure 51. setup_server error allocating the SPOT

The message is clear: The SPOT does not have support for the network interface of the machine object. We execute the following command:

```
lsnim -l spot_aix432
```

We get the output shown in Figure 52 on page 80.

```
spot_aix432:
   class         = resources
   type          = spot
   Rstate        = ready for use
   prev_state    = verification is being performed
   location      = /spdata/sys1/install/aix432/spot/spot_aix432/usr
   version       = 4
   release       = 3
   mod           = 2
   alloc_count   = 0
   server        = master
   if_supported  = rs6k.up tok
   Rstate_result = success
   plat_defined  = chrp
   plat_defined  = rs6k
   plat_defined  = rspc
```

Figure 52. lsnim output—Platforms supported

The output shows the platforms that are supported on this SPOT. But the only interface supported is token-ring for the rs6k uniprocessor platform.

So now, we must get all the device filesets from the AIX CD-ROM. If we have additional support devices, we put them in the lppsource directory also and execute the following commands:

```
nim -o check lppsource_aix432
nim -o cust -a lpp_source=lppsource_aix432 -a installp_flags=acXg -a
filesets=ALL spot_aix432
```

Now, if we run the command:

```
lsnim -l spot_aix432
```

We get the output shown in Figure 53 on page 81.

```
spot_aix432:
   class         = resources
   type          = spot
   Rstate        = ready for use
   prev_state    = verification is being performed
   location      = /spdata/sys1/install/aix432/spot/spot_aix432/usr
   version       = 4
   release       = 3
   mod           = 2
   alloc_count   = 0
   server        = master
   if_supported  = rs6k.mp ent
   if_supported  = rs6k.up ent
   if_supported  = rs6k.up tok
   Rstate_result = success
   plat_defined  = chrp
   plat_defined  = rs6k
   plat_defined  = rspc
```

Figure 53. lsnim output—High node support

Now, we have support for High nodes as we can see from the if_supported attribute in Figure 53.

In this case, because the boot/install server does not have any chrp machine object as client, the boot images are not created. The boot/install servers that have chrp machine objects defined in the NIM master must have the following:

```
         if_supported = chrp.mp ent
```

2.3 Node installation

The topics in this section are:

- The basic checklist
- An overview of the network installation process
- A methodology for finding network installation problems
- Problem determination tools
- Case studies

2.3.1 The basic checklist

If you have a problem when a node boots from the network, you should do the following:

- Verify the cable connections.
- Verify the nodecond log (/var/adm/SPlogs/spmon/nc directory), and see if there are errors.
- Try manual node conditioning.
- Investigate if there is a Kerberos or hardmon ACL problem.
- Investigate if there is an SDR problem.

2.3.2 Node installation process

When you want to perform an installation or migration of a node, use the **Network Boot** option from the Hardware Perspectives, which invokes the following command:

```
nodecond <frame#> <node#>
```

where:

`<frame#>` is the frame number selected

`<node#>` is the node number selected

The `nodecond` command performs the following functions:

1. To retrieve the node type value, the command `spmon -G frame<x>/node<y>/type/value`

   is issued, where:

   `<x>` is the frame number

   `<y>` is the node number

2. If the node type retrieved has one of the following values:

   177 (332 MHz and POWER3 Thin nodes), 178 (332 MHz and POWER3 Wide nodes), 179 (POWER3 High node), or 10 (S70 node)

The platform type is chrp, and the nodecond_chrp expect script is invoked.

If the value is one of the following:

33, 65, 97, 81, 83, 113, 115, or 161

The platform is rs6k and the nodecond_mca expect script is invoked.

The nodecond_mca and nodecond_chrp expect scripts perform the same actions as a "manual node conditioning". This is accomplished by capturing messages on the node console and sending keystrokes back.

The `nodecond_mca` flow and logic is as follows:

1. The bootp_response attribute is retrieved from the Node SDR class.
2. The enet_type attribute for the en0 interface is retrieved from the Adapter SDR class. Based on this information, the boot interface type is selected.
3. Perform the same actions as in manual node conditioning. For information on how to perform a manual node conditioning, see 2.3.10, “Problem determination tool” on page 143.
4. After the message `STARTING SYSTEM (BOOT)` is sent to the node’s console, `nodecond_mca` terminates.

The `nodecond_chrp` flow and logic is as follows:

1. The bootp_response attribute is retrieved from the Node SDR class.
2. The enet_type, enet_rate and duplex attributes for the en0 interface are retrieved from the Adapter SDR class. Based on this information, the boot interface type is selected.
3. Perform the same actions as in manual node conditioning. For information on how to perform a manual node conditioning, see 2.3.10, “Problem determination tool” on page 143.
4. After the message `Welcome to AIX` is sent to the node’s console, `nodecond_chrp` terminates.

When `nodecond` exits, the node is in the process of booting through the network, that is, the node is sending a bootp request.

The network boot process can be described as follows:

1. LED 231 is shown on the node.
2. The node sends a bootp broadcast packet through the network.
3. The machines in the same network that have an entry in the /etc/inetd.conf file, as shown in Figure 54, are listening to bootp packets.

```
bootps dgram udp wait root /usr/sbin/bootpd bootpd /etc/bootptab
```

Figure 54. Sample inetd.conf file entry

These machines retrieve the packet and verify that the hardware address of the sending node is the same as the hardware address specified in one of the entries in the /etc/bootptab file.

If it is, they send back a bootp reply containing the information retrieved from the /etc/bootptab entry. A sample /etc/bootptab entry is presented in Figure 55.

```
sp5n13.msc.itso.ibm.com:bf=/tftpboot/sp5n13.msc.itso.ibm.com:ip=192.168
.6.13:ht=Ethernet:ha=02608CE8FCB3:sa=192.168.6.1:sm=255.255.255.0:
```

Figure 55. Sample bootptab entry

The following information is sent back in the bootp reply packet is the following:

- The hostname of the client node
- The IP address of the client node
- The subnet mask of the network
- The IP address of the server that contains the boot image
- The boot image file name that the node must request

4. If the client node does not receive a reply packet within a time-out value, another bootp request is sent across the network. If the limit is reached, the Main Menu of the network boot process is displayed on the console, and LED 260 is shown on the node.
5. If the client node receives the bootp reply packet, the information received is stored in the IPL Control Block, which is stored in NVRAM.
6. The node attempts to retrieve the boot image file from the server via tftp. In the tftp request, the node uses the server IP address and the image file name that were received in the bootp reply packet.
7. The node performs a verification to confirm that the file received is a valid boot image file. On MCA nodes, to accomplish this, the four initial bytes of the file are read. If a magic word (IBMA in the EBCDIC character set) is found, the file is considered to be a valid boot image file.

If the file received is not a valid image file, the Main Menu panel of the network boot process is displayed and LED 260 is shown on the node.

If the file received is a valid boot image, LED 299 is shown on the node.

The network boot process continues by reading the boot record from the boot image. The boot record has pointers to the boot code, boot image file system, and a subset of the ODM database inside the boot image. This structure is shown in Figure 56.

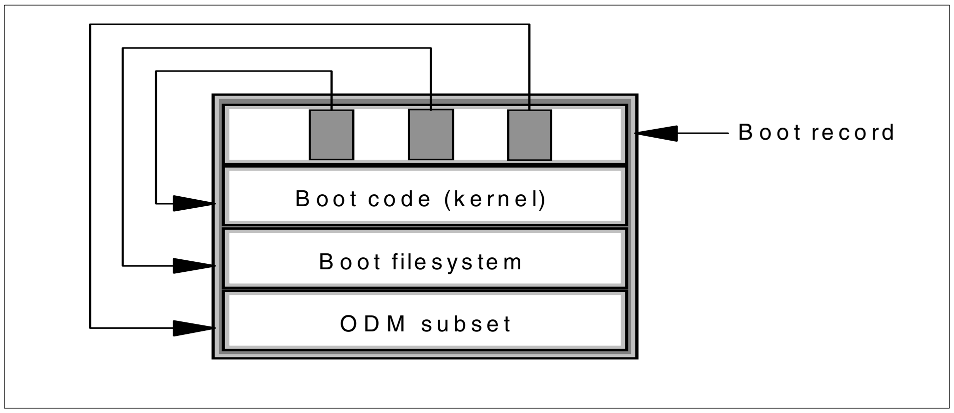


Figure 56. Boot record

The boot image file system is based on the prototype file that was specified in the creation of the boot image. An extract of the /usr/lib/boot/network/rs6k.ent.proto file is shown in Figure 57.

```
etc     d--- 755 0 0
        drivers l--- 777 0 0 /usr/lib/drivers
        init    ---- 555 0 0 /usr/lib/boot/ssh
        methods l--- 777 0 0 /usr/lib/methods
        microcode l--- 777 0 0 /usr/lib/microcode
        niminfo l--- 755 0 0 /SPOT/niminfo
        objrepos d--- 777 0 0
                Config_Rules    ---- 777 0 0
/tmp/boot_ODM/Config_Rules
                CuDep   ---- 777 0 0 /tmp/boot_ODM/CuDep
                CuDv    ---- 777 0 0 /tmp/boot_ODM/CuDv
```

Figure 57. Sample proto file

At this stage, the RAM file system is created with the contents of the boot image file system and control is passed to the boot code.

The boot code invokes /etc/init, which, in fact, is /usr/lib/boot/ssh.

/etc/init invokes /sbin/rc.boot with the value 1 as argument.

The rc.boot flow and logic are as follows:

1. If the file bootinfo_<platform> is not executable, the boot process is stopped with LED C10.
2. Unnecessary files from the RAM file system are removed.
3. The IPL Control Block is read. If it cannot be determined what type of boot is being performed, the boot process is stopped with LED C06.
4. The RAM file system is increased.
5. LED 600 is shown.
6. The configuration manager command `cfgmgr -fv` is executed.
7. The IP resolution is set to the local /etc/hosts file.
8. The IPL control block is read. This block has the following information:
    - The node IP address
    - The boot server IP address
    - The boot image file name
9. LED 606 is shown.
10. The loopback interface (lo0) is configured.
11. The boot interface (en0) is configured. If there is an error in the interface configuration, the boot process is stopped with LED 607 or LED 605.
12. LED 608 is shown.
13. The niminfo file is retrieved from the server via tftp. The file name that is requested is the /tftpboot/<reliable_hostname>.info file. It is placed on the node as /SPOT/niminfo.
14. If there is an error retrieving the file, retry the operation until there is no error. LED 608 shows on the LCD or LED display.
15. If the file retrieved is not empty, load the environment variables defined there. Otherwise, stop the boot process with LED 609.

Figure 58 on page 87 contains a sample of the niminfo file.

```
export NIM_NAME=sp5n13
export NIM_HOSTNAME=sp5n13.msc.itso.ibm.com
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=sp5nb01.msc.itso.ibm.com
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export RC_CONFIG=rc.bos_inst
export NIM_BOSINST_ENV="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_b
osinst_env"
export NIM_BOSINST_RECOVER="/../SPOT/usr/lpp/bos.sysmgt/nim/methods
/c_bosinst_env -a hostname=sp5n13.msc.itso.ibm.com"
export
SPOT=sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/aix432/spot/spot_
aix432/usr
export NIM_BOSINST_DATA=/NIM_BOSINST_DATA
export NIM_CUSTOM="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script
-a location=sp5nb01.msc.itso.ibm.com:/export/nim/scripts/sp5n13.scr
ipt"
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/bos
export NIM_BOS_FORMAT=rte
export NIM_HOSTS=" 192.168.6.13:sp5n13.msc.itso.ibm.com
192.168.6.1:sp5nb01.msc.itso.ibm.com  192.168.5.150:sp5en0.msc.itso
.ibm.com"
export NIM_MOUNTS=" sp5en0.msc.itso.ibm.com:/spdata/sys1/install/ai
x432/lppsource:/SPOT/usr/sys/inst.images:dir  sp5nb01.msc.itso.ibm.
com:/spdata/sys1/install/pssp/bosinst_data_prompt:/NIM_BOSINST_DATA:
file"
export ROUTES=" 192.168.5.0:255.255.255.0:192.168.6.1 "
```

Figure 58. Sample niminfo file

The variables used in this file are defined as follows:

NIM_NAME: Is the short reliable hostname for this node. It is the NIM machine name for this node.

NIM_HOSTNAME: Defines the reliable hostname for this node.

NIM_CONFIGURATION: Stand-alone is the only supported configuration on SP nodes. Diskless or dataless configurations are not supported.

NIM_MASTER_HOSTNAME: Defines the reliable hostname of the boot/install server. It is the NIM master hostname.

NIM_MASTER_PORT: Defines the IP port number that is used to communicate with the NIM master.

NIM_REGISTRATION_PORT: This is the IP port number used for registration.

RC_CONFIG: Defines the script that is invoked by rc.boot during the boot phase. If you are performing an installation or migration, the file defined in this variable should be the rc.bos_inst file.

NIM_BOSINST_ENV: Defines the script that is used to set the NIM environment.

NIM_BOSINST_RECOVER: Defines the script that is used to recover the NIM environment during a network install.

SPOT: Defines the SPOT that will be used during installation or migration. The format is <spot_server_hostname>:<path_of_spot_on_server>.

NIM_BOSINST_DATA: Defines where the NIM_BOSINST_DATA resource will be placed on the node.

NIM_CUSTOM: Defines the script to run in the customization phase. The defined script configures the node as a NIM client, NFS-mounts the /export/nim/scripts/<host>.script script on the node, and invokes it. The <host>.script script NFS-mounts the pssp_script script file on the node and runs it there.

NIM_BOS_IMAGE: Defines the location of the bos file on the node if this is a migration or installation. The directory defined in the variable should be /SPOT/usr/sys/inst.images, which is where the lppsource directory is NFS-mounted. Defines the location of the mksysb image file on the node if this is an overwrite installation. The file defined in the variable should be /NIM_BOS_IMAGE, which is where the mksysb image file is NFS-mounted.

NIM_BOS_FORMAT: Could be rte or mksysb. The rte value is defined in the case of a migrate installation. The mksysb value is defined in the case of an overwrite installation.

NIM_HOSTS: Defines the entries that will be defined in the /etc/hosts file on the node.

NIM_MOUNTS: Defines the NFS-mounts that will be done by the node. The format is <server_hostname>:<server_location>:<local_location>. If you are performing a migration installation, you will only have two triplets (lppsource and bos_inst_data resources). If you are performing an overwrite installation, you will see three triplets (mksysb, lppsource, and bos_inst_data resources).

ROUTES: Defines the IP route to access the lppsource resource. This variable is only defined when the boot/install server of the node is not the CWS and the node is not on the same subnet as the CWS.

16. The /etc/host file is created based on the NIM_HOST environment variable.
17. The IP route defined in the ROUTES environment variable is configured. If there is an error while adding this IP route, the boot process is stopped with LED 613.
18. LED 610 is shown.
19. The info attribute of the NIM machine object is changed to LED610: mount $SPOT /SPOT/usr.
20. The NFS-mount of the SPOT file system is performed. If the NFS-mount was unsuccessful, the following actions are performed:
    - The info attribute of the NIM machine object is changed to LED611: failure: mount $SPOT /SPOT/usr.
    - The boot process is stopped with LED 611.
21. LED 612 is shown.
22. The command that is defined in the environment variable RC_CONFIG is invoked. For an overwrite or migrate installation, the rc.bos_inst script is executed.

The rc.bos_inst flow and logic are as follows:

1. The Mstate attribute of the NIM machine object is changed to In the process of booting.
2. LED 610 is shown.
3. For each file/directory specified in the environment variable NIM_MOUNTS, the following steps are performed:
    - If there is an error while creating the directory for the local mount point, the boot process is stopped with LED 625.
    - The info attribute of the NIM machine object is changed to LED610: mount $1 $2 $3.
    - An attempt to NFS-mount the file/directory is made. If there is an error, the following steps are performed:
        - The info attribute of the NIM machine object is changed to LED611: failure: mount $1 $2 $3.
        - The boot process is stopped with LED 611.
4. The info attribute of the NIM machine object is cleared.
5. LED 622 is shown.

6. The configuration methods that are needed for the `cfgmgr` command are linked from the SPOT.
7. The first phase configuration manager, `cfgmgr -v -fis`, is executed. This is the second time `cfgmgr` is executed for the first phase, because the system now has additional configuration methods.
8. LED 622 is shown.
9. The full ODM database and the configuration methods for the console are linked from the SPOT.
10. The second phase configuration manager, `cfgmgr -v -fis`, is executed.
11. LED 622 is shown.

The execution of rc.boot is now finished.

The /etc/init script invokes the /sbin/rc.boot script with value 2 as argument. Its flow and logic are as follows:

1. The IP resolution is set to local (/etc/hosts file).
2. The environment variables from the niminfo file (/SPOT/niminfo) are reloaded.
3. The rc.bos_inst script is executed again, and the following actions are performed:
    1. The rc.boot file is deleted, so there is no way to go back.
    2. The IP parameters tcp_keepintvl and tcp_keepidle are defined.
    3. The ODM objects that are needed for pre-test diagnostics on rootvg disks are copied to the RAM file system.
    4. The info attribute of the NIM machine object is cleared.
    5. The bi_main script is invoked.

2.3.2.1 The bi_main script

In this section, we describe the functionality of the bi_main script .

First, we should analyze two important files used in the installation process: the bosinst.data file and the image.data file. Figure 59 on page 91 shows a sample of the bosinst.data file.

```
    control_flow:
        CONSOLE = /dev/tty0
        INSTALL_METHOD = overwrite
        PROMPT = no
        EXISTING_SYSTEM_OVERWRITE = yes
        INSTALL_X_IF_ADAPTER = no
        RUN_STARTUP = no
        RM_INST_ROOTS = no
        ERROR_EXIT =
        CUSTOMIZATION_FILE =
        TCB = no
        INSTALL_TYPE = full
        BUNDLES =

    target_disk_data:
        LOCATION =
        SIZE_MB =
        HDISKNAME = hdisk0

    locale:
        BOSINST_LANG = en_US
        CULTURAL_CONVENTION = en_US
        MESSAGES = en_US
        KEYBOARD = en_US
```

Figure 59. Sample bosinst.data file

There are important fields in bosinst.data that deserve some consideration:

CONSOLE: This attribute defines the console device that is used during the installation process. When prompt is set to yes, the CONSOLE attribute must be a valid tty.

INSTALLATION_METHOD: This attribute defines the installation method, which can be:

- overwrite
- migrate
- preserve

On the RS/6000 SP, only overwrite and migrate are supported.

PROMPT: This attribute defines the mode of installation, which can be set to yes (prompt mode) or no (non-prompt mode).

If it is set to yes, the Main Menu of the installation process is displayed on the node's console. Then we have to open a tty (s1term -w) to be able to proceed with the installation.

Note

If the bi_main script detects inconsistencies in bosinst.data or image.data, the prompt mode is changed to yes. This scenario is covered in the problem determination for LED C48 (see 2.3.11.4, “LED C48” on page 164). So, even if the prompt is set to no, the Main Menu of the installation process can still appear.

EXISTING_SYSTEM_OVERWRITE: This attribute defines whether the disks selected that contain data can be overwritten. This attribute can be changed by the bi_main script, but in that case, the prompt attribute will also be changed to yes, and the Main Menu of the installation process is displayed on the node’s console.

ERROR_EXIT: This attribute defines what additional process should be invoked when an error is detected. The process defined in this attribute will be invoked whenever the BI_Error function is invoked.

CUSTOMIZATION_FILE: This attribute defines which process should be invoked after the post-installation function of the bi_main script. This has nothing to do with the pssp_script script, which is invoked inside the post-installation function.

One target_disk_data_stanza must be present for each disk that is defined in the selected volume group for the node.

LOCATION: This attribute defines the physical location of the disk that will be part of the rootvg volume group.

HDISKNAME: This attribute defines the name of the disk that will be part of the rootvg volume group. Be aware that hdisk numbering could be different when booting from a network. Use LOCATION instead of HDISKNAME when entering data to the SDR via the `spmkvgobj` or `spchvgobj` commands.

Suppose that you defined the hdisk0 as the target installation disk, and the node has one internal disk (hdisk0) and 10 external SSA disks that contain other volumes groups. Someone decides to reinstall this node, but during the installation process, the internal disk was not detected, so one of the external SSA disks becomes hdisk0. Since EXISTING_SYSTEM_OVERWRITE is true (default value), the install process continues without prompting you for anything, and the operation system is installed on the hdisk0 disk. After installation has finished, you discover that the operating system has not been reinstalled on the internal disk and access to one of the external volume groups is lost.

Note

Use LOCATION instead of HDISKNAME whenever possible.

Note

Do not change any of these attributes manually unless you understand all the implications.

Let us now describe how bosinst.data is obtained:

- First, populate the SDR:
  - Define the disks that are to be part of the rootvg volume group.
  - Define the volume group and the bootp_response for the node.
- When running setup_server on the boot/install server, the mknimres wrapper is invoked. This wrapper creates the following resources:
  - If the bootp_response is migrate, and the disk defined in the selected Volume_Group in the SDR class for the node is hdisk0 alone, perform the following actions:
    - If the migrate resource object of bosinst_data type does not exist, a template file is copied to /spdata/sys1/install/pssp/bosinst_data_migrate, and the migrate resource object is created.
    - If the resource object already exists, do nothing.
  - If the bootp_response is migrate, and the disk defined in the selected Volume_Group in the SDR class for the node is not hdisk0 alone, perform the following actions:
    - Create the /spdata/sys1/install/pssp/<node_number>.migrate file. This file is identical to the bosinst_data_migrate file, except on the target_disk_data stanzas. One entry is added for each disk/location that is defined in the selected Volume_Group SDR class.
  - If the bootp_response is install, and the disk defined in the selected Volume_Group in the SDR class for the node is hdisk0 alone, perform the following actions:

    - If the migrate resource object of the bosinst_data type does not exist, a template file is copied to /spdata/sys1/install/pssp/bosinst_data, and the NIM noprompt resource is created.
    - If the resource object already exists, do nothing.
  - If the bootp_response is install, and the disk defined in the selected Volume_Group in the SDR class for the node is not hdisk0 alone, perform the following actions:
    - Create the /spdata/sys1/install/pssp/<node_number>.noprompt file. This file is identical to the bosinst_data file, except on the target_disk_data stanzas. One entry is added for each disk/location that is defined in the selected Volume_Group.
- When running setup_server on the boot/install server, another wrapper is invoked: the allnimres wrapper, which performs the following steps (in respect to bosinst_data resources):
  - One of the following resources is allocated to the node depending on the bootp_response and the disks defined on the selected Volume_Group SDR class:
    - migrate
    - noprompt
    - <node_number>_migrate
    - <node_number>_noprompt
  - The allocation process NFS-exports the file that is associated with the resource.
  - The bos_inst NIM operation is performed on the machine object. Since the resource object of type bosinst_data is allocated to the node, the bos_inst operation creates /tftpboot/<realible_hostname>.info with the following entries:
    - NIM_BOSINST_DATA=/NIM_BOSINST_DATA
    - NIM_MOUNTS="...sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/pssp/<resource_name>:/NIM_BOSINST_DATA:file..."
- When a network boot is performed on the node, the rc.boot script is invoked. Within the rc.boot process, the file /spdata/sys1/install/pssp/<resource_name> is NFS-mounted on /NIM_BOSINST_DATA.
- When the bi_main script invokes the internal function extract_data_files, the following actions are performed:

- When performing an overwrite install, the bosinst.data is retrieved from the mksysb image.
- When performing a migrate installation, the bosinst.data file is extracted from the bos fileset in the lppsource directory.
- And, finally, if the /NIM_BOSINST_DATA file exists, it is copied to bosinst.data.

The image.data file is the other import file used in the installation process. The file is extracted from the mksysb image file when performing an overwrite installation and is extracted from the bos fileset in the lppsource directory when performing a migrate installation.

A sample image.data file is shown in Figure 60., “Sample image.data file” on page 96.

```
image_data:
         IMAGE_TYPE= bff
         DATE_TIME= Mon Aug 24 16:08:10 EDT 1998
         UNAME_INFO= AIX k17n03 3 4 000085628900
         PRODUCT_TAPE= no
         USERVG_LIST=
         OSLEVEL= 4.3.2.0

logical_volume_policy:
         SHRINK= no
         EXACT_FIT= no

ils_data:
         LANG=  en_US

##Command used for vg_data; /usr/sbin/lsvg

vg_data:
         VGNAME= rootvg
         PPSIZE= 8
         VARYON= yes
         VG_SOURCE_DISK_LIST= hdisk0
         QUORUM= 2
         CONC_CAPABLE= no
         CONC_AUTO= no

##Command used for source_disk_data; /usr/sbin/bootinfo

source_disk_data:
         PVID= 00008562000143f3
         CONNECTION= vscsi0//0,0
         LOCATION= 00-00-00-0,0
         SIZE_MB= 4303
         HDISKNAME= hdisk0

##Command used for lv_data; /usr/sbin/lslv

lv_data:
         VOLUME_GROUP= rootvg
         LV_SOURCE_DISK_LIST= hdisk0
         LV_IDENTIFIER= 0000856279682419.1
         LOGICAL_VOLUME= hd5
         VG_STAT= active/complete
         TYPE= boot
         MAX_LPS= 512
         COPIES= 1
         LPs= 1
         STALE_PPs= 0
         INTER_POLICY= minimum
         INTRA_POLICY= edge
         MOUNT_POINT=
         MIRROR_WRITE_CONSISTENCY= on
         LV_SEPARATE_PV= yes
         PERMISSION= read/write
         LV_STATE= closed/syncd
         WRITE_VERIFY= off
```

Figure 60. Sample image.data file

```
                PP_SIZE= 8
                SCHED_POLICY= parallel
                PP= 1
                BB_POLICY= relocatable
                RELOCATABLE= no
                UPPER_BOUND= 32
                LABEL=
                MAPFILE=
                LV_MIN_LPS= 1
                STRIPE_WIDTH=
                STRIPE_SIZE=
....
       fs_data:
                FS_NAME= /
                FS_SIZE= 65536
                FS_MIN_SIZE= 10192
                FS_LV= /dev/hd4
                FS_FS= 4096
                FS_NBPI= 2048
                FS_COMPRESS= no
                FS_BF= false
                FS_AGSIZE= 8

       fs_data:
                FS_NAME= /usr
                FS_SIZE= 671744
                FS_MIN_SIZE= 633344
                FS_LV= /dev/hd2
                FS_FS= 4096
                FS_NBPI= 4096
                FS_COMPRESS= no
                FS_BF= false
                FS_AGSIZE= 8
....
```

Figure 61. Sample image.data file (continued)

Let us describe some of the important attributes defined in this file:

IMAGE_TYPE: Defines the format of the overwrite/migrate installation file (bff — backup file format).

PRODUCT_TAPE: Can be yes or no. If it is set to no, it means that the image.data file was extracted from an mksysb image file.

OSLEVEL: Defines the operating system level of the mksysb image or bos file.

Now, let us discuss the functions of the bi_main script, which can be divided in four functions:

1. The image.data and the bosinst.data file are extracted, and the stanzas in both files are validated. Tests are done to guarantee consistency of the attributes of these files.

2. If the installation method is defined as prompt, or if inconsistencies were found, the Main Menu of the installation process is displayed on the console. In these cases, we have to open a tty to proceed with the installation (`s1term -w <x> <y>`). After entering the data on the tty, or if the installation is no prompt, additional validations are performed, and if a fatal error is detected at this stage, the Main Menu of the install process is shown again.
3. If the installation method is overwrite, the system creates the rootvg volume group, logical volumes, and file systems, and restores the mksysb image file. If the installation mode is migrate, the rootvg volume group is imported and premigration activities are performed before restoring the bos fileset from the lppsource directory.
4. Finally, the post-installation procedures are performed. During this phase, pssp_script is invoked.

If there is an error after the beginning of step 3, the BI_error function is invoked. This function performs the following steps if the console is a valid tty:

- Invokes the ERROR_EXIT function.
- Displays an error on the console.
- Updates the info attribute of the NIM machine object.
- Prompts the user to perform maintenance activities.

If the console is not a valid tty, and if the error is a fatal error, it does the following:

- Invokes the ERROR_EXIT function.
- Updates the info attribute of the NIM machine object.
- Stops the installation process with LED 623.

How should we read the information that is provided in the bi_main description?

First, locate the LED code from the description given. Locate the next LED in the description. The bi_main script then is performing an action between these two LEDs.

You must also execute `lsnim -a info <short-reliable_hostname>` on the boot/install server of the node. Then, you can locate the info attribute that you have with the ones that are present on the bi_main description.

Now, let us describe the bi_main functionality.

Note

This is the behavior of bi_main in AIX 4.3.2. With other levels of AIX, the behavior may not be exactly the same.

1. Invoke the Initialize function to perform initialization of the variables and of the environment.
2. Invoke the `nim -R success` command. This command changes the Cstate attribute to Base Operating System installation is being performed.
3. LED C40 is shown.
4. Invoke the Get_Data_Files. This function performs the following steps:
   - The NIM info attribute for the node is changed to extract_data_files.
   - The bosinst.data, image.data, and preserve.list files are retrieved from the installation media (mksysb image file, if overwrite install; bos fileset, if migrate).
   - If the NIM bosinst_data resource object was allocated to this node, copy the file associated with this resource to bosinst.data.
   - If the NIM image_data resource object was allocated to this node, copy the file associated with this resource to image.data.
5. The NIM info attribute for the node is changed to query_disks.
6. Execute Get_RVG_Disks. This function is launched in the background, and creates a file with the description of all the disks on the system that include information such as: the VG ID, if the disk is a rootvg disk, and the disk name.
7. LED C42 is shown.
8. The NIM info attribute for the node is changed to: "extract_diskette_data".
9. Invoke the Extract_Diskette_Data function. This function performs the following actions:
   - If there is a file called ./signature on the diskette drive that has the magic word "data" inside, restore all the files from the diskette.
   - If the diskette is a DOS diskette, and there is a preload file inside, execute it.

Note

This diskette can be useful in solving external node installation problems.

10. If the bosinst.data does not exist, invoke the Create_Bosinst_Data functions to create it.

11. Invoke the Image_Data_Exists function. This function verifies the existence of the image.data file and launches the datadaemon daemon in the background (this daemon handles the attributes of bosinst.data and image.data files).

12. Invoke the Set_Console function. This function performs the following steps:

    - The NIM info attribute for the node is changed to setting_console.
    - The console is set according to the CONSOLE attribute in the bosinst.data file. If there is an error configuring the console, the installation process is stopped with LED C45.

13. The NIM info attribute for the node is changed to: "initialization".

14. Invokes the Init_Target_Disks function. This function performs the following steps:

    1. LED C44 is shown on the node.
    2. Waits for the Get_RVG_Disks that was launched in the background to finish.
    3. If there are no available disks on the system, flags an error.
    4. LED C46 is shown on the node.
    5. The target_disk_stanzas (disk information) are validated, taking into account the installation method that is being performed, the level of AIX on the installation media, and the type of install media (mksysb image file or bos file from lppsource).
    6. If there are no target_disk_data stanzas on the bosinst.data file (or the entries are invalid), initialize the target_disk_data stanzas to contain the previous volume group when performing a migration installation, or the first disk found by the `lsdev` command when performing an overwrite install. In both cases, set an error flag so that Main Menu of the installation process is displayed on the console.

15. The NIM info attribute for the node is changed to verifying_data_files.

16. Invoke the Verify_Image_Parameters function. This function verifies the stanzas from the image.data file. It also verifies that there is enough disk space to hold the logical volumes and file systems, taking into account the shrink option. If inconsistencies are found, an error flag is set, so that the Main Menu of the installation process is displayed on the console.

17. Invoke the Fill_Target_Stanzas function, which ensures that the fields in the target_disk_data stanza are filled.

18. Invoke the Check_Other_Stanzas function, which ensures that the non-disk-related stanzas in the bosinst.data file are valid.

19. Invoke the Get_User_Input function. This function performs the following steps:

    - If an error flag was set, it changes the installation mode to prompt.
    - If the installation mode is prompt, it performs the following actions:
        - LED C48 is shown on the node.
        - The NIM info attribute for the node is changed to "prompting_for_data_at_console.
        - Executes the BOSMenus process. (It is here that the Main Menu of the installation process is displayed.) If the return code of this process is 1, it means that the user wants to perform maintenance, so the getrootfs script is invoked. If the return code is 3 or 4, then you booted from the installation and chose to install an mksysb image file from another one. The bi_main script must be reexecuted in this case so that the extracting and validating steps are performed for this new image. If the return code is different from any of these, the BosMenus is reinvoked.
        - Performs additional space requirement tests. Verification is made to make sure there is enough disk space to hold the logical volumes. If there is an error, return to step 3 (BOSMenus).
        - Invokes the pretest diagnostics on the disks that will be part of the rootvg volume group.
    - LED C46 is shown on the node.

Note

From now on, there is no return to the installation menu. So if an error is detected, the BI_Error function is invoked. If the console is not set to a valid tty, the installation process will stop with LED 623.

20. Invoke the Initialize_Log function. This function initializes the log for the bi_main script (/var/adm/ras/bi.log).

21. Invoke the Shrink_It function. This function sets the size of the LVs and file systems to the minimum value, if the option shrink was set in the

image.data file so that when LVs are created, only the needed space is allocated.

22. Invoke the Prepare_Target_Disks function. This function performs two different flows of execution depending on the installation method that was selected. For an overwrite install, the function performs the following actions:

    - For each disk in the target_disk_data stanzas, it changes the PVID of disk. If there is an error, LED C47 is shown on the node and the BI_Error function is invoked.
    - LED C50 is shown on the console.
    - Invokes the Make_SYS_VG function, which creates the rootvg volume group based on the information of the target_disk_data stanzas from the bosinst.data file. If there is an error creating the volume group, the BI_Error function is invoked.
    - LED C46 is shown on the node.
    - The NIM info attribute for the node is changed to Making boot logical volume.
    - The Make_Sys_LV function is invoked to create the boot (hd5) logical volume. If there is an error creating the logical volume, the BI_Error function is invoked.
    - The NIM info attribute for the node is changed to Making paging space.
    - The Make_SYS_LV function is invoked to create the paging logical volume (hd6). If there is an error creating the logical volume, the BI_Error function is invoked.
    - Invokes the Swapon_hd6 function, which activates the hd6 paging device. If there is an error, LED C51 is shown on the node, and the BI_Error function is invoked.
    - Creates and activates other paging devices that were defined in the image.data file.
    - The NIM info attribute for the node is changed to Making logical Volumes.
    - Creates the logical volumes defined in the image.data file. This is done by invoking the Make_Sys_LV function. If there is an error, the BI_Error function is invoked.
    - The NIM info attribute for the node is changed to Formatting the jfs log.

- The `logform` command is executed to form the jfslog device. If there is an error, LED C49 is shown on the node, and the BI_Error function is invoked.
- The NIM info attribute for the node is changed to Making file systems.
- Creates the file systems defined in the image.data file. Make_Sys_FS is invoked to perform this action. If there is an error, the BI_Error function is invoked.
- The NIM info attribute for the node is changed to Mounting file systems.
- The file systems defined in the image.data file are mounted under the /mnt directory.

If the installation method is migrate, the following steps are performed:

- The NIM info attribute for the node is changed to Importing root volume group.
- The rootvg volume group is imported. If there is an error, the BI_Error function is invoked.
- Swapon_hd6 is invoked to activate the paging space (hd6). If there is an error, LED C51 is shown, and the BI_Error function is invoked.
- All other paging logical volumes are activated.
- The NIM info attribute for the node is changed to Replaying the log.
- The `logredo` command is invoked to replay the jfslog. If there is an error, LED C49 is shown on the node, and the BI_Error function is invoked.
- The NIM info attribute for the node is changed to Preserving old data.
- The file systems hd2, hd9var, and hd4 are mounted.
- The following information is saved, so it can be restored later:
  - Non-rootvg volume group information
  - Existing device customization
  - Printer information
- The NIM info attribute for the node is changed to Copying old files.
- Invokes the BI_Copy function, which retrieves the bos.rte.pre_i function from the bos fileset. The bos.rte.pre_i function performs the premigration steps (saves the configuration file and identifies the fileset that will not be migrated).

23. LED C54 is shown on the node.
24. Restore_System is invoked. This function performs the following steps:

- The NIM info attribute for the node is changed to Restoring Base Operating System.
- The operating system is restored. The mksysb image file is restored if the installation method is overwrite. If the installation method is migrate, the bos file from the lppsource directory is restored.

25. LED C52 is shown on the node.
26. The Initialize_Disk_Environment function is invoked. This function performs the following:
    - The NIM info attribute for the node is changed to Initializing disk environment.
    - If the installation method is migrate and you cannot save the device files, LED C59 is shown and the BI_Error function is invoked.
    - The NIM info attribute for the node is changed to Over mounting /.
    - Invokes the Change_Mounts function, which unmounts the system file systems and mounts them over /. The NIM files are recovered from the RAM file system.
    - Invokes the Copy_Customized function. This function performs the following actions:
        - Changes the NIM info attribute for the node to Copy Cu* from disk.
        - Copies the customized database to the disk file system. If there is an error, LED C59 is shown on the node, and the BI_Error function is invoked.
27. LED C46 is shown on the node.
28. Invokes the Post_Install function, which performs the following miscellaneous post-install procedures:
    - Invokes the merge method so that the ODM database is merged.
    - Sets the remove_inst flag if the REMOVE_INST_ROOT attribute is set to yes in the bosinst.data file.
    - Installs mandatory updates.
    - Installs the device support filesets needed for the adapters on the node.
    - Initializes the dump device.
    - Imports previously defined volume groups if the installation method is migrate.
    - Installs the locales filesets.

- Installs X-Windows filesets if the INSTALL_X_IF_ADAPTER attribute is set to yes in the bosinst.data file.
- Invokes the NIM script.
- Sets up the startup entry in the inittab file.
- Installs up or mp filesets depending on the processor type.
- Creates the boot image.

From a debugging point of view, the Post_Install function performs the following actions:

- The NIM info attribute for the node is changed to Merging.
- LED C54 is shown on the node before starting to install additional software.
- Additional filesets are installed from the lppsource directory.
- Installs additional adapter software from the lppsource directory. This is performed by invoking the `cfgmgr` command with the `-i` flag.
- LED C46 is shown on the node.
- Changes the NIM Cstate attribute to customization is being performed.
- Invokes the program defined in the NIM_CUSTOM environment variable. The file defined in this variable is the /export/nim/scripts/<short_reliable_hostname>.script file. This script performs the following actions:
  - Configures the node as a NIM client.
  - NFS-mounts the pssp_script script.
  - Invokes the pssp_script script.
- Changes the NIM Cstate attribute to post install processing is being performed.

29. LED C56 is shown on the node.
30. The Run_Customization function is invoked, which performs the following actions:
    - The NIM info attribute for the node is changed to BOS install customization.
    - Invokes the command defined in the CUSTOMIZATION_FILE environment variable.
31. LED C46 is shown on the console.
32. The Finish function is invoked, which performs the following steps:

- Saves the installation/migration log files to /var/adm/ras.
- If the key is in service mode, and there were missing device filesets during the post installation procedure (this is always true on an overwrite install, since the support for the switch adapter is not in the lppsource directory), LED C58 is shown and the node waits for an answer from the console keyboard.
- Changes the NIM Cstate attribute to ready for a NIM operation.
- Changes the NIM Mstate attribute to not running.
- Displays the copyrights.
- Reboots the system.

2.3.2.2 The pssp_script script

Let us now focus on the pssp_script script. This script is called during the installation or migration of a node, or when the bootp_response from the Node SDR class for the node is set to customize and the node is rebooted or the rc.sp script is invoked. After running this script, the node should have PSSP filesets installed and should be ready for use.

The functions of pssp_script are:

- Create PSSP log directories. Show LED U20 .

Note

The U codes are displayed in nodes with LED displays. However, in nodes with LCD displays (such as 332 Mhz PowerPC and POWER3-based nodes), the U codes are displayed as 0a code. For example, a U20 code will be displayed as 0a20 on LCD displays.

Create the log file for pssp_script (this is the NODE.config.log.<pid> file on the /var/adm/SPlogs/sysman directory).

- Establish environment definitions. Show LED U21.
  - Set environment variables, and figure out the IP address of the boot/install server and node (different approaches are taken if installing or customizing).
  - LED U03 is shown.
  - Get the <reliable_hostname>.install_info file in the /tftpboot directory from the boot/install server.
  - Show LED U04.
  - Run the install_info file that is retrieved.

  - Set the clock to the clock of the boot/install server.
  - Move the logfile from NODE.config.log.<pid> to <short-reliable_hostname>.log.<pid>.
- Configure nodes (not adapters). Show LED U22.
  - Show LED U57.
  - Get the <reliable_hostname>.install_info file in the /tftpboot directory from the boot/install server.
  - Show LED U59.
  - Get the template file to add the node_number in the ODM database.
  - Delete the ODM entry for node_number if there is one.
  - Add an ODM entry for this node number.
- Create /etc/ssp files. Show LED U23.
  - Show LED U60.
  - Create the /etc/ssp files (they are needed for supper, for instance).
- Update /etc/hosts file. Show LED U24 (only runs if you are installing).
- Get files from the server. Show LED U25.
  - Show LED U61.
  - Get the /etc/SDR_dest_info file.
  - Show LED U79.
  - Get the /tftpboot/script.cust file.
  - Show LED U50.
  - Get the /tftpboot/tunning.cust file.
  - If the node is installing, show LED U54, get the spfbcheck file, show LED U56, and get the psspfb_script script.
  - If the node is customizing, show LED U58 and retrieve psspfb_script.
- Do authentication (Kerberos). Show LED U26.
  - Retrieve the Kerberos configuration files.
  - Show LED U67.
  - Get the krb.conf file.
  - Show LED U68.
  - Get the krb.realms file.

    - Show LED U69.
    - Get the krb-srvtab file.
    - Update the ACLs.
    - Update the Kerberos entries in the /etc/services file if necessary (taking into account the AIX version installed).
- Update the /etc/inittab file. Show LED U27.
- Install the bos.[um]p fileset, and set the MP configuration. Show LED U28.
    - Show LED U52 for MP nodes and U51 for UP nodes.
    - Update the BUMP for SMP High nodes.
    - NFS-mount the location of the lppsource resource for the node, and if there is an error, return.
    - Install the bos.*[um]p fileset, depending on the AIX version and processor type.
    - Install additional device software.
    - Run bosboot, and if there is an error, stop the process with LED U55.
- Install prerequisite software. Show LED U29.
    - NFS-mount the lppsource directory if necessary.
    - Install the perfagent.server fileset if the PSSP level is 2.2, 2.3, or 2.4; install perfagent.tools otherwise.
    - Install the bos.rte.tty fileset if it is not already installed.
    - Install the devices.sio.sa.diag fileset if it is not already installled.
    - Unmount the lppsource directory.
- Install the ssp.clients fileset. Show LED U30.
    - Show LED U80.
    - NFS-mount the PSSP code directory of the node directory.
    - If the PSSP version for the node is less than PSSP 3.1, install the ssp.clients fileset. Otherwise, install the ssp.clients and the rsct.clients.rte filesets.
- Install the ssp.basic fileset. Show LED U31.
    - If the PSSP code version for the node is between PSSP 2.2 and PSSP 2.4, install the ssp.basic, ssp.perlpkg, and ssp.sysman filesets.

    - If the PSSP code version for the node is PSSP 3.1 or later, install the ssp.basic, rsct.basic.rte, ssp.perlpkg, and ssp.sysman filesets.
- Install the ssp.ha filesets. Show LED U32.
    - If the PSSP code version for the node is PSSP 2.4, install the ssp.ha and the ssp.ha_clients filesets.
    - If the PSSP code version for the node is PSSP 3.1 or later , install the ssp.ha_topsvcs.compat fileset.
- Install the ssp.sysctl fileset. Show LED U33.
- Install the ssp.pman fileset. Show LED U34.
- Add the ODM entries for the switch. Show LED U41.
- Install the ssp.css fileset (if the switch_node_num exists in ODM). Show LED U35.
    - Show LED U84.
    - Install the ssp.css fileset.
    - Run the configuration manager.
- Install the ssp.st fileset. Show LED U36.

  If the ssp.st fileset is installed on the CWS, show LED U85.
- Delete the /.rhosts file that was needed for NIM operations. Show LED U37.
- Create the dump logical volume. Show LED U38.
- Start the mirroring/unmirroring process. Show LED U43.
    - If the PSSP code version for the node is less than 3.1, do nothing.
    - Show LED U43.
    - Run the spmirror or spunmirror function to mirror/unmirror the rootvg volume group, based on the information obtained from the Volume_Group SDR class.
- Run the tuning.cust script. Show LED U39.
- Run the script.cust script. Show LED U40.
- Run the psspfb_script script if the bootp_reponse attribute from the Node SDR class is customize.

2.3.3 How to diagnose boot/install problems

This section describes common boot/install problems.

First, you need to know in which phase of the installation process you currently are. If you do not know this, collect the following:

- The LED code information from the node.
- The output from the execution of the following command on the boot/install server:

  ```
  lsnim -l <node-name>
  ```

  Where:

  `<node-name>` is the short reliable hostname of the node.

The output from this command is shown in Figure 62.

```
sp5n13:
    class            = machines
    type             = standalone
    platform         = rs6k
    netboot_kernel   = mp
    if1              = spnet_en1 sp5n13 02608CE8FCB3 ent
    cable_type1      = bnc
    Cstate           = Base Operating System installation is being performed
    prev_state       = BOS installation has been enabled
    Mstate           = in the process of booting
    info             = verifying_data_files
    boot             = boot
    bosinst_data     = 13_noprompt
    lpp_source       = lppsource_aix432
    mksysb           = mksysb_3
    nim_script       = nim_script
    script           = psspscript
    spot             = spot_aix432
    cpuid            = 00085138A400
    control          = master
    Cstate_result    = success
```

Figure 62. lsnim output—The Cstate attribute

If the Cstate attribute from the output of the `lsnim` command is `BOS installation has been enabled`, we are in the initial phase of the installation process, and the bi_main script did not start yet.

If the Mstate attribute is `in the process of booting`, the rc.bos_inst script has not been invoked yet.

If the CState attribute is `Base Operating System installation is being perrformed`, the bi_main script is running, but the pssp_script script has not been invoked yet.

If the Cstate attribute is `customization is being performed`, the node is being configured as a NIM client, or the pssp_script script is currently running on the node.

If the CState attribute is `post install processing is being performed`, bi_main is running, but the pssp_script has already finished.

Note

If the installation fails for some reason, and you want to repeat the boot/installation process, you should invoke the following command:

```
unallnimres -l <node_number>
allnimres -l <node_number>
```

The Cstate attribute from the machine object shows the correct information, and the allocated resource objects are not deallocated in the middle of the installation process.

Now that you know in which phase of the installation process you are, we describe what kind of problems may arise, and how to solve them.

2.3.4 Node conditioning problems

The problems that we cover in this section may arise when the system is network booted but does not reach LED 231/251/260.

The approach used is described in Figure 63 on page 112.

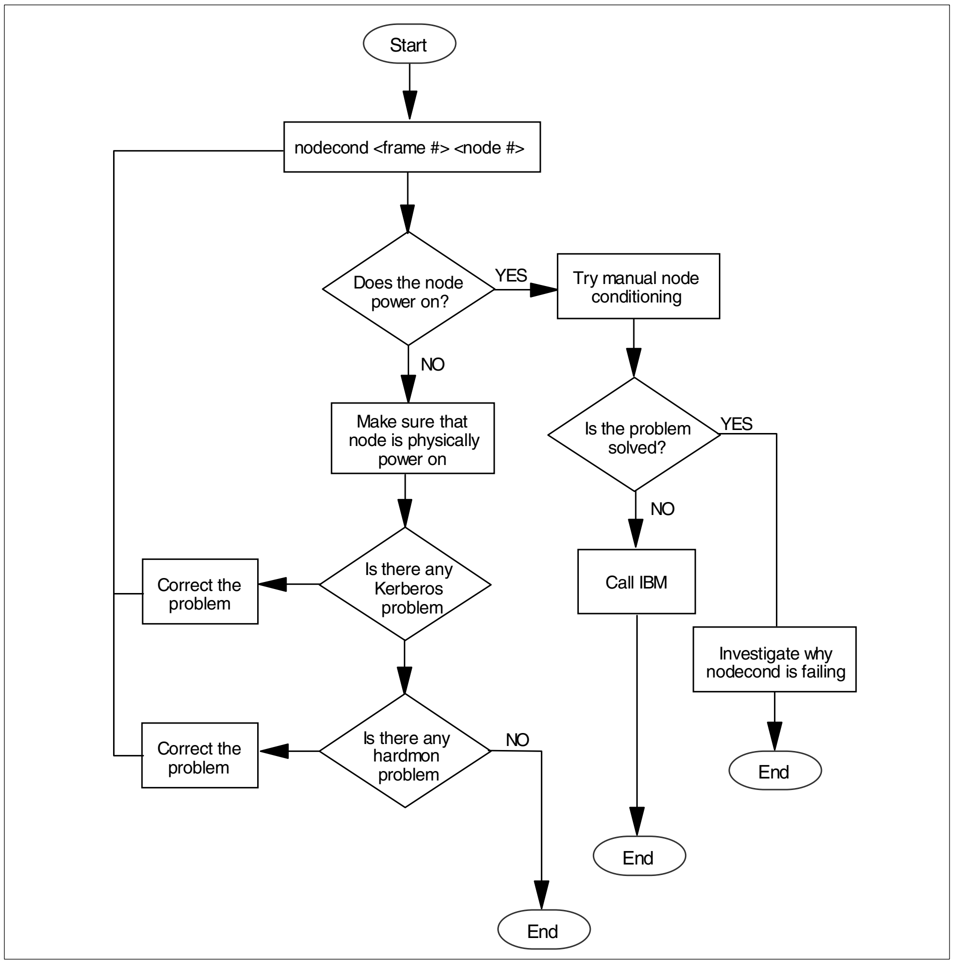

Start
nodecond <frame #> <node #>
Does the node power on?
YES
Try manual node conditioning
NO
Make sure that node is physically power on
Is the problem solved?
YES
NO
Is there any Kerberos problem
Correct the problem
Call IBM
Investigate why nodecond is failing
Is there any hardmon problem
Correct the problem
NO
End
End
End

Figure 63. Node conditioning flowchart

The flowchart can be described as follows:

1. If the power on the node is on, issue the following command to power it off:

   ```
   spmon -p off frame<x>/node<y>
   ```

   Where:

`<x>`	is the frame number that contains the node you want to power off.
`<y>`	is the node number from the selected frame that you want to power off.

2. Perform a network boot by executing the following command:

```
nodecond <frame#> <node#>
```

Where:

`<frame#>`	is the frame number that contains the node you want to power on.
`<node#>`	is the node number from the selected frame that you want to power on.

3. Issue the following command:

   ```
   spmon -l frame<x>/node<y>/powerLED/value
   ```

   If the attribute returned by this command is `1`, the node is powered on. If it is `0`, the node is powered off.

4. If the power on attribute of the node is on, and you still have problems, perform the following steps:

   1. Power the node off by executing:

      ```
      spmon -p off frame<x>/node<y>
      ```

   2. Perform the steps defined in 2.3.10.1, “Manual node conditioning” on page 145.

   3. If you still have a problem when performing the manual node conditioning, document your procedures and call your local IBM support center.

   4. If the problem is solved with manual node conditioning procedures, investigate why `nodecond` failed. The `nodecond` log file /var/adm/SPlogs/spmon/nc/nc.<frame#>.<node#> should give a clear indication of the failure. A nodecond log file is shown in Figure 64 on page 114.

```
Nodecond Status: invoking /usr/lpp/ssp/bin/nodecond_mca
Nodecond Status: start frame 1, slot 5
Nodecond Status: get bootp response type from SDR
Nodecond Status: bootp response type is install
Nodecond Status: get default boot device from SDR
Nodecond Status: default boot device is en0
Nodecond Status: get Ethernet type from SDR
Nodecond Status: Ethernet type is bnc
Nodecond Status: get nodes card type
Nodecond Status: nodes card type is 161
Nodecond Status: get node type (thin/wide/high)
Nodecond Status: node type is high
Nodecond Status: power off the node
Nodecond Status: open S1 port
Nodecond Status: change key to Service
Nodecond Status: sending <enter> to wake up BUMP
Nodecond Status: sending <sbb> to start BUMP
Nodecond Status: parsing "0604" style BUMP menus
Nodecond Status: Autoservice IPL is "Disabled"
Nodecond Status: Bump Console Present is "Enabled"
Nodecond Status: Fast IPL is "Enabled"
Nodecond Status: power on the node
Nodecond Status: SYSTEM BOOT selected
Nodecond Status: BOOT FROM NETWORK selected
Nodecond Status: in main menu, get adapter address
Nodecond Status: selected adapter matched ==> 3. Ethernet:  Slot 1/1, BNC / modular
Jacks
Nodecond Status: checking IP addresses
Nodecond Status: all IP addresses are zero; continuing
Nodecond Status: go back to main menu
Nodecond Status: in main menu, start network boot
Nodecond Status: change key to Normal
Nodecond Status: start network boot
Nodecond Status: waiting for "Booting . . .  Please wait." menu.
Nodecond Status: sent XON to S1
Nodecond Status: holding the s1 port for 4 minutes 0 seconds
Nodecond Status: network boot proceeding, nodecond is exiting
```

Figure 64. Sample node conditioning log

5. If the power on attribute of the node is off, perform the following steps:
   1. Verify that the power on switch and the circuit breaker on the node are on.
   2. Check if there is a serial cable problem.
   3. Check if the tty0 is enabled, and if so, disable it.
   4. Check if there is a Kerberos problem. Execute `k4list` on the CWS and confirm that the tickets did not expire. If they expired, issue `k4init`.

      If you are having any other kind of Kerberos problems, consult Chapter 3, “Security environment” on page 167.
   5. Check if there is a hardmon problem. On the CWS, execute

```
spmon_itest
```

Analyze the /var/adm/SPlogs/spmon_itest.log file. If the log file is empty, you do not have a problem with the hardmon daemon or hardmon ACL file. If you find errors in the file, consult *PSSP: Diagnosis Guide*, GA22-7350, to identify and correct the problem.

If all these steps have been exhausted, and you still have problems, document your procedures and call the local IBM support center.

2.3.5 231/251/260 LED problems

LED 251/231 appears when the node attempts to retrieve the boot image from the boot/install server.

Figure 65 shows the events during LED 231.

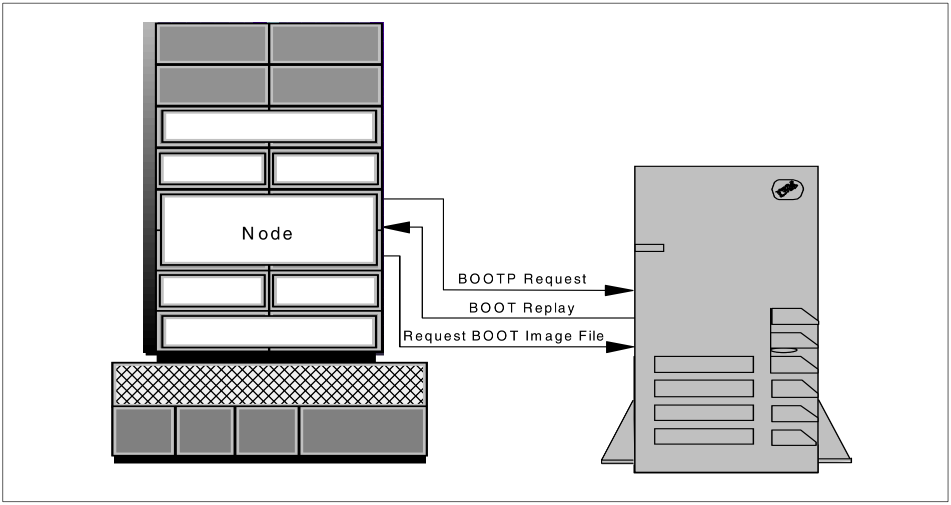


Figure 65. bootp processing

When the boot sequence reaches LED 231/251, open a tty in read-only mode:

```
s1term <frame#> <node#>
```

You should see output similar to what is shown in Figure 66 on page 116.

```
STARTING SYSTEM (BOOT)

Booting . . .  Please wait.


Ethernet:  Slot 1/1, BNC / modular Jacks
Hardware address ................................. 02608C2D08D7


....            Packets Sent      Packets Received

BOOTP             00004              00000
```

Figure 66. Node console with LED 231

The problems that could arise in this phase can be categorized as:

- bootp problems
- tftp problems
- boot file problems

Table 2 summarizes possible causes of LED 231/251 problems.

Table 2. LED 231—Type of problems

Type of Problem	Causes
bootp	Internal network connectivity problem Incorrect SDR information bootps missing in /etc/inetd.conf Entry missing in /etc/bootptab Hardware address mismatch
tftp	tftpd missing in /etc/inetd.conf Definitions in /etc/tftpacces.ctl boot file described in bootptab file does not exit Incorrect boot file permissions
boot file	boot file corrupted boot file not compatible with platform type Other machine sent the bootp reply

Let us start with the bootp problems.

The first thing to do is to verify that there is no network connectivity problem. The procedures for doing this are as follows:

1. Power off the node.
2. Open a tty connection to the node in read/write mode. Execute:

   ```
   s1term -w <frame#> <node#>
   ```

3. Perform a manual node conditioning (see 2.3.10, “Problem determination tool” on page 143).

```
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 67. Network boot menu

4. After obtaining the menu described in Figure 67, perform the following steps:
   1. Select option `1`. Select **BOOT** (Startup) Device.
   2. Select the Ethernet interface by
      - Choosing the first Ethernet adapter.
      - Choosing the correct connection type (bnc/tp/aui).
   3. Check if any of the following fields have been defined:
      - Client address
      - BOOTP server address
      - Gateway address
   4. If any of the previous fields have been set, check if they have the correct information. The following steps must be performed:
      1. Invoke `splstdata -b` on the CWS to identify the boot/install server of the node. On the boot/install server, get the IP address of the interface that connects to the node and verify that it is the same that

is defined in the BOOTP server address field. If it is not, clear the data in this field.

2. Look for the IP address in the Client address field on the panel and verify that it is the same that is defined in the SDR. This is done by issuing:

   ```
   SDRGetObjects Adapter adapter_type==en0 node_number==<x> netaddr
   ```

3. Check the Gateway address field on the panel. If this field contains an IP address, clear the field.

Note

Even if you did not find any incorrect information in these fields, you should clear the fields if they contain data, because if you change the boot/install server of the node you do not want to go through the same process again.

5. Select `99` to go to the Main Menu panel of the installation process.
6. From the Main Menu, select option `3` to test the network connectivity with the boot/install server. Enter the data for the fields and start the ping test. Figure 68 on page 119 shows the Send Test Transmission panel.

```
SEND TEST TRANSMISSION (PING)

A test to see if the machine at the origin
address can communicate, thru the network, with the
machine at the destination address.

Currently selected BOOT (startup) device is:
Ethernet:  Slot 1/1, BNC / modular Jacks
Hardware address .................................... 02608C2D08D7

Select an address to change or select "4" to begin the test.

1. Origin address                                     192.168.006.009
2. Destination address                                192.168.006.001
3. Gateway address                                    000.000.000.000
     (Optional, required if gateway used)
4. START PING TEST

99. Return to Main Menu


Type the number for your selection, then press "ENTER"
```

Figure 68. Ping test panel

If the ping test was unsuccessful, you have a network connectivity problem. In this case, the following components should be tested:

- Ethernet cables.
- If you are using 10/100baseT, check the hubs/switch.
- If you are using 10Base2, check the terminators. Verify that they are 50 ohm terminators.
- Verify that the Ethernet cable is connected to the correct adapter.

If the ping test was successful, perform the following steps:

- Select `99` to go to the Main Menu of the boot process.
- Select `4` to start the system boot.

If the network boot is successful, the problem is solved. The causes of the initial problem can be the following:

- There was wrong information in the fields on the selected adapter panel. If this was the case, the problem is solved and there is nothing else to do.

- The SDR contains wrong information. In this case, issue the following command:

  ```
  SDRGetObjects Adapter node_number==<x> adapter_type==en0 enet_rate
  enet_type duplex
  ```

  Compare the output with the selections you made on the panel to select the adapter. You could have selected BNC as the adapter type when populating the SDR, but the actual Ethernet type is TP.

  To solve this problem, execute the following command:

  ```
  SDRChangeAttrValues Adapter node_number==<x> adapter_type==en0
  enet_rate=<enet_rate> enet_type=<enet_type> duplex=<duplex>
  ```

Note

Before making any changes to the SDR, execute the `SDRArchive` command to back up the SDR.

5. If you did not find any inconsistency between the selection made on the panel and SDR information, you have a `nodecond` command problem. Analyze the /var/adm/SPlogs/spmon/nc/nc.<frame#>.<node#> log file.

Now that you have verified that there is no network connectivity problem, investigate why the bootp reply is not received by the node.

The first thing to do is to check whether the inetd subsystem is running on the boot/install server. This is done by issuing: `lssrc -s inetd`

If the inetd is not active, start it up with: `startsrc -s inetd`

Now, check the /etc/inetd.conf file on the boot/install server of the node. Verify that the file has an entry as shown in Figure 69.

```
bootps  dgram udp wait root /usr/sbin/bootpd bootpd /etc/bootptab
```

Figure 69. bootps entry in the inetd.conf file

If the entry is commented out or does not exist, uncomment or create the entry and refresh the inetd subsystem by issuing: `refresh -s inetd`

If the problem is still not solved, check the /etc/bootptab file on the boot/install server of the node. Figure 70 on page 121 shows a sample entry in the /etc/bootptab file.

```
sp5n09.msc.itso.ibm.com:bf=/tftpboot/sp5n09.msc.itso.ibm.com:ip=192.168.6.9:ht=Ether
net:ha=02608C2D08D7:sa=192.168.6.1:sm=255.255.255.0:
```

Figure 70. bootptab entry

If there is no entry in the bootptab file for the node, you have a setup_server problem on the boot/install server of the node. To solve the problem, perform the following steps:

1. Confirm that the bootp_reponse in the SDR is install or migrate by executing: `splstdata -b`
2. Execute `setup_server` on the boot/install server (the bos_inst NIM operation on this node in the allnimres wrapper should create the entry in the bootptab file).

If there is an entry in the bootptab file for this node, check the hardware address. Compare the ha=<mac> address obtained from the /etc/bootptab file with the Hardware address field from the ping test screen (Figure 68 on page 119). If they are different (the adapter was probably replaced and the appropriate procedures were not followed), issue the following command:

```
SDRGetObjects Node hdw_enet_addr node_number==<x>
```

If the hardware address obtained by this command is the same as the one obtained from the bootptab file, but is different from the one obtained from the panel on the node, the SDR and NIM contain invalid information. In this case, perform the following steps on the boot/install server of the node:

1. Invoke: `SDRChangeAttrValues Node node_number==<x> hdw_enet_addr =<mac>`

Note

Before making any changes to theSDR, execute the command `SDRArchive` to back up the SDR.

2. Invoke: `nim -Fo reset <node_name>`
3. Invoke: `unallnimres -l <node_number>`
4. Invoke: `delnimclient -l <node_number>`
5. Invoke: `mknimclient -l <node_number>`
6. Invoke: `allnimres -l <node_number>`

Or use `setup_server` instead of steps 5 and 6.

If the hardware address obtained by the `SDRGetObjects` command is the same as the one obtained from the node panel, but is different from the one obtained from the bootptab file, the NIM master contains invalid information. In this case, perform the following steps on the boot/install server of the node:

1. Invoke: `nim -Fo reset <node_name>`
2. Invoke: `unallnimres -l <node_number>`
3. Invoke: `delnimclient -l <node_number>`
4. Invoke: `mknimclient -l <node_number>`
5. Invoke: `allnimres -l <node_number>`

Or use `setup_server` instead of steps 4 and 5.

If these procedures have been followed, and you still have bootp problems, contact your local IBM support center.

If you have a bootp reply, the process continues with a tftp request from the node to the boot/install server.

If the problem is a tftp problem, do the following:

1. Check whether the /etc/inetd.conf file on the boot/install server has an entry similar to the one shown in Figure 71.

```
tftp     dgram  udp6    SRC     nobody  /usr/sbin/tftpd         tftpd -n
```

Figure 71. The tftpd entry in the inetd.conf file

2. If the entry is commented out or does not exist, uncomment or create the entry and refresh the inetd subsystem with `refresh -s inetd`.
3. Now, verify the bootptab file.

Pick the line with the hardware address (ha field) that is equal to the hardware address from the panel on the console, and check the following fields:

- The node hostname (first field in the bootptab entry)

  If this field is incorrect, the hostname of the node was changed and the appropriate procedures were not followed. Refer to Appendix H of PSSP: *Administration Guide* , SA22-7348, for more information.

- The IP address of the node (ip field)

If this field is incorrect, the IP address of the node was changed and the appropriate procedures were not followed. Refer to Appendix H of *PSSP: Administration Guide* , SA22-7348, for more information.

- The IP address of the boot/install server (sa field)

  If this field is incorrect, the IP address of the boot/install server was changed and the appropriate procedures were not followed. Refer to Appendix H of *PSSP: Administration Guide*, SA22-7348, for more information.

- The network mask (sm field)

  If the subnet mask is not correct, the SDR information was incorrectly populated. To solve the problem, perform the following steps:

  1. Invoke: `SDRGetObjects Adapter node_number==<x> adapter_type==en0 netmask`
  2. If the netmask obtained from this command is not correct, execute:

     ```
     SDRChangeAttrValues Adapter node_number==<x> adapter_type==en0
     netmask=<correct_netmask>
     ```

Note

Before making any changes to the SDR, execute the command `SDRArchive` to back up the SDR.

  3. Invoke: `nim -Fo reset <node_name>`
  4. Invoke: `unallnimres -l <node_number>`
  5. Invoke: `delnimclient -l <node_number>`
  6. Invoke: `mknimclient -l <node_number>`
  7. Invoke: `mkinstall`
  8. Invoke: `mkconfig`
  9. Invoke: `export_clients`
  10. Invoke: `allnimres -l <node_number>`

  Or use `setup_server` instead of steps 4 to 8.

- The boot file (bf field)

  Verify the following:

  1. The boot file must exist on the boot/install server. If it does not exist, execute the following commands on the boot/install server:

Verify that the bootp_response in the Node SDR class is set to install/migrate:

```
nim -Fo reset <node_name>
unallnimres -l <node_number>
nim -o check <spot_name>
allnimres -l <node_number>
```

2. The directory where the boot file resides must have the following permissions: rw-r--r--.
3. The node boot file in tftpboot (both the link and the generic boot file) must have the following permissions: rw-r--r--.

Check whether there is a tftpaccess.ctl file problem.

The /etc/tftpaccess.ctl file should be similar to the one shown in Figure 72. The directory where the boot file resides must be present in the /etc/tftpaccess.ctl file.

```
# 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
```

Figure 72. The tftpaccess.ctl file

If, after this, you still do not see any tftp activity on the console screen of the node, do a local tftp test on the boot/install server to see whether there is a tftpd problem. Execute:

```
tftp -go /tmp/trash <boot_server_ipaddress> <bootfile> image
```

If you cannot do the local tftp test successfully, check the permissions of the /usr/bin/tftpd daemon. These must be r-sr-xr-x, and the owner should be root. If these are not correct, you can set them with the commands `chmod 4555` and `chown root`.

If, after performing these steps, you are still unable to do this file transfer, contact your local IBM support center.

If you can do the local tftp test, then, since the client node does not receive packets, it is possible that another machine on the network sent the bootp reply. It could be a previous boot/install server that still has an entry in the bootptab file for the node but does not have the bootfile that is defined in the bootptab file. The only way to be sure that this is not happening is by doing a trace of the network (use `iptrace` in promiscuous mode).

If you see on the node console that tftp packets are received, but then are asked again to define the network adapter to boot, there is a problem on the boot file that the node received. The system tests the received boot image by reading the boot record (the first 512 bytes of the bootfile), and if it does not find the magic word ("IBMA" in EBCDIC for an MCA node, 0x7F"ELF" for a CHRP node) at the beginning of the boot record, the node concludes that this is not a valid boot image file; so, the process of trying to retrieve the boot image file is repeated.

If you suspect that the boot file is corrupted, run the command shown in Figure 73.

```
# dd if=/tftpboot/<bootfile_name> ibs=1 count=4 2>/dev/null | od -x
0000000  c9c2 d4c1 ------ for MCA nodes
0000000  7f45 4c46 ------ for CHRP nodes
```

Figure 73. Magic words on boot image

If the output is different than what is shown in Figure 73, then re-create the boot image file with:

```
nim -Fo check <spot_name>
```

and recheck the boot file.

Verify that the boot file in the /tftpboot directory is linked to the correct file. The file name should have the following format:

<spot_name>.<platform>.<processor_type>.ent

where:

<platform> is chrp or rs6k

<processor_type> is up or mp

Retrieve the platform and type of the node from the SDR and see if they are not the same. If they are not, the information on the SDR or on NIM is not correct. To retrieve the information from the SDR, run:

```
SDRGetObjects Node processor_type platform node_number=<x>
```

If the information on the SDR is correct, then correct the NIM database wih the following commands on the boot/install server:

1. `nim -Fo reset <node_number>`
2. `unallnimres -l <node_number>`
3. `delnimclient -l <node_number>`
4. `mknimclient -s <boot/install_server_node_number>`
5. `nim -o check <spot_name>`
6. `allnimres -l <node_number>`

or use `setup_server` instead of steps 4 through 6.

If the information in the SDR is incorrect, perform the following steps on the boot/install server:

1. `nim -Fo reset <node_number>`
2. `unallnimres -l <node_number>`
3. `delnimclient -l <node_number>`
4. `spdelnode -l <node_number>`
5. Enter the node information as if you were installing a new node on the CWS.
6. Run `setup_server` on the boot/install server of the node.

If the boot file has the correct link, execute the following commands on the boot/install server:

1. `nim -Fo reset <node_number>`
2. `unallnimres -l <node_number>`
3. Remove the boot file from the /tftpboot directory
4. `nim -o check <spot_name>`
5. `allnimres -l <node_number>`

and restart the boot process.

2.3.6 Problems after LED 299 but before bi_main is started

If the system crashes with LED 888 just after LED 299, then the boot file is probably corrupted. The boot record has a pointer to the boot code inside the boot image; so, the system passes control to the boot code. If the pointer is not valid or if the boot code inside the boot image is corrupted, the system will crash with LED 888 102 700 or 888 102 300. To solve this problem, re-create the boot image on the boot/install server with the following actions:

1. `nim -Fo reset <node_number>`

2. `unallnimres -l <node_number>`
3. Remove the boot file from the /tftpboot directory.
4. `nim -Fo check <spot_name>`
5. `allnimres -l <node_number>`

and restart the boot process.

If, after this, you still have LED 888, then you could have a corrupted SPOT (the boot image file is created from the SPOT). So, re-create the SPOT on the boot/install server. See 2.2.3, “NIM operations” on page 35, for information on how to re-create the SPOT.

Make sure the lppsource on the CWS has all the "devices" filesets (if you have additional device filesets media, they should be copied to the lppsource directory) for the AIX version you are using before recreating the SPOT. See 2.2.3, “NIM operations” on page 35 for information on how to install and update filesets on the SPOT.

If, after performing these steps, you still have LED 888, contact your local IBM support center.

If the node hangs with an LED that is not described in this section, the node is running a `cfgmgr` command, and the LED that we see is the LED associated with the configuration method for that device.

The `cfgmgr` command runs three times during this phase as follows:

- The first time it is called with the `-f` flag, and it uses only the information from the boot file.
- The second time it is also called with the `-f` flag, but it uses configuration methods from the SPOT.
- The third time it is called with the `-s` flag and uses the full populated ODM from the SPOT plus additional configuration methods.

So, if you have problems with `cfgmgr`, do the following:

1. Verify that all the device filesets are on the lppsource directory.
2. If you have any additional device software media, copy the fileset to the lppsource directory.
3. If you have any preventive maintenance package, copy it to the lppsource directory.

4. Check the lppsource directory. See 2.2.3, “NIM operations” on page 35 to check the software on lppsource.
5. Re-create/update the SPOT. See 2.2.3, “NIM operations” on page 35 on how to re-create, update, and install software on the SPOT.

If you still have the same problem, reinitialize the boot/installation process with NIM debug (see 2.3.10, “Problem determination tool” on page 143).

You should also look for hardware problems (332 MHz nodes stop with LCD 0000 if they do not have the SCSI terminators plugged in or if the adapters are not affixed properly).

Now, let us discuss the LED problems that you could have in this phase:

LED 600, 606, 612, 622 - These only indicate where you are in the installation process.

LED C10, C06, 625 - These are very unlikely to occur; so, we will not explain them.

LED 607 - The system stops the boot process with this LED if it is not able to configure the en0 interface. The reasons could be as follows:

1. Incorrect proto file or corrupted SPOT; so, re-create the SPOT and build new boot files. See 2.2.3, “NIM operations” on page 35 on how to re-create the SPOT and create boot images.
2. The support for this adapter (ent0) is not installed on the SPOT. In this case, copy the filesets that are missing to lppsource and install them on the SPOT. Then re-create the boot image. See 2.2.3, “NIM operations” on page 35 for information on how to install additional software on the SPOT and re-create boot images.
3. Bad level of bos.net.tcp.client on lppsource. Because the SPOT was built from lppsource, copy the last preventive maintenance level to lppsource and update the SPOT. Then re-create the boot image. See 2.2.3, “NIM operations” on page 35 for information on how to update software on the SPOT and re-create boot images.

LED 608 - The node having a problem retrieving the /tftpboot/<reliable_hostname>.info file from the boot/install server.

Confirm on the boot/install server that the file exists. The file name that this node is trying to retrieve is the same as the boot file with the extension .info.

Also, check the permissions of this file; they should be rw-r--r--.

LED 609 - The file /tftpboot/<reliable_hostname>.info in the boot/install server is empty. Re-create it and proceed with the following actions.

Make sure the node is in install, maintenance, migration, or diag mode (run `splstdata -b`), and perform the following steps:

1. `nim -Fo reset <node_name>`
2. `unallnimres -l <node_number>`
3. `allnimres -l <node_number>`
4. Restart the boot/installation process

If the file is still empty, check the file system space. If this is not the case, you have NIM master problems on the boot/install server of the node. You can re-create the NIM master on the boot/install server by running the following commands:

```
delnimmast -l <boot_install_server_node_number>
```

and then setup_server on the boot/install server of this node.

If you are still having problems with LED 609, contact your local IBM support center.

LED 613: The node cannot add the route that is defined in environment variable ROUTES. This error should only appear if the node is on a different subnet than the CWS, and the route in the NIM database is incorrect.

If the node needs an IP route to access the lppsource resource, the gateway address that should be defined in this variable is the IP address of the boot/install interface that connects to this node.

If the installation process is stopped with LED 613, correct the information on the NIM master of the node. Remove the IP route if you do not need it, or change it if you need a route to access lppsource, and the route is not correct. Then, run the following on the boot/install server:

1. `nim -Fo reset <node_name>`
2. `unallnimres -l <node_name>`
3. `delnimclient -l <node_number>`
4. `mknimclient -l <node_number>`
5. `allnimres -l <node_name>`
6. Restart the boot process.

LED 610: This LED could arrive from different scripts. It can be from the mount of the SPOT on the rc.boot script, or it can be from the mounts defined in the environment variable NIM_MOUNTS in the rc.bos_inst script.

So, the first thing to do is to determine which file systems the node is trying to mount. To do this, use

```
lsnim -a info -l <node_name>
```

If the node is trying to mount the SPOT, bos_inst_data, or mksysb resources, then check the /etc/exports file on the boot/install server to see if these file systems are NFS-exported to the node that is trying to mount them.

If they are not defined correctly, change the NFS-export file so that the node can mount the file systems, and try to NFS-mount them locally. To perform the mount, execute:

```
mount -o ro <boot/install_server>:<path_name> /mnt
```

If you are not able to mount them, then there is an NFS server problem on the boot/install server. Consult the AIX documentation to find out how to solve this problem.

If you are able to NFS-mount them locally, then there is an NFS client problem on the node. Do the following:

1. Check the SPOT and boot image. If they are corrupted, re-create them. See 2.2.3, “NIM operations” on page 35 for information on how to re-create the SPOT.
2. Verify that bos.net.nfs.client is correctly installed on the SPOT in the boot server. See 2.2.3, “NIM operations” on page 35 for information on how to check software on the SPOT.
3. Copy the last maintenance level that contains the bos.net.nfs.client fileset to lppsource and update the SPOT from the lppsource resource. See 2.2.3, “NIM operations” on page 35 for information on how to install and update filesets on the SPOT.

The NFS-mount of the lppsource directory can be a different case if the boot/install server of the node is not the CWS, and the node is not on the same subnet as the CWS. In this case, look at the NIM niminfo file on the boot/install server and check if the ROUTES environment variable is set up correctly. The niminfo file is the /tftpboot/<reliable_hostname>.info file on the boot/install server.

If the ROUTES environment variable is not set up properly, define an IP route in the NIM master of this node.

Perform the following steps on the boot/install server:

1. `nim -Fo reset <node_name>`
2. `unallnimres -l <node_name>`
3. `delnimclient -l <node_number>`
4. `mknimclient -l <node_number>`
5. `allnimres -l <node_name>`

If the niminfo file still does not show the correct IP route, then there is some incorrect information in the SDR, in which case, you should delete the node and reenter the node information in the SDR.

If the route is correct, check whether the CWS has an IP route defined to the node network, that is, there should be a route to the node IP network in which the gateway is the boot/install server.

Also, check whether the network parameter ipforwarding is set to 1. If not, issue:

```
no -o ipforwarding=1
```

LED 611: This is the same problem as LED 610, but, in this case, the mount failed (the node stopped the boot/installation process). So, look for the same resolution that is used for LED 610 and restart the installation process.

2.3.7 bi_main installation problems

In this section, we cover the most common problems found during the bi_main phase of the installation process.

The installation of an SP should not require intervention on the node console. So, if you are prompted to enter data on the node console, it is because there is a validation problem on the bosinst.data file or image.data file, or there is a fatal error.

If a fatal error is detected after the validation phase, and you have a valid console, an error is displayed on the console, and you have the option to enter in maintenance mode. If you do not have a valid console, the system stops the installation process with LED 623.

Let us now analyze what kind of LED problems could occur during the bi_main phase.

LED C48: The user is being prompted to enter data on the console, or there is a hang problem in the prediagnostics of the disks that are to be part of the rootvg volume group.

Let us first analyze why a node would change from a non-prompt installation to a prompt installation. This happens if any of the following conditions evaluates to true:

1. The file bosinst.data does not exist.

   This is only possible if the NIM bosinst_data resource was not allocated to this node.

   To trace this error, issue the following command on the boot/install server:

   ```
   lsnim -l <node_name>
   ```

   If the bosinst_data resource is not allocated, then issue the following on the boot/install server:

   ```
   nim -Fo reset <node_name>
   unallnimres -l <node_number>
   allnimres -l <node_number>
   ```

   If the resource is still not allocated, make sure the boot_response attribute of the node is set to install or migrate (run `splstdata -b` on the CWS to check this) and issue the following commands on the boot/install server:

   ```
   nim -Fo reset <node_name>
   unallnimres -l <node_number>
   delnimclient -l <node_number>
   setup_server
   ```

   If there still is a problem, it must be on the NIM master. Use the drastic approach on the boot/install server by issuing:

   ```
   delnimmast -l <boot/install_node_number>
   setup_server
   ```

   If this does not solve the problem, contact your local IBM support center.

   If the bosinst_data resource is allocated by this node, then check the name of the resource by using the following command on the boot/install server:

   ```
   lsnim -l <resource_name>
   ```

   Take note of the "location of the resource". Then, validate that this file exists and has the correct permissions. Also, verify that it is similar to the sample presented in Figure 59 on page 91.

2. The image.data file does not exist.

   This problem could be caused by the fact that the image.data file does, in fact, not exist on the installation media, or there are some file permission problems on the boot/install server.

If you are performing a migration install of the node, the system tries to read the bosinst.data file from the bos file in the lppsource directory. This file is on the CWS since lppsource resides there. So, verify the permissions of that file (as a preventive method for other problems, go to the lppsource directory and run `chmod 755 *` and also `inutoc`). Check whether the lppsource resource is allocated to the node, and if it is not, perform the same steps as in 1.

If you are performing an overwrite install of the node, the system tries to read bosinst.data from the mksysb image on the boot/install server. So, check the permissions of that file on the boot/install server. They should be set to rw-r--r--. If they are not correctly set, check whether mksysb is allocated to the node, and if it is not, perform the same steps as in 1.

Also, test the unlikely possibility that the image.data file does not exist in the install file (bos file or mksysb image file). To check this possibility, perform the following commands:

```
dd if=<install_file> ibs=1k count=128 | restore -Tvqf- ./image.data
```

If the file does not exist on the media, then you have to get another image that has the image.data file.

3. There are no available disks on the node.

   This could be a hardware problem, but it could also be a SPOT problem if there are missing device filesets from the SPOT or the SPOT is corrupted.

   If you are having this problem and suspect that it is a SPOT problem, copy the last AIX preventive maintenance package and the missing devices support filesets to the lppsource directory and update the SPOT on the boot/install server. See 2.2.3, “NIM operations” on page 35 for information on how to install and update software to the SPOT.

   If you suspect that there is a corruption problem, re-create the SPOT. See 2.2.3, “NIM operations” on page 35 for information on how to re-create the SPOT.

4. There are disks in a "defined" state.

   This is a non-fatal error, and you should be able to proceed with the installation by selecting other disks.

5. The entry NIM_BOS_FORMAT=spot is defined in the niminfo file.

   If you have an entry NIM_BOS_FORMAT=spot in the file that is referenced in the "location of resource" attribute for the bosinst_data resource allocated by the node on the boot/install server, perform the following steps:

   1. Issue `lsnim -l <node_name>`.

2. Identify the bosinst_data resource allocated to the node.
3. Issue `lsnim -l <bosint_resource>`.
4. Identify the attribute "location of the resource."
5. Edit the file identified in the last step.
6. Check whether there is an entry NIM_BOS_FORMAT=spot.
7. If there is no such entry, then this problem is not present. Skip the next steps.
8. Issue `nim -Fo reset <node_name>`.
9. Issue `unallnimres -l <node>`.
10. Issue `nim -o remove <bosint_resource>`.
11. Remove the file that was identified in step 3.
12. Issue `mknimres -l <node_number>`.
13. Issue `allnimres -l <node_number>`.
14. Restart the installation process.

6. This node is not migratable (that is, the current AIX level on the selected disks is greater or equal to the AIX level on the bos file in the lppsource directory), and INSTALLATION_METHOD is defined as migrate in the bosinst.data file.
7. The installation method is migrate, but there is no rootvg disk on the system.

   This is a non-fatal error, and you can change the installation method in the installation menu options. But be careful: Since you chose to migrate the node, it had rootvg disks. So, perform diagnostics on the node.
8. The OSLEVEL attribute in the image.data file is not the same as the version and release levels on the SPOT.

   If this is true, you are doing something wrong: Either you have selected an lppsource version that is at the same level as the image, or the image is not at the level you were expecting.
9. The vg_data stanza in image.data does not exist.

   Get another mksysb image file and restart the installation process. If you do not have another mksysb image, try to resolve the problem by performing the following steps on the boot/install server:

   1. `cd /tmp/img`.
   2. Include “allow:/tmp/img” in the /etc/tftpaccess.ctl file.

    3. `restore -xvqf <mksysb_location> ./image.data`.
    4. `chmod 644 image.data`.
    5. Make the necessary corrections to the image.data file.
    6. Execute `lsnim -l <node_name>` and take note of the resource object name for each resource allocated to the node.
    7. `nim -Fo reset <node_name>`.
    8. `unallnimres -l <node_number>`.
    9. `nim -o define -t image_data -a server=master -a location=/tmp/img/image.data imgdata`.
    10. `nim -o allocate -a spot=<spot_name> <node_name>`.
    11. `nim -o allocate -a lpp_source=<lppsource_name> <node_name>`.
    12. `nim -o allocate -a bosinst_data=<bosinst_name> <node_name>`.
    13. `nim -o allocate -a script=psspscript <node_name>`.
    14. `nim -o allocate -a mksysb=<mksysb_name> <node_name>`.
    15. `nim -o allocate -a image_data=imgdata <node_name>`.
    16. `nim -o bos_inst -a no_client_boot=yes -a source=mksysb <node_name>`.
    17. Restart the boot/installation process.

10. There is a missing lv_data stanza in the image.data file for one of the following logical volumes: hd2, hd4, hd5, hd6, hd8, or hd9var.

    Get another mksysb image file and restart the installation process.

    If you do not have another mksysb image, try to resolve the problem by performing the same steps as in 9.

11. There is a missing fs_data stanza from the image.data file for one of the following file systems: hd2, hd4, or hd9var.

    Get another mksysb image file and restart the installation process.

    If you do not have another mksysb image, try to resolve the problem by performing the same steps as in 9.

12. This is an mksysb overwrite installation, and the disks defined in bosinst.data do not have enough disk space to hold all file systems (taking in account the shrink option) and non-jfs logical volumes that are defined in the image.data file.

    There are two possibilities:

    - The first is that the disks defined in the bosinst.data file were not enough to hold the rootvg volume group, but there is enough disk space on the node to restore the mksysb image. This is not a fatal error since you can add more disks to the destination disks through the install menu.

- But, if there is not enough space on all the disks to hold all the data from the mksysb image, then there is no other option than to get another image and repeat the boot/installation process. If you need to put this image on the system, you have to add more disks to the node.

13. EXACT_FIT in the logical_volume stanza is set to yes, but the disks in the image.data file do not correspond to the capacity of the disks on this system.

    This is a non-fatal error. By using the installation menu options, you should be able to solve this problem.

14. SHRINK is set to yes and EXACT _FIT is also set to yes.

    This is a non-fatal error. By using the installation menu options, you should be able to solve this problem.

15. Either PVID or CONNECTION was specified, but neither is equal to the disks on this node.

    This is a non-fatal error. By using the installation menu options, you should be able to solve this problem. Be careful with the selection of the disks. Make sure that the disks you select are actually the disks you want to perform the operation.

16. There is no disk defined in the location specified by LOCATION in bosinst.data.

    This is a non-fatal error. By using the installation menu options, you should be able to solve this problem. Be careful with the selection of the disks. Make sure that the disks you select are actually the disks that you want to perform the operation. The information on the selected volume group in the SDR is probably incorrect.

17. A disk size is specified in the bosinst.data file, but the system does not locate a disk with at least that capacity.

    This is a non-fatal error. By using the installation menu options, you should be able to solve this problem. Be careful with the selection of the disks. Make sure that the disks you select are actually the disks that you want to perform the operation.

18. The size specified in the bosinst.data file is invalid.

    This is a non-fatal error. By using the installation menu options, you should be able to solve this problem. Be careful with the selection of the disks.

Make sure that the disks you select are actually the disks that you want to perform the operation.

19. A disk name was specified in the bosinst.data file, but the system is unable to find it.

    This is a non-fatal error. By using the installation menu options, you should be able to solve this problem. Be careful with the selection of the disks. Make sure that the disks you select are actually the disks that you want to perform the operation. The information on the selected volume group on the SDR probably has incorrect information.

20. The INSTALLATION_METHOD is migrate and the disk defined in bosinst.data does not define a volume group.

    This is a non-fatal error. By using the installation menu options, you should be able to solve this problem. Be careful with the selection of the disks. Make sure that the disks you select are actually the disks that you want to perform the operation. The information on the selected volume group on the SDR probably has incorrect information.

21. The installation method is overwrite, but there is no bootable disk specified in the bosinst.data file.

    This is a fatal error. The best way to solve this problem is to run diagnostics on the disks.

22. The bosinst.data file indicates that we want an overwrite installation and EXISTING_SYSTEM_OVERWRITE is set to no, but the disks defined in bosinst.data specify disks that contain data.

    This is a non-fatal error. By using the installation menu options, you should be able to solve this problem.

23. The installation method defined in bosinst.data file is invalid.

    This is a non-fatal error. By using the installation menu options, you should be able to solve this problem.

24. INSTALL_X_IF_ADAPTER is not defined.

    This is a non-fatal error. By using the installation menu options, you should be able to solve this problem.

25. BOSINST_LANG is invalid and PRODUCT_TAPE is set to yes.

    This is a non-fatal error. By using the installation menu options, you should be able to solve this problem.

26. CULTURAL_CONVENTION is invalid and PRODUCT_TAPE is set to yes.

    This is a non-fatal error. By using the installation menu options, you should be able to solve this problem.

27.MESSAGES is invalid and PRODUCT_TAPE is set to yes.

This is a non-fatal error. By using the installation menu options, you should be able to solve this problem.

28.TCB=C2, but this is a non-C2 media.

This is a non-fatal error. By using the installation menu options, you should be able to solve this problem.

29.There are duplicate lv_data stanzas in the image.data file.

Get another mksysb image file and restart the installation process. If you do not have another mksysb image, try to resolve the problem by performing the same steps as in 9.

30.There are duplicate fs_data stanzas in the image.data file.

Get another mksysb image file and restart the installation process. If you do not have another mksysb image, try to resolve the problem by performing the same steps as in 9.

31.There are duplicate target_disk_data stanzas in bosinst.data.

Correct the volume group information in the SDR, use the `spchvgobj` command or smitty changevg_dialog to enter the correct data in the SDR, and issue the following commands on the boot/install server of this node:

1. `nim -Fo reset <node_name>`
2. `unallnimres -l <node_number>`
3. `mknimres -l <node_number>`
4. `mkinstall`
5. `mkconfig`
6. `allnimres -l <node_number>`

Let us now look at what kind of situation can stop the installation process. If the BI_Error function is called, you should be able to get a shell prompt.

LED C40, C42: These only indicate where you are. It is very unlikely that the node stops with one of these LEDs.

LED C44: The node is waiting for the Get_RVG_Disk process (launched in the background) to finish. If you get stopped with this LED, perform diagnostics on the disks.

LED C45: The installation process stopped; an error occurred while running `cfgcon`. If this LED appears, look for SPOT problems.

If there are missing filesets on lppsource, copy them to lppsource on the CWS and install them on the SPOT of the boot/install server of the node. See 2.2.3, “NIM operations” on page 35 for information on how to install/update the SPOT.

If you suspect that the SPOT is corrupted, re-create it. See 2.2.3, “NIM operations” on page 35 for information on how to re-create the SPOT.

Copy the last preventive maintenance packet to lppsource on the CWS and update the SPOT on the boot/install server of the node. See 2.2.3, “NIM operations” on page 35 for information on how to install/update the SPOT.

This LED is usually associated with the devices.sio.sa fileset (for MCA nodes). Verify that this fileset is in the SPOT by running the following command on the boot/install server:

```
nim -o lslpp -a filesets=devices.sio.sa* spot_<aixlvl>
```

LED C47: Will appear if the installation mode is overwrite and the system cannot set PVID on the disks that are on the rootvg. This could be the same kind of problem described for LED C45. But ,you could have some hardware problem. If you have additional disks on the system, try to install the OS on these other disks.

LED C50: Shown before creating rootvg if the installation method is overwrite. If the system hangs with LED C50, apply the same solution that was discussed for LED C45. But, you could have some hardware problem. If you have additional disks on this system, try to install the OS on these other disks.

LED C51: The system is trying to activate the paging space. If the system stops with this LED, apply the same methodology that was used for LED C45.

LED C49: The system failed the `logform` command if the installation mode is overwrite; it failed the `logredo` command if this is a migration. If the system stops with this LED, apply the same methodology that was used for LED C45 if you are performing an overwrite install. If you are performing a migration, stop the installation process here and redefine the SDR information so that you can perform maintenance with the actual AIX version on the node.

LED C54: The system is restoring the image. This may take some time; so, do not be impatient with this LED (and usually you can see if there is progress with the restore). If the installation fails with this LED, you can have network problems, or the mksysb image file is corrupted.

Make sure that you use the "no packing" option flag when you execute mksysb to create an image.

The man page of the `backup` command has the following warning about the packing flag:

> This option should only be used when backing up files from an inactive file system. Modifying a file when a backup is in progress may result in corruption of the backup and an inability to recover the data. When backing up to a tape device that performs compression, this option can be omitted.

If you have enough space on another node, try to restore the image and make sure that there is no problem with the image file (do not use `-T` to verify the backup since this option does not detect packing problems). Perform the following steps:

```
cd /very_large_filesystem
restore -xvf<image_file>
```

LED C52: If there is an LED C52 problem, then there is probably some incompatibility problem between the SPOT and the restored mksysb image. So, check whether the version, release, and maintenance levels are the same from both the SPOT and the mksysb image. The LED C52 problem usually occurs with the “symbol not found” problem because you have mounted the file system on root; so, the executables are read from the disk file system instead of RAM, but the libc.a library comes from the SPOT.

LED C59: If the system is saving RAM file system information to disk, it means that you probably have a full file system. Enter maintenance mode and try to remove temporary files from the file system.

LED C58: The installation is finished, and you have to change the key switch to normal and press **Enter** on the console.

LED C46: This is presented in many functions of the bi_main script and in different phases. To debug a LED C46 problem, the best approach is to look at the info field in the output of the command `lsnim -l <node_name>`.

With this information, you should be able to restrict the function that is being called. Look for SPOT problems and missing filesets on lppsource. If you can install the image on a different node, then there is a possibility of a hardware problem, and you should contact your local IBM support center.

Also, when migrating, you should be sure that the rootvg volume group is not corrupted and that there is no problem with any of the system's file systems or paging devices or jfslog devices before starting the migration process.

If you are still not able to solve the problems, put NIM in debug mode. See 2.3.10, "Problem determination tool" on page 143 for information on how to put NIM in debug mode and analyze the failure.

2.3.8 pssp_script problems

In this section, we cover pssp_script problems.

If pssp_script was not run, there should be a permission problem on the /spdata/sys1/install/pssp/pssp_script file on the boot/install server or on the /export/nim/scripts/<node_name>.script file.

The pssp_script log file is located in the directory /var/adm/SPlogs/sysman and contains a good description of what the script is doing. Also, the LEDs indicate the function that the script is running.

Confirm that the perfagent.server is present on the lppsource directory on the CWS installed for PSSP Version 2, or on perfagent.tools for PSSP 3.1 or later.

Problems with pssp_script are usually related to the following:

- NFS mounts

  Take a look at the permissions in the files under the /spdata/sys1/install/pssplpp/<pssp_level>/ directory on the boot/install server. As normal maintenance, you should run:

  ```
  cd /spdata/sys1/install/pssplpp

  chmod -R 755 <pssp_level>

  cd <pssp_level>

  inutoc .
  ```

  Also, look at /etc/exports on the boot/install server and verify that the directory is exported and that the node has permissions to mount it.

- The creation of the boot image on the boot disk

  You should see an LED U55 if there are missing filesets on the lppsource directory, or if you do not have enough space on the /tmp file system.

  Previous levels of PSSP had some problems with LED U55 if there is more than one logical volume name beginning with hd5, or if the hd5 logical volume has more than one logical partition.

- Customization scripts that were modified (for instance, script.cust and psspfb_script)

  This is another common failure in pssp_script. Test the customization scripts (and run them if possible).

If you have a hang problem with pssp_script during installation/migration of a node, then perform the following:

- Power off the node.
- Execute `spbootins -r maintenance -l <node_number> -s no` on the CWS.
- Execute `allnimres -l <node_number>` on the boot/install server.
- Network boot the node.
- Select the option to enter in the maintenance shell.
- Remove not needed files from /tmp.
- Execute `bosboot -a -d hdisk<x>`, where `<x>` is the boot disk number.
- Execute `sync;sync;sync`.
- Power off the node.
- Execute `spbootins -r customize -l <node_number> -s no` on the CWS.
- Execute `setup_server` on the boot/install server.
- Power on the node in normal mode.
- Analyze the pssp_script log file to determine why it failed and correct it.

2.3.9 Rebooting problems after installation

The most common cause of problems after installation is LED 231, which means that the bootlist is not correct, or that the disks do not have a valid boot file if the LEDs are cycling through 223, 229, and 231.

To solve this problem, perform the following steps:

1. Configure the node to boot in maintenance mode:

   ```
   spbootins -r maintenance -l <node_number>
   ```

2. Perform a network boot of the node.
3. Open a tty in read/write mode after LED 299 by executing:

   ```
   s1term -w <frame#> <node#>
   ```

4. In the installation menu, select the **maintenance shell** option.
5. If the system is able to vary on the rootvg volume group, execute:

```
bootlist -m normal -o.
```

6. Verify that these are the boot devices that were expected. If not, set the bootlist devices by executing:

   ```
   bootlist -m normal <dev1> <dev2> ... <devn>
   ```

7. Invoke the command:

   ```
   bosboot -a -d /dev/hdisk<x>
   ```

8. Reboot the system.

The 223, 229, and 231 cycling could also indicate a problem on the PVID on the boot record of the disk. If the boot record does not have the correct PVID, you can try to perform steps 1 through 4. If you are able to vary on, and the `bosboot` command does not solve the problem, you can get the PVID value from the ODM and change it on the boot record with a C program.

2.3.10 Problem determination tool

The problem determination tool to use for debugging installation and migration problems is the NIM debugging facility. To enable NIM debug, perform the following steps on the boot/install server:

First, confirm that the resource object is not allocated with the command:

```
lsnim -a alloc_count <spot_object>
```

If `alloc_count` is not `0`, there are machine objects allocating this resource object. Use the following command to list them:

```
lsnim -a spot
```

For each machine object that has the SPOT object allocated, execute:

```
nim -Fo reset <machine_object>
nim -o deallocate -a spot=<spot_object> <machine_object>
```

Then issue:

```
lsnim -a alloc_count <spot_object>
```

and confirm that `alloc_count` is `0`.

Now, you can enable NIM debug mode by executing:

```
nim -o check -a debug=yes <spot_object>
```

The next step is to invoke setup_server so that the resources that were deallocated are allocated back.

Before manual node conditioning the node, execute:

```
lsnim -l <spot_name>
```

You should get the output shown in Figure 74 on page 144.

```
spot_default:
   class         = resources
   type          = spot
   enter_dbg     = "chrp.mp.ent 0x001f3d1c"
   enter_dbg     = "rs6k.mp.ent 0x001f3d1c"
   enter_dbg     = "rs6k.up.ent 0x001bd844"
   Rstate        = ready for use
   prev_state    = verification is being performed
   location      = /spdata/sys1/install/default/spot/spot_default/usr
   version       = 4
   release       = 3
   mod           = 2
   alloc_count   = 0
   server        = master
   if_supported  = chrp.mp ent
   if_supported  = rs6k.mp ent
   if_supported  = rs6k.up ent
   Rstate_result = success
   plat_defined  = chrp
   plat_defined  = rs6k
   plat_defined  = rspc
```

Figure 74. Sample lsnim output

Take note of the enter_dbg attribute for the type of node that you plan to put in debug mode.

Now, perform a manual node conditioning of the node.

After the node starts to boot from the network, you should get the output presented in Figure 75 on page 145.

Now, start collecting the output from the console. On the aixterm window, press **Ctrl** and the left mouse button and select **logging**. This creates an AixtermLog.<ext> file in your home directory.

```
   GPR0  00000003 001F384C 001F3AB8 00000001 003034F0 001EC4F8 00000000 00000000
   GPR8  00000000 00000000 001EC648 00003006 001E8578 0044A150 0000009C 00000020
   GPR16 00000020 0800004C 02000000 07FFFD25 07FD2000 00000000 00003E88 00000000
   GPR24 00003A60 00003A5C 0044A150 0044A000 00003A58 000C0B08 000034E0 00000001
   MSR 00021030  CR   44000242  LR   001E85A0  CTR   00000000  MQ   00000000
   XER 20000000  SRR0 0007EE68  SRR1 00021030  DSISR 40000000  DAR  00000000
   IAR 0007EE68  (ORG+0007EE68)  ORG=00000000 Mode: VIRTUAL
   0007EE60 8000116C 00000000 7C810808 4E800020 |...l....|...N.. |
   0007EE70 60000000 2C030010 38A00001 41800008 |`...,...8...A...|
   0007EE60 8000116C 00000000 7C810808 4E800020 |...l....|...N.. |
   0007EE70 60000000 2C030010 38A00001 41800008 |`...,...8...A...|
   0007EE80 38A00000 0C850000 38A00001 1C630150 |8.......8....c.P|
   0007EE90 7CA903A6 80C21994 7CA61A14 38C5FF5C ||.......|...8..\|
   0007EEA0 3860FFFF 84E600A8 48000018 41820010 |8`......H...A...|
   0007EEB0 80060094 7C002040 41820074 84E600A8 |....|. @A..t....|
   0007EEC0 7C071840 4200FFE8 41820010 80060094 ||..@B...A.......|

Trap instruction interrupt.
   >0>
```

Figure 75. Console output

Now, enter the value of the enter_dbg attribute. In our case, we are performing the debug on a 332 MHz SMP node; so, the type is MP, and the platform is CHRP. So, at the >0> prompt, enter:

```
st 001f3d1c 2
```

When the prompt returns (`>0>`), enter `go`. This starts the node boot.

To disable NIM debug, execute:

```
nim -o check <spot_name>
```

Note

On High nodes, if you experience problems with NIM debug turned on, press **Ctrl-Q** to continue processing.

2.3.10.1 Manual node conditioning

The following steps detail the procedures for manual node conditioning on a High node and a CHRP node.

For a High node, do the following:

1. Power the node off by invoking:

   ```
   spmon -p off frame<x>/node<y>
   ```

2. Set the key service by invoking:

   ```
   spmon -k service frame<x>/<node<y>
   ```

3. Open a tty connection to the node by executing:

   ```
   s1term -w <x> <y>
   ```

   where `<x>` is the frame number, `<y>` is the node number.

4. Press **Enter** to get the BUMP prompt, and type `sbb` as shown in Figure 76.

```
>sbb
```

Figure 76. The BUMP console

5. Choose option 1 from the STAND-BY MENU as shown in Figure 77.

```
 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 77. The STAND-BY menu

6. Check whether Fast IPL is enabled. If it is not, then select option 6 to enable it as shown in Figure 78 on page 147.

```
 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 78. Fast IPL

7. Select X until the > prompt reappears.
8. Power on the node with:

   `spmon -p on frame<x>/node<y>`

9. Wait for the MAINTENANCE MENU as shown in Figure 79.

```
 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 79. MAINTENANCE menu

10. Select option 6, and get the SYSTEM BOOT menu shown in Figure 80 on page 148.

```
SYSTEM BOOT



                0> BOOT FROM LIST
                1> BOOT FROM NETWORK
                2> BOOT FROM SCSI DEVICE
```

Figure 80. SYSTEM BOOT menu

11.Select option 1 to boot from the network and the MAIN MENU of the boot/install process appears as shown in Figure 81.

```
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 81. MAIN menu

For a CHRP node, perform the following steps:

1. Power the node off by invoking:

   ```
   spmon -p off frame<x>/node<y>
   ```

2. Open a tty connection to the node by executing:

   ```
   s1term -w <x> <y>
   ```

   where `<x>` is the frame number, `<y>` is the node number.

3. Power on the node by invoking:

   ```
   spmon -p on frame<x>/node<y>
   ```

4. Wait for the panel, as shown in Figure 82 on page 149, to appear.

```
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000


memory      keyboard     network      scsi
```

Figure 82. Boot screen

5. Type `1` after `scsi` appears on the screen.
6. When the SMS Main Menu screen appears, as shown in Figure 83, select option 3 to go to the Utilities Menu.

```
   RS/6000 Firmware
   Version WIL98112
   (c) Copyright IBM Corp. 1997  All rights reserved.
   -------------------------------------------------------------------
   System Management Services

   1  Display Configuration
   2  Multiboot
   3  Utilities
   4  Select Language


                                                          .------.
                                                          |X=Exit|
                                                          `------'
   ===>3
```

Figure 83. SMS main menu

7. When the Utilities Menu appears, as shown in Figure 84, choose option 4 to go to the Remote Initial Program Load Setup Menu.

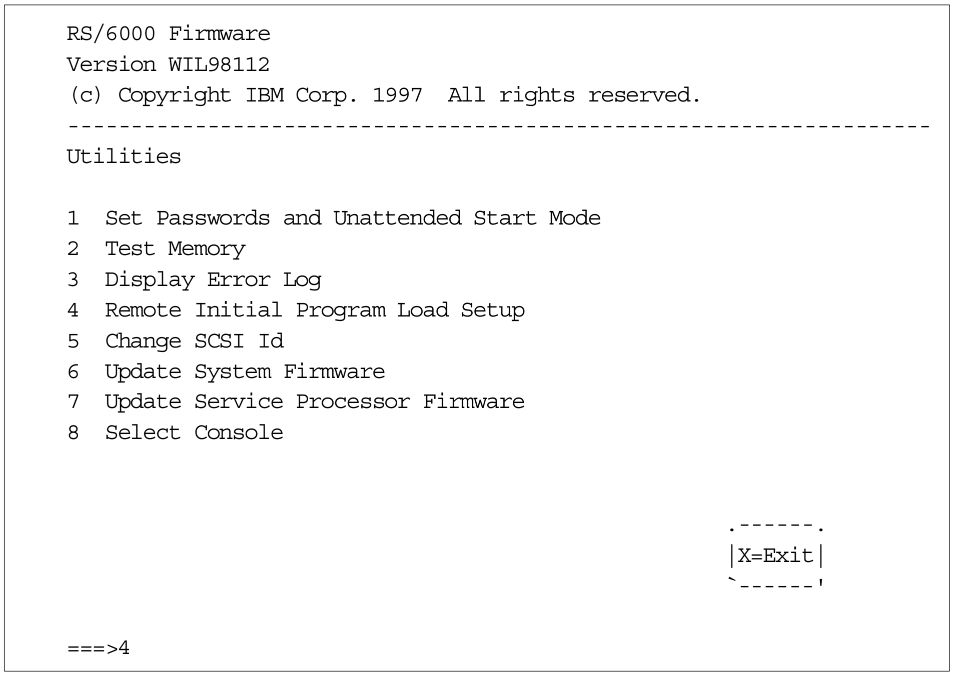

```
RS/6000 Firmware
Version WIL98112
(c) Copyright IBM Corp. 1997  All rights reserved.
-------------------------------------------------------------------
Utilities

1  Set Passwords and Unattended Start Mode
2  Test Memory
3  Display Error Log
4  Remote Initial Program Load Setup
5  Change SCSI Id
6  Update System Firmware
7  Update Service Processor Firmware
8  Select Console



                                                    .------.
                                                    |X=Exit|
                                                    `------'


===>4
```

Figure 84. Utilities menu

8. When the Network Parameters Menu appears, as shown in Figure 85 on page 151, choose option 2 to go to the Adapter Parameters Menu.

```
RS/6000 Firmware
Version WIL98112
(c) Copyright IBM Corp. 1997  All rights reserved.
-------------------------------------------------------------------
Network Parameters

1.  IP Parameters
2.  Adapter Parameters
3.  Ping

                                                   .------.
                                                   |X=Exit|
                                                   `------'
===>2
```

Figure 85. Network parameters menu

9. When the Adapter Parameters Menu appears, as shown in Figure 86, choose the appropriate adapter parameters.

```
RS/6000 Firmware
Version WIL98112
(c) Copyright IBM Corp. 1997  All rights reserved.
-------------------------------------------------------------------
Adapter Parameters

1 .  Ethernet      Integrated    0004ac494b40



                                                   .------.
                                                   |X=Exit|
                                                   `------'
===>1
```

Figure 86. Adapter parameters menu

10. Select X until you get the SMS Main Menu as described in Figure 83 on page 149.
11. Select 2 to go to the Multiboot Menu.
12. When the Multiboot Menu appears, as shown in Figure 87 on page 152, select 4 to go to the Boot Devices Menu.

```
    RS/6000 Firmware
    Version WIL98112
    (c) Copyright IBM Corp. 1997  All rights reserved.
    -------------------------------------------------------------------
    Multiboot

    1  Select Software
    2  Software Default
    3  Install From
    4  Select Boot Devices
    5  OK Prompt
    6  Multiboot Startup <OFF>

                                                   .------.
                                                   |X=Exit|
                                                   `------'
    ===>4
```

Figure 87. Multiboot menu

13. When the Boot Devices Menu appears, as shown in Figure 88, select 3 to configure the first boot device.

```
    RS/6000 Firmware
    Version WIL98112
    (c) Copyright IBM Corp. 1997  All rights reserved.
    -------------------------------------------------------------------
    Select Boot Devices

    1  Display Current Settings
    2  Restore Default Settings
    3  Configure 1st Boot Device
    4  Configure 2nd Boot Device
    5  Configure 3rd Boot Device
    6  Configure 4th Boot Device
    7  Configure 5th Boot Device


                                                   .------.
                                                   |X=Exit|
                                                   `------'
 ===>3
```

Figure 88. Boot Devices menu

14. When the Configure 1st Boot Device Menu appears, as shown in Figure 89, select the appropriate network interface.

```
Configure 1st Boot Device

Device Current    Device
Number Position   Name
1       -         SCSI 4512 MB Harddisk id=0 ( Integrated )
2       -         SCSI 4512 MB Harddisk id=1 ( Integrated )
3       1         Ethernet  ( Integrated )
4                 None



                                                  .------.
                                                  |X=Exit|
                                                  `------'
===>3
```

Figure 89. Configure 1st boot device menu

15. Select X until the SMS Main Menu appears.

16. Select X to start the booting process.

2.3.11 Case studies

In this section, we present case studies related to node installation problems.

2.3.11.1 LED C45 problem

You are trying to install a 332 MHZ SMP node, and it stopped the installation process with LED C45.

From 2.3.3, “How to diagnose boot/install problems” on page 109, we know that LED C45 is caused by an error in the `cfgcon` command. The first thing to do is to verify that the `cfgcon` command is on the SPOT; so, execute the following command on the CWS:

```
lslpp -w | grep cfgcon
```

The resulting output is shown in Figure 90.

```
/usr/lib/methods/cfgcon bos.rte.console File
```

Figure 90. Output of the cfgconf command

This shows that the `cfgcon` command is in the bos.rte.console fileset. Let us now see whether this fileset is in the SPOT by executing:

```
nim -o lslpp -a filesets=bos.rte.console spot_aix432
```

The output is shown in Figure 91.

```
Path: /usr/lib/objrepos
  bos.rte.console            4.3.2.0  COMMITTED  Console
```

Figure 91. Output of the nim -o lslpp command

The fileset is installed in the SPOT. Therefore, the problem must be a device fileset missing from the SPOT. Let us confirm this by restarting the node with NIM in debug mode (see 2.3.10, “Problem determination tool” on page 143 for information on how to put NIM in debug mode).

The output from the node console after restarting the node is presented in Appendix B, “NIM output for the C45 case study” on page 351.

Let us first explain how the `cfgmgr` command works.

The `cfgmgr` command invokes a top level program, based on the phase it is running, from the Config_Rules ODM class. This program defines the top level device and sends its name to stdout. The `cfgmgr` command captures this output and invokes the configuration method program for that device based on the PdDv ODM class. The configuration method is responsible for configuring the device and defining all the child devices. It also sends the child device names to stdout. The `cfgmgr` command once again captures this output and invokes the configuration method for the device until there are no more devices to configure.

Which devices are configured depends on the phase that `cfgmgr` is running.

The `cfgmgr` command runs three times in the network boot process:

- The first time it is called with the `-f` flag (first phase) and only uses the information from the boot file.
- The second time it also runs with the `-f` flag but uses configuration methods from the SPOT.
- The third time it runs with the `-s` flag (second phase) and uses the full populated ODM from the SPOT and additional configuration methods.

So, let us look at the output from the first `cfgmgr` command from the console.

In the first invocation of `cfgmgr`, we are only interested in top level command `defsys`.

```
---------------------------------------------------------------------
invoking top level program -- "/etc/methods/defsys"
return code = 0
****************** stdout ***********
sys0
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'sys0'
invoking /usr/lib/methods/cfgsys_chrp -1 -l sys0
return code = 0
****************** stdout ***********
pci0
:devices.chrp.IBM.TB3MX:devices.chrp.IBM.TB3MX
****************** no stderr ***********
---------------------------------------------------------------------
attempting to configure device 'pci0'
invoking /usr/lib/methods/cfgbus_pcic -1 -l pci0
return code = 0
****************** stdout ***********
:devices.isa_sio.serial:devices.isa_sio.pnpPNP.501
:devices.isa_sio.serial:devices.isa_sio.pnpPNP.501
:devices.pci.ssa:devices.pci.14104500:devices.pci.0c000c02
siokma0,fda0,scsi0,ent0,ent1
****************** no stderr ***********
```

This output shows that there is no PdDv ODM support in the boot image file (obtained from the SPOT) for the following devices because we do not see a device name:

- devices.chrp.IBM.TB3MX. This is the switch adapter; so, it is natural that it is missing in the image file since the support for this adapter is in the ssp.css fileset that is not installed on the SPOT.
- devices.isa_sio.serial.
- devices.isa_sio.pnpPNP.501–Native serial port on CHRP nodes.
- devices.pci.ssa
- devices.pci.14104500–SSA fileset.
- devices.pci.0c000c02–SSA fileset.

So, these filesets are not installed in the SPOT. Therefore, the LED C45 problem can be corrected by installing the missing filesets on the SPOT and recreating the boot image. For information on how to install additional filesets on the SPOT and to re-create the boot image files, see 2.2.3, “NIM operations” on page 35.

The LED C45 problem is solved. We now explain the rest of the `cfgmgr` output because it can be useful in solving other LED problems.

The only adapter that must be configured at this stage is `ent0`:

```
attempting to configure device 'ent0'
invoking /usr/lib/methods/cfgkent -1 -l ent0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
```

We see that the configuration method for `ent0` was successful.

Do not worry about errors from the configuration methods for other devices, since the boot image has only a limited set of configuration methods. The devices that must be configured at this time are pci0 and Ethernet devices, and there is no error configuring these devices.

The second time `cfgmgr` is invoked (once again, in first phase mode), we have the following output:

```
--------------------------------------------------------------------
invoking top level program -- "/etc/methods/defsys"
return code = 0
****************** stdout ***********
sys0
****************** no stderr ***********
-------------------------------------------------------------------
attempting to configure device 'sys0'
invoking /usr/lib/methods/cfgsys_chrp -1 -l sys0
return code = 0
****************** stdout ***********
pci0
:devices.chrp.IBM.TB3MX:devices.chrp.IBM.TB3MX
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'pci0'
invoking /usr/lib/methods/cfgbus_pcic -1 -l pci0
return code = 0
****************** stdout ***********
:devices.isa_sio.serial:devices.isa_sio.pnpPNP.501
:devices.isa_sio.serial:devices.isa_sio.pnpPNP.501
:devices.pci.ssa:devices.pci.14104500:devices.pci.0c000c02
ent0,ent1,siokma0,fda0,scsi0
```

```
****************** no stderr ***********
```

In the second invocation of `cfgmgr`, we are only interested in the top level command `defsys`.

The same device filesets are missing. So, these devices are not configured, and their child devices (for instance, pdisk0 for the SSA adapter) are not detected. But, here we should look carefully for the configuration methods of devices because now `cfgmgr` is running with the configuration methods from the SPOT. So, we should look for configuration method errors.

The output shows no configuration method error for the detected and supported devices (from the `defsys` top level program). The scsi0 was configured; hdisk0 and hdisk1 were detected and configured.

Let us explain the third `cfgmgr` command output (second phase) because this output is different.

```
invoking top level program -- "/etc/methods/defsys"
return code = 0
****************** stdout ***********
sys0
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'sys0'
invoking /usr/lib/methods/cfgsys_chrp -1 -l sys0
return code = 0
****************** stdout ***********
pci0
:devices.chrp.IBM.TB3MX:devices.chrp.IBM.TB3MX
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'pci0'
invoking /usr/lib/methods/cfgbus_pcic -1 -l pci0
return code = 0
****************** stdout ***********
****************** stdout ***********
:devices.pci.isa
:devices.isa_sio.serial:devices.isa_sio.pnpPNP.501
:devices.isa_sio.serial:devices.isa_sio.pnpPNP.501
:devices.isa_sio.8042:devices.isa_sio.chrp.8042
:devices.isa_sio.fdc:devices.isa_sio.pnpPNP.700
:devices.pci.scsi:devices.pci.00100300:devices.pci.NCR.53C825:device
s.pci.SYM.53C825:devices.pci.NCR.8251S:devices.pci.SYM.8251S:devices
.pci.AAPL.NCR8250S
:devices.pci.Ethernet:devices.pci.22100020
:devices.pci.ssa:devices.pci.14104500:devices.pci.0c000c02
```

```
:devices.pci.Ethernet:devices.pci.23100020
siokma0,fda0,scsi0,ent0,ent1
****************** no stderr ***********
```

In this case (second phase cfgmgr), the output is presented in a different way: It shows the output of all the PdDv ODM class devices that are needed for this node (not only the missing filesets).

Also, check for configuration method errors for the devices. But instead of looking only for the top level device defsys processing, look for the following top level programs also:

- deflvm
- starttty
- defssar

The output is as follows:

```
invoking top level program -- "/usr/lib/methods/deflvm"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
invoking top level program -- "/etc/methods/starttty"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defssar"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
```

In this case, there is no error in top level programs.

2.3.11.2 LED 231 problem

In the process of installing a node, we got LED 231. The node was sending a bootp request. So, we did a manual node conditioning and tested the network connectivity as shown in Figure 92 on page 159.

```
RS/6000 Firmware
Version WIL98112
(c) Copyright IBM Corp. 1997  All rights reserved.
-----------------------------------------------------------------------

Ping IP Address

1.  Client IP Address                   [192.168.4.10]
2.  Server IP Address                   [192.168.4.130]
3.  Gateway IP Address                  [000.000.000.000]
4.  Subnet Mask                         [255.255.255.000]



.---------.                                                    .------.
|E=Execute|                                                    |X=Exit|
`---------'                                                    `------'
===>
```

Figure 92. Ping test

There was no problem. Then, we looked at /etc/inetd.conf, and the entry for bootps was there. The next thing we tried was looking in the /etc/bootptab file as shown in Figure 93.

```
# hd   -- home directory
# bf   -- bootfile
# sa   -- server IP address to tftp bootfile from
# gw   -- gateways
# ha   -- hardware address
# ht   -- hardware type
# ip   -- host IP address
# sm   -- subnet mask
# tc   -- template host (points to similar host entry)
# hn   -- name switch
# bs   -- boot image size
# dt   -- old style boot switch
# T170 -- (xstation only) -- server port number
# T175 -- (xstation only) -- primary / secondary boot host indicator
# T176 -- (xstation only) -- enable tablet
# T177 -- (xstation only) -- xstation 130 hard file usage
# T178 -- (xstation only) -- enable XDMCP
# T179 -- (xstation only) -- XDMCP host
# T180 -- (xstation only) -- enable virtual screen
```

Figure 93. bootptab file

The entry for this node does not exist. But how can this be if setup_server was successful?

This is because in multi-frame configurations, if you do not define the boot/install server in the Volume_Group class, it defaults to the first node in that frame. So, the solution is to change the boot/install server to 0 to make the CWS the boot/install server of the node.

So we ran `smitty changevg_dialog`, and entered the data presented in Figure 94.

```
 Change Volume Group Information

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

                                                        [Entry Fields]
  Start Frame                                        [] #
  Start Slot                                         [] #
  Node Count                                         [] #

  OR

  Node List                                          [10]

  Volume Group Name                                  [aix432rvg]
  Physical Volume List                               []
  Number of Copies of Volume Group                    1 +
  Set Quorum on the Node +
  Boot/Install Server Node                           [0] #
  Network Install Image Name                         []
  LPP Source Name                                    []
  PSSP Code Version                                   PSSP-3.1 +


F1=Help            F2=Refresh          F3=Cancel           F4=List
F5=Reset           F6=Command          F7=Edit             F8=Image
F9=Shell           F10=Exit            Enter=Do
```

Figure 94. SMIT panel

Then, we executed the commands:

```
spbootins -c aix432rvg -r install -l 21 -s no

setup_server
```

and restarted the network boot/installation process. The problem was solved.

2.3.11.3 LED C52 problem

The installation process stopped with LED C52. So, we opened a tty in read-only mode with

```
s1term 1 13
```

and got messages complaining about symbols not defined in libc.a. So, we looked at the info attribute of the machine object on the boot/install server with:

```
lsnim -l sp5n13
```

and got the output shown in Figure 95.

```
sp5n13:
     class          = machines
     type           = standalone
     platform       = rs6k
     netboot_kernel = mp
     if1            = spnet_en1 sp5n13 02608CE8FCB3 ent
     cable_type1    = bnc
     Cstate         = Base Operating System installation is being
performed
     Mstate         = in the process of booting
     info           = Over mounting /
     boot           = boot
     bosinst_data   = 13_noprompt
     lpp_source     = lppsource_aix431
     mksysb         = mksysb_1
     nim_script     = nim_script
     script         = psspscript
     spot           = spot_aix431
     cpuid          = 00085138A400
     control        = master
     Cstate_result  = success
```

Figure 95. lsnim output—The info field

The info field shows that the node is mounting the rootvg file systems. Therefore, the node restored the mksysb image (the file systems were mounted on a temporary directory) and was trying to mount the hard disk file systems in the right place. It started to use the commands from the restored image. This seemed to be an incompatibility between the SPOT and the mksysb image because the libc.a that we were using was loaded in memory and was obtained from the SPOT. Another cause could have been a corrupted image that was restored.

So, how could we be sure that there were no incompatibilities between the SPOT and the mksysb image? First we issued:

```
lsnim -l spot_aix431
```

and got the output presented in Figure 96.

```
spot_aix431:
   class          = resources
   type           = spot
   enter_dbg      = "rs6k.mp.ent 0x001f3d1c"
   enter_dbg      = "rs6k.up.ent 0x001bd844"
   enter_dbg      = "rs6k.up.tok 0x001bd844"
   Rstate         = ready for use
   prev_state     = verification is being performed
   location       = /spdata/sys1/install/aix431/spot/spot_aix431/usr
   version        = 4
   release        = 3
   mod            = 1
   alloc_count    = 1
   server         = master
   if_supported   = rs6k.mp ent
   if_supported   = rs6k.up ent
   if_supported   = rs6k.up tok
   Rstate_result  = success
   plat_defined   = chrp
   plat_defined   = rs6k
   plat_defined   = rspc
```

Figure 96. lsnim output—Looking for SPOT

Then we issued:

```
lsnim -l mksysb_1
```

and got the output presented in Figure 97 on page 163.

```
mksysb_1:
   class       = resources
   type        = mksysb
   Rstate      = ready for use
   prev_state  = unavailable for use
   location    = /spdata/sys1/install/images/bos.obj.node13
   version     = 4
   release     = 3
   mod         = 1
   alloc_count = 1
   server      = master
```

Figure 97. lsnim output—The mksysb object

The version, release, and mod attributes are the same in the mksysb object and in the SPOT resource objects. This is natural; if they were not, the setup_server command would give an error message.

Next, we got the image.data file from the mksysb image file by executing:

```
   cd /tmp
   restore -xvf /spdata/sys1/install/images/bos.obj.node13 ./image.data
```

The extract from the image.data file is shown in Figure 98.

```
image_data:
        IMAGE_TYPE= bff
        DATE_TIME= Mon Aug 24 16:08:10 EDT 1998
        UNAME_INFO= AIX k17n03 3 4 000085628900
        PRODUCT_TAPE= no
        USERVG_LIST=
        OSLEVEL= 4.3.2.0

logical_volume_policy:
        SHRINK= no
        EXACT_FIT= no
```

Figure 98. Extract from the image.data file

The OSLEVEL attribute shows that this is an AIX432 image. So, the information on the mksysb resource object was not correct because we initially had an AIX431 mksysb image file named bos.obj.node13, and we copied an AIX432 image to the same file name. Since the resource already existed on the CWS, it was not re-created.

So, we have to perform the following steps:

1. Invoke `unallnimres -l 13`
2. Invoke `nim -o remove mksysb_1`
3. Copy the correct mksysb image file to the /spdata/sys1/install/images directory
4. Invoke `setup_server`

After restarting the boot/installation process on the node, the problem is solved.

2.3.11.4 LED C48

In an attempt to migrate a node, the process stopped with LED C48.

We executed:

```
lsnim -l sp5n13
```

and got the output presented in Figure 99.

```
 sp5n13:
    class          = machines
    type           = standalone
    platform       = rs6k
    netboot_kernel = mp
    if1            = spnet_en1 sp5n13 02608CE8FCB3 ent
    cable_type1    = bnc
    Cstate         = Base Operating System installation is being
performed
    prev_state     = ready for a NIM operation
    Mstate         = in the process of booting
    info           = prompting_for_data_at_console
    boot           = boot
    bosinst_data   = 13_migrate
    lpp_source     = lppsource_aix432
    nim_script     = nim_script
    script         = psspscript
    spot           = spot_aix432
    cpuid          = 00085138A400
    control        = master
    Cstate_result  = success
```

Figure 99. lsnim output.—The sp5n13 machine object

So, the node is prompting for data at the console. We opened a tty in read-write node with:

```
s1term -w 1 13
```

and got the console screen shown in Figure 100.

```
 Error Warning

Missing image.template file.  The network install
server is not set up correctly.


To reboot in normal mode, turn key to normal (if necessary)
and press reset.

>>> 1 Continue with Install
```

Figure 100. Node console screen

Now, we knew that the node could not read the image.data file, which is read from the bos fileset during a migration. So, we confirmed that the fileset existed in lppsource by executing:

```
nim -o lslpp -a filesets=bos lppsource_aix432
```

The fileset is present. So, we checked whether the image.data file exists in the bos fileset by executing:

```
restore -Tvqf /spdata/sys1/install/aix432/lppsource/bos ./image.data
```

The image.data file exists.

So, we checked the permissions of the bos fileset. They were r--r----- (0440), so that is why image.data could not be read. We changed the permissions and restarted the migration process, which then ended successfully.

Chapter 3. Security environment

The RS/6000 SP uses AIX as the operating system; so, it inherits the security strengths and exposures of AIX. For example, AIX does not have encryption support. In order to secure the environment of SP, an additional element is added to PSSP, known as Kerberos. The purpose of Kerberos is to provide authentication services so that certain distributed components within the SP system can access their resources securely.

In this chapter, we describe how to identify and solve problems related to Kerberos.

3.1 Kerberos

Modern computer systems provide service to multiple users and require the ability to accurately identify the user making a request. In traditional systems, the user's identity is verified by checking a password typed during login; the system records the identity and uses it to determine what operations may be performed. The process of verifying the user's identity is called *authentication*. Password-based authentication is not suitable for use on computer networks. Passwords sent across the network can be intercepted and, subsequently, used by eavesdroppers to impersonate the user. Kerberos is used to remove this vulnerability of the traditional networking system.

Kerberos is a distributed authentication service that allows a process (a client) running on behalf of a principal (a user) to prove its identity to a verifier (an application server, or just server) without sending data across the network that might allow an attacker or the verifier to, subsequently, impersonate the principal. Kerberos optionally provides integrity and confidentiality for data sent between the client and server.

Kerberos was developed in the mid-1980s as part of MIT's Project Athena. As use of Kerberos spread to other environments, changes were needed to support new policies and patterns of use. To address these needs, the design of Version 5 of Kerberos (V5) began in 1989.

PSSP 3.1 and later supports Kerberos Version 4 and Kerberos Version 5, although the Version 4 is required by the sysctl and hardmon facilities.

The implementation of Kerberos in PSSP Version 3 is different than in previous versions. This version, and coming versions of PSSP, plan to exploit the enriched features provided by the AIX 4.3 operating system. The remote commands (`rsh`, `rcp`, `rlogin`, `telnet`, and `ftp`) are now provided by the AIX

4.3. Previously, there were "kerberized" versions of remote commands. But, now PSSP Version 3 (which requires AIX 4.3.2 or later) uses the same set of remote commands, and those remote commands are modified to support multiple authentication methods (Kerberos V4, Kerberos V5, and standard AIX).

The other difference in PSSP Version 3 is that security settings are partition-sensitive. New administrative commands are available to manage the security settings from the Control Workstation (CWS). All nodes within a single partition will have the same set of authentication methods enabled, and on the CWS, the security settings are the combination of all authentication methods enabled for all system partitions.

Since the authentication method is set per partition, nodes get the information about the authentication settings from the SDR. At boot time, the `rc.sp` script will call the `spauthconfig` script to set up the right authentication method for that node. Table 3 shows the options available for using the AIX remote commands within the SP.

Table 3. Options of authentication methods

Options	**Selection criterion**
Kerberos V4 (required) Standard AIX	For root user access using the authenticated remote commands within each system partition
Kerberos V5 (required) Kerberos V4 (provided by DCE) Standard AIX	For the authentication method to enable for each system partition
Kerberos V4 (required) Standard AIX	For the authentication mechanism to be installed and configured for each partition

3.1.1 Kerberos authentication process

In this section, we describe the Kerberos terminology and the Kerberos authentication process in brief. A good understanding of the authentication process helps with Kerberos problem determination.

In an SP, during the installation of the CWS, the Kerberos database gets installed with the basic principals defined. The name *principal* means a user or service that uses the authentication services and is identified in the authentication database. An example of a principal is as follows:

```
root.admin@ITSO.IBM.COM
```

In this example, `root` is the principal, `admin` is the instance, and `ITSO.IBM.COM` is the Kerberos realm. The term *instance*, in the case of a user, represents what authority is given to that user. In the case of a service, the instance represents the node providing the service. An example of a service principal and instance are as follows:

```
hardmon.sp5en0
```

In this example, `hardmon` is the service principal and `sp5en0` is the node providing the service.

The term *realm* represents a Kerberos domain consisting of a number of machines providing authenticated services. The default name of a realm on an SP is the TCP/IP domain name converted to uppercase or hostname if DNS is not used.

Kerberos is a three-way authentication system between the Kerberos server, the Kerberos client, and the service provider. So, a user on the CWS, who is an authenticated client, initiates the communication with the Kerberos server by issuing the `k4init` command (the k4 commands are symbolic links to the Kerberos V4 commands available in the /usr/lpp/ssp/kerberos/bin directory). This is shown in Figure 101 as step 1.

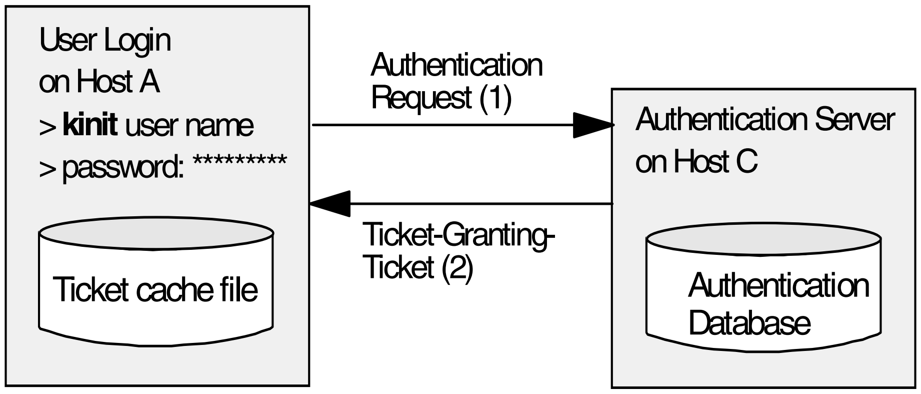


Figure 101. Kerberos client sending request for ticket-granting ticket

The command `k4init` sends a request for ticket-granting services to the Kerberos authentication server. The authentication server checks to see if it knows the client, and then it generates a data packet, known as a ticket-granting ticket, that is composed of the user's name, the current time, the ticket life span, and a randomly generated key, known as the session key.

This is all encrypted in a key known only to the ticket-granting server and the authentication client. In most cases, the authentication server is the same as the ticket-granting server. The authentication server sends back the ticket to the client. This is shown as step 2 in the Figure 101 on page 169. Once the authentication server response has been received by the client, the user is prompted for a password. The password is converted to an encrypted key using the current time. The client uses this encrypted key to decrypt the response from the authentication server.

If the user's password is correct, the ticket sent by the authentication server is stored in the user's ticket cache file, then known as Ticket-Granting-Ticket. For example, the ticket cache file for root looks as follows:

```
/tmp/tkt0
```

Now, we have the ticket-granting ticket for the user or the authenticated client; so, the client can initiate a command to obtain information from a server (in case of SP, a node). For example, consider that the `rsh` command is initiated. The `rsh` command itself builds a packet known as the authenticator containing the client's IP address, the current time, the ticket-granting ticket, and the service name.

The authenticator is encrypted using the session key between this client and the authentication server and is sent to the authentication server. This is shown as step 3 in Figure 102 on page 171. The authentication server decrypts the information and compares all the information of the authenticator and the ticket. If it finds everything okay, the authentication server searches the Kerberos database for a rcmd-command ticket that corresponds to the principal associated to the remote shell service.

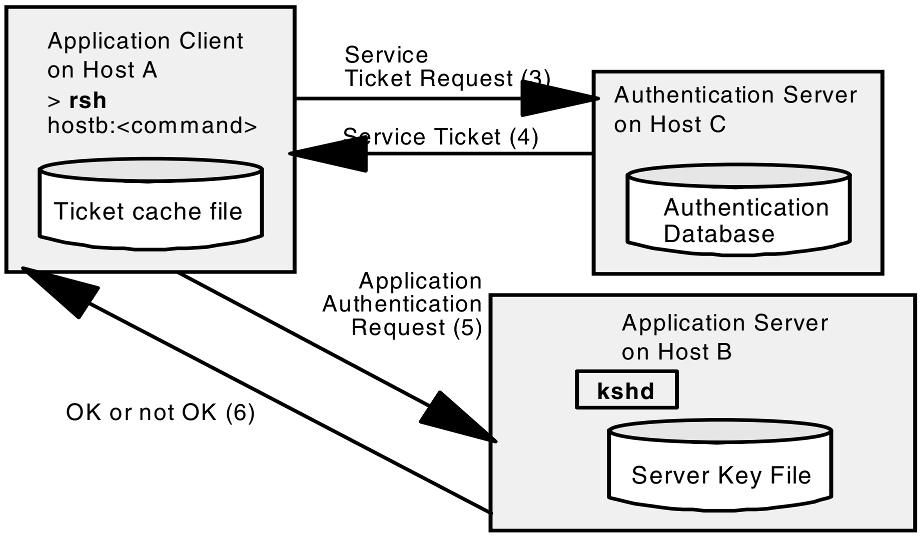


Figure 102. Authentication process after receiving the ticket-granting ticket

The authentication server returns a rcmd-ticket, which also gets stored in the ticket cache file. This is shown as step 4 in Figure 102. Using the `k4list` command, we can display the ticket.

Once the rcmd-ticket is received by the client, it creates an authenticator called auth2 and sends it to the server (in case of SP, a node) along with the rcmd-ticket. This is step 5 in Figure 102.

The krshd daemon on the server (a node) wakes up, decrypts the ticket, and validates the client. If all the tests are passed, the command is executed and the result returns. In Figure 102, this is step 6.

For the sake of simplicity, we described the authentication process from a very high level, for more detailed information, see *The RS/6000 SP Inside Out,* SG24-5374.

3.1.2 Kerberos directories and files

For problem determination, it is important to know about the Kerberos directories and files. Missing or corrupted Kerberos files are often the cause of Kerberos problems. If we know what files Kerberos uses and what their content should be, we can “easily” fix these type of problems.

In an SP, the CWS is usually the authentication server, and on the authentication server, we should find the following files:

- /.k
- $HOME/.klogin
- /etc/krb-srvtab
- /etc/krb-conf
- /etc/krb.realms
- /var/kerberos/database/*
- $KRBTKFILE or /tmp/tkt.<uid>

On the nodes, we should find the following files:

- $HOME/.klogin
- /etc/krb-srvtab
- etc/krb.realms
- /etc/krb-conf
- $KRBTKFILE or /tmp/tkt.<uid>

The explanations of these files are as follows:

/.k: This file gets created when `setup_authent` runs on the CWS. This file holds the Kerberos master password, which is set during the installation of the CWS.

If this file is missing or corrupted, it can be re-created using the `kstash` command. The command is located in the /usr/lpp/ssp/kerberos/etc directory. The only requirement for recreating this file is the Kerberos master password. If the master password is lost, and the file is missing, the workaround is to re-create the Kerberos database. See 3.3.1, “Re-creating the Kerberos database” on page 181 for information about rebuilding the Kerberos database from scratch.

$HOME/.klogin: This is a text file containing the name of principals in the format principal_name.instance@realm. Figure 103 on page 173 shows the content of the file for our four nodes SP. If the file is missing on the node, the `dsh` or `rsh` command will not work on that node. While you perform problem determination, it is necessary to check that all the nodes have this file and that the content is same as that of the CWS or Kerberos authentication server. Usually when the nodes are installed, pssp_script pulls the .klogin file from the boot/install server. If the file is missing or corrupted, it can be copied from the Kerberos authentication server.

The .klogin file is used in the same way as the .rhosts file. The difference between the two files is that .klogin does not specify remote hosts allowed to log in without password but allows remote principals to use a local user account.

```
root.admin@MSC.ITSO.IBM.COM
rcmd.sp5en0@MSC.ITSO.IBM.COM
rcmd.sp5cw0@MSC.ITSO.IBM.COM
rcmd.sp5n01@MSC.ITSO.IBM.COM
rcmd.sp5nb01@MSC.ITSO.IBM.COM
rcmd.sp5n05@MSC.ITSO.IBM.COM
rcmd.sp5n09@MSC.ITSO.IBM.COM
rcmd.sp5n13@MSC.ITSO.IBM.COM
rcmd.""@MSC.ITSO.IBM.COM
```

Figure 103. Sample output of the .klogin file

/etc/krb.conf: This is also a text file containing the local authentication realm and the Kerberos authentication server name for the local Kerberos domain. Figure 104 shows the content of the /etc/krb.conf file on our SP nodes. The first line in the file is the name of the Kerberos realm, and the second line shows the name of the Kerberos authentication server, which is the CWS in this case.

```
MSC.ITSO.IBM.COM
MSC.ITSO.IBM.COM sp5en0.msc.itso.ibm.com admin server
```

Figure 104. Sample of the /etc/krb.conf file

If this file is missing on the nodes, the `k4init` command will fail to communicate with the authentication server. Also, if the file is missing or corrupted on the CWS, the Kerberos daemons (kerberos and kadmind) will not become active after they die. The best problem determination tip for this file is that it has to be available to the nodes, and the content must be the same as that of the CWS. This file is created by the `setup_authent` script on the CWS.

/etc/krb-srvtab: This is a binary file holding the names and private keys of the local instances of the Kerberos authenticated services. This file exists on the CWS and on the nodes. The content of the file is specific to the node or CWS. Figure 105 on page 174 shows the content of the krb-srvtab file on a node called sp5n09. It has an entry for `rcmd` and the instance names are the SP Switch and Ethernet hostnames.

```
# /usr/lpp/ssp/kerberos/bin/k4list -srvtab
Server key file:   /etc/krb-srvtab
Service         Instance        Realm      Key Version
------------------------------------------------------
rcmd            sp5sw09         MSC.ITSO.IBM.COM 1
rcmd            sp5n09          MSC.ITSO.IBM.COM 1
```

Figure 105. Sample content of the krb-srvtab file on a node

Figure 106 shows the krb-srvtab file's content on the CWS, which has the entries for `rcmd` and `hardmon`. Since we have an Ethernet and a Token Ring adapter on the CWS, we see `rcmd` and `hardmon` entries for the Token Ring adapter (`sp5cw0`) instance and Ethernet adapter (`sp5en0`) instance.

```
[sp5en0:/etc] k4list -srvtab
Server key file:   /etc/krb-srvtab
Service         Instance        Realm      Key Version
------------------------------------------------------
rcmd            sp5cw0          MSC.ITSO.IBM.COM 1
hardmon         sp5cw0          MSC.ITSO.IBM.COM 1
rcmd            sp5en0          MSC.ITSO.IBM.COM 1
hardmon         sp5en0          MSC.ITSO.IBM.COM 1
```

Figure 106. Sample output of the krb-srvtab file on the CWS

In both figures, we see the key version used. It must correspond to the versions that are held in the Kerberos database. If we run the `setup_server` script for a node in customize mode, a krb-srvtab file is created in the /tftpboot directory as <nodename>-new-srvtab file. If we do a reboot of the node, `pssp_script` copies the file to the /etc directory and renames the file as krb-srvtab file. The other way to create the file is using the `ext_srvtab` command or the create_krb_files wrapper. See 3.2.3, "Recreating authentication key file using ext_srvtab" on page 177 for information on using the `ext_srvtab` command.

If you are missing this file on a node, `dsh` or `rsh` commands issued to that node will fail.

/etc/krb.realms: This plain text file contains the name of the Kerberos domain or realm. It exists on the nodes and on the CWS. The file should have the same content everywhere within a partition.

/var/kerberos/database: This directory contains binary files that make up the Kerberos Authentication server database. This directory only exists on the authentication server (by default, the CWS). There are two types of files, the Kerberos database files (the principal* files) and the ACL files. These files are set up by the `setup_authent` script.

3.1.3 Kerberos daemons

There are three Kerberos daemons:

- kerberos: This daemon runs on the primary and secondary authentication server. This daemon provides the functionality to process all the authentication requests coming from clients.
- kadmind: This daemon runs only on the primary authentication server, which is usually the CWS. This is the authentication database server.
- kpropd: This runs only on the secondary authentication database server hosts. It updates the secondary server database if changes occur on the primary server database.

From a problem determination point of view, it is important to check if these daemons are active The common setup is running the Kerberos server on the CWS; therefore, we have to make sure that kerberos and kadmind are always active on the CWS. We can start and stop these daemons using the `startsrc` and `stopsrc` commands.

For problem determination of the Kerberos daemons, there are log files available in the /var/adm/SPlogs/kerberos directory. The logs available are kerberos.log for the kerberos daemon, admin_server.sys.log for the kadimd daemon, and kpropd.log for the kpropd daemon.

3.2 Kerberos commands

Here we discuss the usage of some Kerberos commands that are useful for Kerberos problem determination. The commands we discuss are as follows:

- `k4list`
- `kdb_util`
- `ext_srvtab`

The purpose of discussing these commands is to show how we can use the commands in different problem solving scenarios. For syntax and options of these commands see *Parallel System Support Program for AIX Command and Technical Reference*, (Volume 1 and Volume 2) SA22-7351.

3.2.1 Comparing the service key version using k4list

To check the version of the service key in the /etc/krb-srvtab file, we can issue the `k4list` command as shown in Figure 107.

```
# /usr/lpp/ssp/kerberos/bin/klist -srvtab
Server key file:   /etc/krb-srvtab
Service         Instance        Realm      Key Version
------------------------------------------------------
rcmd            sp5sw05         MSC.ITSO.IBM.COM 1
rcmd            sp5n05          MSC.ITSO.IBM.COM 1
```

Figure 107. Checking the service key version in /etc/krb-srvtab file

Figure 107 shows the output of the `k4list` command on node sp5n05. This output shows that the key version in the /etc/krb-srvtab file is 1.

Now, we can take the dump of the Kerberos database on the CWS and check the key version in the database. The key version in the /etc/krb-srvtab file and in the Kerberos database must be same. For details about dumping the Kerberos database, refer to 3.2.2, “Taking the Kerberos database dump using kdb_util” on page 177. You may also use the `lskp` command.

For example, we assume that we have taken the dump of the Kerberos database in a flat file called kdump.file in the /tmp directory. Now we want to check the key version for the rcmd service principal of node sp5n05. To do that, we issue the command shown in Figure 108:

```
[sp5en0:/tmp] grep "^rcmd sp5n05 " ./kdump.file
rcmd sp5n05 255 1 1 0 5a839819 6dc8e7ab 203801010459 199809231405 root
admin
```

Figure 108. Checking the key version of rcmd in the Kerberos database dump

We need to compare the key version if we experience an authentication failure attempting to `dsh` or `rsh` to a particular node.

We can check if the available adapters on a node are registered to the Kerberos database by using the `k4list` command. We may encounter authentication failure, attempting a remote command over a particular network interface. At that time, we should check if the hostname of that interface is registered to Kerberos. If `K4list` output does not show the hostname of the interface we want to use, we may need to add the service principal and also re-create the /etc/krb-srvtab file.

3.2.2 Taking the Kerberos database dump using kdb_util

To avoid creating the Kerberos database from scratch in case of database corruption, it is advisable to keep a flat file dump of the Kerberos database in a secured directory. The command that enables you to take the a dump of the Kerberos database is `kdb_util dump`. The syntax of the command is as follows:

```
kdb_util dump /var/kerberos/database/dump_<datestamp>
```

Figure 109 shows the content of a flat file dump of the Kerberos database. To reload the Kerberos database, use the following command:

```
kdb_util load /var/kerberos/database/dump_<datestamp>
```

```
rcmd sp5cw0 255 1 1 0 be6eca67 512ec419 203801010459 199809222331 root
admin
hardmon sp5sw01 255 1 1 0 21713140 9d6581a5 203801010459 199810112132
root admin
K M 255 1 1 0 8f83b333 81c5ef6d 203801010459 199809222331 db_creation *
rcmd sp5n09 255 1 1 0 478d20ad f1df2cf 203801010459 199809231405 root
admin
changepw kerberos 255 1 1 0 fe779a0f 44e8309b 203801010459 199809222331
db_creation *
rcmd sp5sw01 255 1 1 0 7dff1d19 8cfa497b 203801010459 199809231405 root
admin
krbtgt MSC.ITSO.IBM.COM 255 1 1 0 687fb4dc b68e6dbf 203801010459
199809222331 db_creation *
rcmd sp5en0 255 1 1 0 69d82099 5f353db3 203801010459 199809222331 root
admin
default * 255 1 1 0 0 0 203801010459 199809222331 db_creation *
hardmon sp5cw0 255 1 1 0 35e807f6 c4233725 203801010459 199809222331
root admin
```

Figure 109. Sample of Kerberos database dump

3.2.3 Recreating authentication key file using ext_srvtab

We know that `setup_authent` creates the /etc/krb-srvtab file on the CWS, and for nodes, `setup_server` creates the authentication key files (see A.6, “create_krb_files” on page 337) and saves them in the /tftpboot directory as <node name>-new-srvtab. But, it is also possible to create these autentication key files by using the `ext_srvtab` command on the Kerberos authentication server.

The following example shows how to create an authentication key file for a node on the CWS. In this example, the CWS is the Kerberos authentication server.

On the Kerberos authentication server (in our case, the CWS), we run `ext_srvtab` (located in /usr/lpp/ssp/kerberos/etc) to re-create the authentication key for node sp5n09 as follows:

```
cd /tmp
ext_srvtab -n sp5n09 sp5sw09
```

In this example, sp5n09 is the hostname for the internal Ethernet interface (en0), and sp5sw09 is the host name for the SP Switch interface (css0). Two files named sp5n09-new-srvtab and sp5sw09-new-srvtab are created in the /tmp directory on the CWS. We combine the two files as follows:

```
cat sp5n09-new-srvtab and sp5sw09-new-srvtab >/tmp/new-srvtab
```

Now, we transfer (using FTP) the new-srvtab file to sp5n09 node as /etc/krb-srvtab.

The first problem determination checkpoint for the authentication key is to read the service key from the /etc/krb-srvtab file and then check if the instances have the correct hostname by using the `klist -srvtab` command. The second checkpoint is to check if the key version is same as that of the Kerberos database. If those two items of information are not correct, we need to create the authentication key manually or by using the create_krb_files wrapper.

3.3 Kerberos problem determination procedure

Here, we describe potential problems with Kerberos and also discuss the methods you can use to solve them. If there is a Kerberos problem, it can be detected from the error messages. The error messages give an indication of the type of Kerberos failure. We lists the general type of failures and the recovery procedures in Table 4 on page 179.

Kerberos can be the root cause for many other problems. For example, if Kerberos does not work properly, `setup_server` fails on the boot/install server. The switch commands, such as `Estart` or `Eunfence`, use Kerberos; therefore, a Kerberos problem can also lead to a switch problem. Different SP-specific services, such as hardmon, use Kerberos and will not work if the Kerberos authentication is not correct. The best way to determine if Kerberos is causing a problem is to analyze the error messages. Our approach is to determine the nature of the failure and then do a cross reference with Table 4 to find out the recovery actions.

Table 4. Kerberos problems

Problem symptom	**Recovery action**

Problem establishing a user’s principal identity.
 	
An example of bad Kerberos name format:
 	
`sp5en0{/} k4init`
`Kerberos Initialization`
`Kerberos name: root.admin`
`k4init: 2502-003 Bad Kerberos name format`
 	
An example of failure messages for missing root.admin principal in /.klogin file on the Control Workstation:
 	
`sp5n01{/} dsh -w sp5en0 date`
`sp5en0:krshd:Kerberos Authentication Failed:User root.admin@MSC.ITSO.IBM.COM is not authorized to login to account root.`
`sp5en0: spk4rsh: 0041-004 Kerberos rcmd failed: rcmd protocol failure.`
	The probable causes are:
	• Bad kerberos name format
	• Kerberos principal does not exist incorrect Kerberos password
	• Corrupted Kerberos database
	The error message of the failing command indicates bad Kerberos name format, or unknown Kerberos principal, or incorrect password. The recovery action is to repeat the command with correct information.
	
	Check the /.klogin file to see if it has an entry for this principal.
	
	If all the information is correct, but the Kerberos command fails, suspect a database corruption. See 3.3.1, “Re-creating the Kerberos database” on page 181 for information about how to re-create the Kerberos database.

Problem symptom	Recovery action

Problem establishing a service's principal identity.
 	
An example of messages received when the /etc/krb-srvtab file is corrupted and the remote command service (rcmd) fails to work from the CWS:
 	
`sp5en0{/} dsh -w sp5n01 date`
`sp5n01:krshd:Kerberos Authentication Failed.`
`sp5n01: spk4rsh: 0041-004 Kerberos rcmd failed: rcmd protocol failure.`
	The probable causes for this problem are:
	•krb-srvtab file does not exist on the node or on the CWS
	•The krb-srvtab has the wrong key version The krb-srvtab file is corrupted
	Analyze the error messages to confirm the services's principal identity problem.
	
	See 3.2.1, "Comparing the service key version using k4list" on page 176 for information about how to check the key version.
	
	Make sure /.klogin, /etc/krb.realms and /etc/krb-conf files are consistent with those of the Kerberos authentication server.
	
	See 3.2.3, "Recreating authentication key file using ext_srvtab" on page 177 to re-create the authentication key.
	
	If the problem persists re-create the Kerberos database; see 3.3.1, "Re-creating the Kerberos database" on page 181.

Problems with authenticated services.
 	
An example of an authenticated service is hardmon. An example of messages received while hardmon is having problems due to Kerberos errors is as follows:
 	
`sp5en0{/} spmon -d`
 	
`Opening connection to server`
`0026-706 Cannot obtain service ticket for hardmon.sp5en0`
`Kerberos error code is 8, Kerberos error message is:`
`2504-008 Kerberos principal unknown`
	The probable causes are: Ticket has expired, valid ticket does not exist, host name resolution is not correct, or ACL files do not have correct entries.
	
	Destroy the ticket using `k4destroy` and issue a new ticket by using the `k4init` command.
	
	Check the hostname resolution.
	
	Check the ACL files.
	
	Check the Kerberos database.

Problem symptom	Recovery action

Kerberos database corruption.
 	
The database can be corrupted for many reasons, and messages also vary based on the nature of the corruption. Here we provide an example of a message received because of Kerberos database corruption:
 	
`sp5en0{/} k4init root.admin`
`Kerberos Initialization for "root.admin"`
`k4init: 2504-010 Kerberos principal has null key`
	Re-create the database; see 3.3.1, "Re-creating the Kerberos database" on page 181 for the steps describing how to re-create the database.

Problems with the Kerberos daemon.
 	
Here, we provide an example of a message when Kerberos daemons are inactive because of a missing krb.realms file on the CWS. This message is an excerpt from the admin_server.syslog file:
 	
`26-Oct-98 17:47:43 Shutting down admin server`
`26-Oct-98 17:48:15 kadmind: 2503-001 Could not get local realm.`
	Check that all the kerberos files exist on the autentication server, which is usually the CWS. See 3.2.1, "Comparing the service key version using k4list" on page 176 for the list of Kerberos files and directories on the authentication server. Check the contents of the files to make sure the files are not corrupted.
	
	Check the /var/adm/SPlogs/kerberos log directory for messages related to Kerberos daemons.

3.3.1 Re-creating the Kerberos database

The following procedure outlines how to destroy the Kerberos database on the SP and then rebuild it. At the CWS, log in as root and execute the following commands:

```
/usr/lpp/ssp/kerberos/bin/kdestroy
```

The `kdestroy` command destroys the user's authentication tickets, which are located in /tmp/tkt<uid> by default:

```
/usr/lpp/ssp/kerberos/etc/kdb_destroy
```

The `kdb_destroy` command destroys the Kerberos authentication database, which is located in the /var/kerberos/database:

```
rm /etc/krb*
```

This removes the following files:

- krb-srvtab, which contains the keys for services on the nodes
- krb.conf, which contains the SP authentication configuration
- krb.realms, which specifies the translations from host names to authentication realms

```
rm /.klogin
```

This removes the .klogin file, which contains a list of principals that are authorized to invoke processes as the root user with the SP authenticated remote commands (`rsh`,`rcp`).

```
rm /.k
```

This removes the Kerberos master key cache file.

```
rm /var/kerberos/database/*
```

This command insures that the authentication database files are completely removed.

```
   /usr/lpp/ssp/bin/setup_authent
```

This command configures SP authentication services. Executing this command invokes an interactive dialog in which various utility programs are invoked to accomplish this configuration.

The final step involves propagating the /etc/krb-srvtab files onto the nodes. This can be done automatically or manually. Here, we describe how to propagate the Kerberos database manually. For automatic transfer, customize the nodes.

On the CWS, run the following command to set all the nodes to customize mode:

```
spbootins -r customize -s no -l <node list>
```

then run the create_krb_files wrapper on the CWS and verify that there is a <node_name>-new-srvtab file for each node set to customize in the /tftpboot directory.

Transfer each node's respective /tftpboot/<node-name>-new-srvtab file from the CWS to the node and rename the file to /etc/krb-srvtab.

Set the nodes back to disk using the following command:

```
spbootins -r disk -s no -l <node list>
```

When using FTP, make sure to do the transfer in binary mode. Once the nodes are customized with the new /etc/krb-srvtab, then you can test the functionality of Kerberos by obtaining a ticket (`k4init root.admin`) and executing the `rsh <any_node> date` command.

3.4 Case studies

Here, we describe two case studies related to Kerberos. The first case study is an example of fixing the kerberos database, and the second one shows the Kerberos problem due to improper hostname resolution.

3.4.1 Fixing wrong information in the Kerberos database

This case study shows how to determine if wrong information is in the Kerberos database and how to repair the database instead of re-creating it from scratch.

Problem symptom

In our SP environment, we have one boot install server (node 1). The bootinstall server has two Ethernet interfaces and an SP Switch interface. Assuming that the CWS crashed, we rebuild the CWS from an old mksysb. In the testing phase, while working with the `dsh` command, we encountered a problem with the boot/install server. We try the `dsh` command as follows:

```
dsh -w < en0 interface name> date
dsh -w <en1 interface name> date
dsh -w <css0 interface name> date
```

The `dsh` command for the en1 interface failed with the messages as shown in Figure 110 on page 183.

```
sp5en0{/tftpboot} dsh -w sp5nb01 date
sp5nb01: Thu Oct 22 16:23:37 EDT 1998
sp5nb01: krshd: Kerberos Authentication Failed.
sp5nb01: spk4rsh: 0041-004 Kerberos rcmd failed: rcmd protocol failure.
```

Figure 110. Kerberos failing message—Part one

As part of the recovery, the host name resolution was checked on the nodes and on the CWS. It was found that the long hostname for the nodes had an incorrect entry. This error was corrected, but that still did not solve our problem.

Problem determination

To solve the problem, we need to analyze the message first. In Figure 110 on page 183, we see that the prompt returns the date and then complains about `krshd`. The first line appears because we set the standard AIX and Kerberos Version 4 authentication method for nodes; so, the standard `rsh` works, and it returns the date of the node. So, we can discard the first message.

The second message indicates Kerberos authentication failure. But the message does not clearly indicate if it is service or user principal authentication failure. We need to use the process of elimination to find out which type of authentication is failing. To do this, we first consider user principal authentication failure, as shown in Table 4 on page 179, and follow the recovery actions. We destroy the ticket on the CWS using the `k4destory` command and reissue the ticket using the `k4init` command. This time, we receive a different message as shown in Figure 111:

```
sp5en0{/tftpboot} dsh -w sp5nb01 date
sp5nb01: Thu Oct 22 17:25:39 EDT 1998
sp5nb01: spk4rsh: 0041-002 Host sp5nb01.msc.itso.ibm.com is not
registered for Kerberos service.
```

Figure 111. Kerberos failing message—Part two

This time message indicates that hostname sp5nb01 is not registered to the Kerberos database. The message implies that, for some reason, host name sp5nb01 is not known to the database. Before checking the database itself, we check the service key on the node to see what host names are available in the krb-srvtab file on the node. To do this, we run the command `k4list -srvtab` on the node. The output is shown in Figure 112 on page 185.

In this figure, we see that the instance name is sp5n01b, not sp5nb01. The krb-srvtab file is created based on the information available in the database. So, for some reason, in the database, the node hostname appeared as sp5n01b.

```
# k4list -srvtab
Server key file:   /etc/krb-srvtab
Service         Instance        Realm      Key Version
------------------------------------------------------
rcmd            sp5n01b         MSC.ITSO.IBM.COM 1
rcmd            sp5sw01         MSC.ITSO.IBM.COM 1
rcmd            sp5n01          MSC.ITSO.IBM.COM 1
```

Figure 112. Content of krb-srvtab file on the node

Now, we take a dump of the database as follows:

```
kdb_util dump /tmp/kerby.dump
```

Figure 113 shows the result:

```
cmd sp5cw0 255 1 1 0 103f9317 b0c7670f 203801010459 199810202119 root
adminK M 255 1 1 0 8f83b333 81c5ef6d 203801010459 199810202118
db_creation *
rcmd sp5n09 255 1 1 0 5253c118 dfa4778f 203801010459 199810202120 root
admin
rcmd spn01b 255 1 1 0 4b7b1470 1ec3e564 203801010459 199810202120 root
admin
changepw kerberos 255 1 1 0 6567c352 690ed259 203801010459 199810202118
db_creation *
```

Figure 113. Wrong information shown in the dump file

We can simply edit the dump file using an editor and change the hostname from sp5n01b to the actual hostname sp5nb01. Then, we reload the database as follows:

```
kdb_util load /tmp/kerby.dump
```

or we list its contents by using the `lskp` command.

However, fixing only the Kerberos database is not enough to solve the problem; we also have to generate the krb-srvtab file for node 1. To do this, we set the node to customize and run the `create_krb_files` wrapper on the CWS. The wrapper creates the krb-srvtab file as <node name>-new-srvtab file in the /tftpboot directory (see Figure 114).

Now, we need to transfer the <node name>-new-srvtab file into the /etc directory on the node as krb-srvtab file. The transfer must be done in binary mode see (Figure 114).

```
sp5en0{/tftpboot} ls -l sp5n01*
-r--------   1 nobody   system       116 Oct 22 16:15 sp5n01-new-srvtab
-rw-r--r--   1 root     system       230 Oct 22 13:27
sp5n01.msc.itso.ibm.com.config_info
-rw-r--r--   1 root     system       777 Oct 22 13:27
sp5n01.msc.itso.ibm.com.install_info
```

Figure 114. The /tftpboot directory after running create_krb_files

Conclusion

While rebuilding the CWS, the /etc/hosts file had the wrong long hostname for the second interface of node 1. This caused the setup_authent script to build the Kerberos database with incorrect information. Therefore, care must be used while building the /etc/hosts file and while building the DNS server information.

3.4.2 Estart does not work because of Kerberos problems

Kerberos is simple, but if it is not working properly, it can cause numerous problems in an SP environment. This case study is an unique example of the previous statement.

Problem symptom

After running the `Eprimary` command to change the oncoming primary and oncoming primary backup nodes, the `Estart` command fails with a message about Kerberos. The failing message is shown in Figure 115 on page 187. The message indicates that Kerberos cannot match the host address, but it does not say which host address.

```
sp5en0{/etc} Estart
 Estart: Oncoming primary != primary, Estart directed to oncoming
primary
Estart:  0028-061 Estart is being issued to the primary node:
sp5n05.msc.itso.ibm.com.
rshd: 0826-825 There is a host address that does not match.spk4rsh:
0041-004 Kerberos rcmd failed: rcmd protocol failure.
rshd: 0826-825 There is a host address that does not match.>>> The
return code from the command was 1.
/usr/lpp/ssp/css/Estart_sw[364]:
/u/dawn/trout/obj/power/ssp/css/msg/spmsg_css:  not found.
sp5en0{/etc}
```

Figure 115. Estart failing message

Problem determination

To solve this problem, we need to know which host address Kerberos is complaining about. Since the problem starts after running the `Eprimary` command, we run the command to list the hosts involved in the switch operation. Figure 116 shows that the oncoming primary is node 5, and the primary backup is node 1. We can make an educated guess that Kerberos is complaining about node 5.

```
sp5en0{/etc} Eprimary
1       - primary
5       - oncoming primary
5       - primary backup
1       - oncoming primary backup
```

Figure 116. Eprimary output

To make sure that Kerberos is complaining about node 5, we turn on the debug mode in the syslog.conf file. The procedure for turning on debug is very simple. First, we need to add an entry in the syslog.conf file as shown in Figure 117 on page 188. We then touch the file debug.info in the /var/adm directory. Then, we stop and start the syslogd daemon using the `startsrc` command.

```
#       emerg/panic,alert,crit,err(or),warn(ing),notice,info,debug
#       (meaning all messages of this priority or higher)
#
# <destination> is:
#       /filename - log to this file
#       username[,username2...] - write to user(s)
#       @hostname - send to syslogd on this machine
#       * - send to all logged in users
#
# example:
# "mail messages, at debug or higher, go to Log file. File must exist."
# "all facilities, at debug and higher, go to console"
# "all facilities, at crit or higher, go to all users"
#  mail.debug           /usr/spool/mqueue/syslog
#  *.debug              /dev/console
#  *.crit                       *
*.debug         /var/adm/debug.info
```

Figure 117. Bottom portion of syslog.conf: Debugging turned on

After turning debugging on in the syslog.conf file, we run the `Estart` command again. Now, we check the /var/adm/debug.info file to see if we have received any information. There are entries about the node 5 address as shown in Figure 118. This confirms our guess that Kerberos is complaining about node 5.

```
Oct 22 19:14:53 sp5en0 rshd[38174]: rshd: 0826-824 Host address
c0a8:605::1 is not listed for host sp5n05.msc.itso.ibm.com.
Oct 22 19:14:59 sp5en0 rshd[38176]: rshd: 0826-824 Host address
::ffff:192.168.6.5 is not listed for host sp5n05.msc.itso.ibm.com.
```

Figure 118. Information from the debug.info file

Now, we check the hostname resolution for node 5 on the CWS using the `host` command as shown in Figure 119 on page 189. The host name resolution works correctly.

```
 sp5en0{/} host sp5n05
sp5n05.msc.itso.ibm.com is 192.168.6.5,  Aliases:   sp5n05
sp5en0{/} host 192.168.6.5
sp5n05.msc.itso.ibm.com is 192.168.6.5,  Aliases:   sp5n05
```

Figure 119. Checking the hostname resolution on CWS

Now, we check the IP address for node 5 in the SDR as follows:

```
SDRGetObjects Adapter node_number==5 netaddr
```

The output of this command is shown in Figure 120. We find that the IP address is the same as that of the /etc/hosts file.

```
sp5en0{/} SDRGetObjects Adapter node_number==5 netaddr
netaddr
192.168.6.5
192.168.15.5
```

Figure 120. SDR information for node 5 adapter IP address

The next question is: Are we using some other mechanism for hostname resolution, like DNS? We check to see if there is a /etc/resolv.conf and /etc/netsvc.conf file on the CWS. We find that there is a resolv.conf file; therefore, the DNS server must be complaining about node 5. We can query the DNS server as follows:

```
nslookup sp5n05
```

The output of the `nslookup` command is shown in Figure 121 on page 189. The IP address returned from the DNS server is totally different from the one we have in the SDR and in the /etc/hosts file. We know that our SDR and /etc/hosts file entries match the node’s IP address. So, to solve the problem, we can either turn off DNS resolution or correct the DNS server information.

```
sp5en0{/} nslookup sp5n05
Server:  riscserver.msc.itso.ibm.com
Address:  9.12.1.30
Non-authoritative answer:
Name:    sp5n05.msc.itso.ibm.com
Address:  192.168.5.5
```

Figure 121. Output of the nslookup command

The quick workaround for this problem is to set the NSORDER variable in the /etc/environment variable or to create the /etc/netsvc.conf file. We can add the following entry to the /etc/netsvc.conf file:

```
hosts=local,bind
```

Now, we issue an `Estart` on the CWS, and it works successfully.

We could also have changed the /etc/environment file as shown in Figure 122:

```
# ODM routines use ODMDIR to determine which objects to operate on
# the default is /etc/objrepos - this is where the device objects
# reside, which are required for hardware configuration

ODMDIR=/etc/objrepos
NSORDER=local,bind
```

Figure 122. NSORDER Variable is Set in /etc/environment

Conclusion

Hostname resolution is one of the main reasons for Kerberos problems. In this case study, the problem, at first, looked like a switch problem, but underneath `Estart`, actually Kerberos failed because of hostname resolution.

Chapter 4. RS/6000 Cluster Technology (RSCT)

The name RS/6000 Cluster Technology (RSCT) was introduced in PSSP 3.1 as part of the cluster initiative within IBM. This technology has been available in RS/6000 SP platforms for several years and now has been made available to the entire RS/6000 family.

To understand the impact that this technology will have in the way how we do business, we need to understand how this technology delivers high availability, online monitoring and automatic recovery actions to a cluster of RS/6000 machines.

The key is distributed subsystems. All the components of this technology are distributed across multiples machines or nodes. These subsystems communicate will each other through multiple networks in order to eliminate the single point of failure of a single network and to deliver a set of services to applications or upper layers for synchronization and sharing of data and state information.

Figure 123 shows a graphical representation of the RS/6000 Cluster Technology. Although this figure shows an additional RSCT stack (set of daemon running in a domain) for High Availability Cluster Multiprocessing (HACMP) environment as well, we will concentrate on the SP domain (daemons running within an RS/6000 SP system partition).

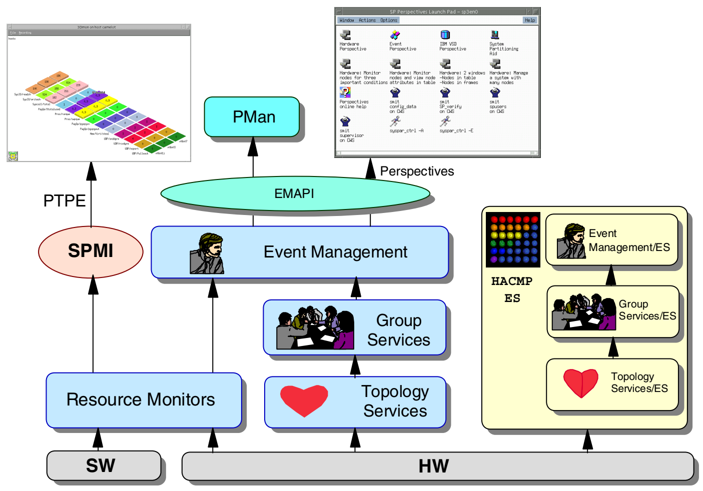


Figure 123. RS/6000 Cluster Technology overview

Problem determination for the RSCT stack running in HACMP environments is somehow similar to SP environments, and many of the concepts and procedures you will see here may apply to HACMP environments. However, this chapter is not intended for such environments; so, you may refer to the HACMP official documentation and to the *HACMP Enhanced Scalability Handbook*, SG24-5328, for more detailed information on HACMP environments.

4.1 Topology Services

Topology Services (TS) is a distributed subsystem. It is the lowest layer in the RS/6000 Cluster Technology (RSCT), and it maintains availability information regarding nodes and network adapters (IP adapters). TS considers a node up if it can be reached through any of its adapters following any of the available paths.

Topology Services is implemented as a daemon (hatsd). All nodes in the domain (system partition), including the Control Workstation (CWS), run this daemon and share a common set of definitions. The CWS may have multiple hats daemons running if there are more than one partition. Each hats daemon on the CWS will communicate and synchronize with the other hats daemons running in the same system partition.

4.1.1 Topology Services concepts

TS has to provide availability and connectivity information to the upper layers (discussed later on in the chapter). However, TS is a distributed subsystem; so, the information may be generated anywhere in the domain or system partition, but it has to be the same no matter where TS clients get it from.

Moreover, TS has to be able to maintain availability and connectivity information for several hundred nodes (SP systems may go from a couple to several hundred nodes); so, traffic over the network should be maintained as low as possible. To accomplish this, the TS design considers that all the daemons running in the domain or system partition will communicate with each other in an ordered fashion (ring fashion). There will be a *Leader (Group Leader or GL)* who will maintain this information per network in the case of nodes with multiple adapters.

Let us assume that we have the configuration depicted in Figure 124 on page 193. In this configuration, there are four nodes connected to two networks: SP Ethernet and SP Switch. The CWS is only connected to the SP Ethernet network. The IP addresses are known by all the nodes because they are stored in the SDR.

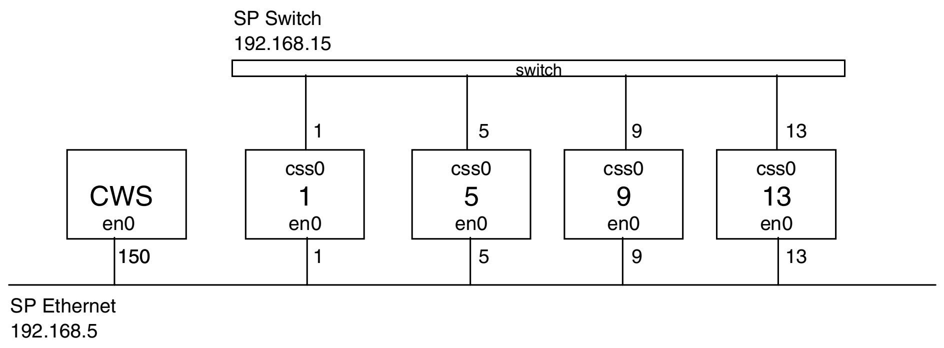


Figure 124. Topology Services—Simple SP configuration

In order to minimize the traffic over the network being monitored, TS creates logical rings. There will be one ring per network and one Group Leader (GL) per ring. The selection of the GL is based on the IP address. The highest IP address will be the GL for that ring. Notice that the GL node is not necessary the same for all the rings; it depends exclusively on IP addresses. In the example above, the GL leader for the SP Ethernet ring will be the CWS (just because it has the highest IP address in this network), and for the SP Switch, it will be node 13.

The Topology Services daemon will try to form rings as large as possible. To form these rings, daemons must alert the other daemons on the other nodes of its presence using a PROCLAIM message. According to the hierarchy defined by Topology Services, a daemon may only proclaim to IP addresses that are lower than its own, and a daemon may only accept a proclaim message from an IP address higher than its own. Also, a daemon only proclaims if it is the leader of a ring. When a daemon first starts up, it builds a heartbeat ring for every local adapter containing only that local adapter. This is called a Singleton group, and this daemon is the leader in each one of these singleton groups.

To manage the changes in these groups, Topology Services defines the following roles:

Group Leader: The daemon whose local adapter in this group has the highest IP address of all its group members. The Group Leader proclaims, handles JOIN requests, handles DEATH notifications, coordinates with group members any group membership changes, and sends around connectivity information.

Crown Prince: This is the daemon whose local adapter in this group has the second highest IP address of all its group members. It is also the daemon that detects the death of the Group Leader and has the authority to takeover the leadership of the group if that happens.

Mayor: This is a daemon, with a local adapter present in this group, that has been picked by the Group Leader to broadcast a message to all the adapters in the group that are also members of its subnet. The way a daemon knows that it has been picked is when it receives a message that has to be broadcasted.

Generic: This is any daemon with a local adapter in the heartbeat ring. The role of the Generic daemon is to monitor the heartbeat of the upstream neighbor (actually, it expects to receive a heartbeat from the upstream neighbor every few seconds) and inform the Group Leader if there is something wrong with it.

Once the daemons have determined who will be the leader, they start heartbeating in a ring fashion following the IP addresses. Each daemon will heartbeat to the next lower IP address in the ring, and the lowest IP address in the ring will heartbeat to the highest or GL closing the ring. This happens for every ring; so, hats daemons will, most of the time, heartbeat in multiple rings.

The way of each daemon obtains the information about the IP addresses and nodes in the domain or system partition is part of what is called initialization.

4.1.1.1 Topology Services initialization

TS is managed through a System Resource Controller (SRC) subsystem called hats. *<syspar_name>* is used only on the CWS to identify the right daemon since more than one may be running. Every time a node starts up, it will start the hats SRC subsystem automatically from the /etc/inittab list. The entry in the inittab list is similar to this:

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

You may also stop and restart the subsystem manually by using the `stopsrc` and `startsrc` commands or the Topology Services control `hatsctrl` command.

When the SRC subsystem starts, it calls the hats script (/usr/sbin/rsct/bin/hats) as you can see from the SRC subsystem definition:

```
sp5en0:/>odmget -q subsysname=hats.sp5en0 SRCsubsys

SRCsubsys:
        subsysname = "hats.sp5en0"
        synonym = ""
        cmdargs = "192.168.5.150"
        path = "/usr/sbin/rsct/bin/hats"
        uid = 0
        auditid = 0
        standin = "/dev/console"
        standout = "/var/ha/log/hats.sp5en0"
        standerr = "/var/ha/log/hats.sp5en0"
        action = 1
        multi = 0
        contact = 3
        svrkey = 0
        svrmtype = 0
        priority = 20
        signorm = 0
        sigforce = 0
        display = 1
        waittime = 30
        grpname = "hats"
```

This script prepares the configuration files needed by the hats daemon and then starts the daemon. There two configuration files that define how the hats daemon will behave. The configuration files are generated at the CWS and stored in the SDR. The hats script running on the nodes will later copy these files from the CWS to the /var/ha/run directory on the node. However, if the configuration files get obsolete because of changes in the SDR information (adapters or IP addresses), the script will replace those configuration files in the SDR.

Since TS is a partition-sensitive subsystem, the configuration files contain information about the nodes and adapters within the partition plus the CWS that is part of all partitions. The files can be retrieved for verification as follows:

```
[sp5en0:/]# SDRRetrieveFile hats.machines.lst /tmp/machines.inst
[sp5en0:/]# SDRRetrieveFile hats.machines.inst /tmp/machines.inst
```

These two commands will copy the machines.lst and machines.inst files (configuration files) to the /tmp directory.

In our sample configuration (shown in Figure 124 on page 193), the machines.lst file would look like the one shown in Figure 125 on page 196.

```
*InstanceNumber=917277434
*configId=2225957613
*FileVersion=1
*!TS_realm=PSSP
*!TS_PinText
TS_Frequency=1
TS_Sensitivity=4
TS_FixedPriority=38
TS_LogLength=5000
Network Name SPether
Network Type ether
*
*Node Type Address
    0 en0 192.168.5.150
    1 en0  192.168.5.1
    5 en0  192.168.5.5
    9 en0  192.168.5.9
   13 en0  192.168.5.13
Network Name SPswitch
Network Type hps
*
*Node Type Address
    1 css0 192.168.15.1
    5 css0 192.168.15.5
    9 css0 192.168.15.9
   13 css0 192.168.15.13
```

Figure 125. Topology Services—machines.lst file

The machines.inst files contains only one line, which is the first line of the machines.lst file.

As you can see in Figure 125 on page 196, the machines.lst file contains all the IP addresses for all the supported networks. TS supports only the SP Ethernet and the SP Switch when running on an SP domain. If TS is running on an HACMP environment, it will support other IP networks, such as Token Ring, FDDI, ATM, and others.

When the SRC subsystem is started, it invokes the /usr/sbin/rsct/bin/hats script. If the script is run on the CWS, it will create a machines.lst file with the information it obtains from the SDR. Then, it compares it with the machines.lst file stored in the SDR. If they do not match, it will replace the file in the SDR with this new file. The configID value that appears in the second line of the machines.lst file corresponds to a check sum of the file.

If the script is run on a node, it will copy the machines.lst file from the CWS to the /var/ha/run directory. The daemon then decides whether a daemon refresh is needed.

Once the script verifies that the machines.lst file is current, it will copy the file to the run-time location (/var/ha/run/hats.<syspar_name>/machines.lst), and then it will start the hats daemon (/usr/sbin/rsct/bin/hatsd).

The hats daemon will read this machines.lst file, and it will determine the highest IP address in each network (or ring), and then it will wait for a PROCLAIM message from the Group Leader. When it receives it, it sends a JOIN request back to the Group Leader. JOIN messages are only sent by a node that receives a PROCLAIM message.

Assuming that the Group Leader (hats daemon running on the node with the highest IP address for a particular network) is up and it receives the JOIN request from the rest of the nodes listed in the machines.lst file, it will initialize Topology Services by sending a packet call Prepare To Commit (PTC), which contains a list (or membership list) of all the nodes "alive" in that ring. This membership looks like the following:

```
01/25  10:19:33 hatsd[0]: Sending PTC to new membership:
        0 (192.168.5.1:0x46ac8b28)      1 (192.168.5.5:0x46ac8b4b)
        2 (192.168.5.9:0x46ac8b66)      3 (192.168.5.13:0x46ac8b80)
        4 (192.168.5.150:0x46ac8afe)
```

This is an excerpt from the hats daemon’s log file. The PTC packet contains a list of all nodes in the ring. This list is distributed by the Group Leader to all members of the ring. Each member will then start heartbeating, following the IP addresses, according to this list and not to the machines.lst file.

From here, you can see that the machines.lst file is used for initialization only, and after the daemons have started and communicated with each other, the file is not longer used, and a run-time version of it is used instead.

The only one capable of changing the membership information is the Group Leader for a particular ring (network).

4.1.2 Topology Service logs

Topology Services logs error and informational messages to the AIX Error log and two log files located in the /var/ha/log directory. TS uses mostly its own log files rather than using the AIX Error log facility. The entries in the AIX Error log for Topology Services are usually from the SRC master process, and they are related to starting or stopping the SRC subsystem.

The two log files stored in the /var/ha/log directory correspond to the hats script and daemon. Let us first list this directory and look for the hats related files:

```
sp5en0:/var/ha/log>ls -l hats*
-rw-rw-rw-   1 root     system     17436 Jan 25 10:19 hats.25.101716.sp5en0
-rwxr-xr-x   1 root     system        90 Jan 25 10:17 hats.sp5en0
```

From the previous output, you can see the two log files for hats. The file named hats.sp5en0 corresponds to the standard output and standard error from the /usr/sbin/rsct/bin/hats script, the one started by the SRC process. The *sp5en0* extension is the partition name. If you have more than one system partition, you will see several hats.<syspar_name> files, one per partition if you list this directory on the CWS. Any error or message from the hats script will be logged into this file.

The other file in the listing is the daemon's log file (hatsd daemon). This log is updated regularly by the daemon. This is the file you want to check out if there is any problem with Topology Services. The syntax for the name is as follow:

```
   hats.<DoM>.<Time>.<syspar_name> where       DoM is Day of Month
                                               Time (hhmmss)
```

This file is created by the hats daemon; so, if the hats script fails for some reason before starting the daemon, you will not see this file, and the reason for the failure should be in the hats.<syspar_name> log file.

To analyze the daemon's log file, you have to be familiar with the activities that the daemon carries out while running. As we mentioned earlier in the section, the concept behind Topology Services is to provide availability and connectivity information. This information is know by the hats daemon as a membership lists or the lists of adapters active on different rings (networks).

To establish this membership information, there must be a Group Leader so that each daemon will seek out its Group Leader for each ring it is on.

Let us analyze a real log file taken from a running system. Assuming that the system described in Figure 124 on page 193 is up an running (all nodes are active), the Group Leader for the SP Ethernet should be the hats daemon running on the CWS, while the Group Leader for the SP Switch network should be the hats daemon running on node 13 (sp5n13), which has the highest IP address on the switch network.

Let us analyze the log file for the Group Leader of the SP Ethernet network (the CWS). Since this file is rather large and always increasing in size (there is a trimming value set to 5000 lines by default), let us take parts and analyze them separately[1]:

Initialization: Reading the machines.lst file

```
+1  01/25  10:17:16 hatsd[0]: High Availability Topology Services Daemon.
+2  01/25  10:17:16 hatsd[0]: Using socket Port number = 10000.
+3  01/25  10:17:16 hatsd[0]: Using server socket = /var/ha/soc/hats/server_socket.sp5en0.
+4  01/25  10:17:16 hatsd[0]: The background flag = 0
+5  01/25  10:17:16 hatsd[0]: System Configuration File:
/var/ha/run/hats.sp5en0/machines.lst.
+6  01/25  10:17:17 hatsd[0]: Network SPether.
+7  01/25  10:17:17 hatsd[0]: Node 0 adapter 0 Interface en0 Address 192.168.5.150.
+8  01/25  10:17:17 hatsd[0]: Node 1 adapter 0 Interface en0 Address 192.168.5.1.
+9  01/25  10:17:17 hatsd[0]: Node 5 adapter 0 Interface en0 Address 192.168.5.5.
+10  01/25  10:17:17 hatsd[0]: Node 9 adapter 0 Interface en0 Address 192.168.5.9.
+11  01/25  10:17:17 hatsd[0]: Node 13 adapter 0 Interface en0 Address 192.168.5.13.
+12  01/25  10:17:17 hatsd[1]: Network SPswitch.
+13  01/25  10:17:17 hatsd[1]: Node 1 adapter 1 Interface css0 Address 192.168.15.1.
+14  01/25  10:17:17 hatsd[1]: Node 5 adapter 1 Interface css0 Address 192.168.15.5.
+15  01/25  10:17:17 hatsd[1]: Node 9 adapter 1 Interface css0 Address 192.168.15.9.
+16  01/25  10:17:17 hatsd[1]: Node 13 adapter 1 Interface css0 Address 192.168.15.13.
+17  01/25  10:17:17 hatsd[0]: Adapter[0] Name = 192.168.5.150, Address = 192.168.5.150.
```

As you can see from the file, the daemon initializes by opening two sockets: One for internode communication (UDP port 10000 by default) and one for intranode communication (UNIX Domain Socket located at /var/ha/soc/server_socket.<syspar_name>).

Line 5: The daemon open the machines.lst file and list the IP addresses for all the nodes in all the networks (Lines 6 to 17).

Initialization: Setting up tunables

```
    +18  01/25  10:17:17 - (0) - incorporate_tunables()    LOCKING 2
    +19  01/25  10:17:17 hatsd[0]: Timer intervals used by offset 0 *****************
    +20  01/25  10:17:17 hatsd[0]: HEARTBEAT Send Interval        = (1.000000) seconds.
    +21  01/25  10:17:17 hatsd[0]: HEARTBEAT Receive Interval     = (2.000000) seconds.
    +22  01/25  10:17:17 hatsd[0]: Group Stability Interval       = (10.000000) seconds.
    +23  01/25  10:17:17 hatsd[0]: If Unstable Set NCT Interval   = (40.000000) seconds.
    +24  01/25  10:17:17 hatsd[0]: JOIN Interval                  = (10.000000) seconds.
    +25  01/25  10:17:17 hatsd[0]: Batch Join Interval            = (2.000000) seconds.
    +26  01/25  10:17:17 hatsd[0]: PTC_ACK Interval               = (2.000000) seconds.
    +27  01/25  10:17:17 hatsd[0]: COMMIT_ACK Interval            = (2.000000) seconds.
    +28  01/25  10:17:17 hatsd[0]: COMMIT_BROADCAST_ACK Interval  = (2.000000) seconds.
    +29  01/25  10:17:17 hatsd[0]: COMMIT Interval                = (5.000000) seconds.
    +30  01/25  10:17:17 hatsd[0]: PPROCLAIM Interval             = (5.000000) seconds.
    +31  01/25  10:17:17 hatsd[0]: NODE_CONNECTIVITY Interval     = (10.000000) seconds.
    +32  01/25  10:17:17 hatsd[0]: GROUP_CONNECTIVITY Interval    = (20.000000) seconds.
    +33  01/25  10:17:17 hatsd[0]: Default Wait Interval          = (10.000000) seconds.
    +34  01/25  10:17:17 hatsd[0]: Fibrillate Count               = 4
    +35  01/25  10:17:17 hatsd[0]: MAX_PTC_RETRY 3. MAX_COMMIT_RETRY 3. MAX_ACK_RETRY 3
    +36  01/25  10:17:17 - (0) - incorporate_configuration() IPAT disabled on offset 0
```

Lines 18 to 35 correspond to tunable settings. Some of those tunables come from the TS_Config SDR classes. The content of this SDR class is as follow:

[1] The line numbers are not part of the file. They have been added for easier identification.

```
sp5en0:/>SDRGetObjects TS_Config
Frequency   Sensitivity Run_FixPri  FixPri_Value Log_Length  Pinning
         1          4          1          38        5000 ""
```

This class is used by Topology Services to specify some of the tunables you see in the log file. Their description is as follow:

- **Frequency**: Number of seconds between each packet sent (heartbeat rate).
- **Sensitivity**: Number of packets (heartbeat) lost before considering the upstream neighbor down.
- **Run_FixPri**: This tunable is the AIX priority value.
- **FixPri**: Fix AIX priority for the hats daemon.
- **Log_Length**: Number of lines in the log file before trim it.
- **Pinning**: This controls the memory pinning strategy. For PSSP 3.1 levels below PTF set 5, this value is being ignored, and the text is pinned by default. For system with PSSP 3.1 and PTF set 5 or later, the TS_Config attribute is honored, and the possible values are:
  - Text: The daemon will attempt to pin Text pages.
  - Data: The daemon will attempt to pin Data pages.
  - Proc: The daemon will attempt to pin all pages.
  - None: Causes no pages to be pinned by the daemon.
  - No value causes the default behavior: Text pages are pinned.

The other values you seen in the log file are coming from the daemon itself (or values passed by the hats script) that are not to be changed by the system administrator (unless instructed to do so).

Line 19 specifies that the values listed below are for the ring in offset 0, which is the SP Ethernet network in this case. The hats daemon manages a complete separate set of tunables for each ring (although PSSP does not provide any interface at the present release to specify different tunables for different networks; HACMP 4.3 does).

```
+37  01/25  10:17:17 hatsd[1]: Timer intervals used by offset 1 *****************
+38  01/25  10:17:17 hatsd[1]: HEARTBEAT Send Interval        = (1.000000) seconds.
+39  01/25  10:17:17 hatsd[1]: HEARTBEAT Receive Interval     = (2.000000) seconds.
+40  01/25  10:17:17 hatsd[1]: Group Stability Interval       = (10.000000) seconds.
+41  01/25  10:17:17 hatsd[1]: If Unstable Set NCT Interval   = (40.000000) seconds.
+42  01/25  10:17:17 hatsd[1]: JOIN Interval                  = (10.000000) seconds.
+43  01/25  10:17:17 hatsd[1]: Batch Join Interval            = (2.000000) seconds.
+44  01/25  10:17:17 hatsd[1]: PTC_ACK Interval               = (2.000000) seconds.
+45  01/25  10:17:17 hatsd[1]: COMMIT_ACK Interval            = (2.000000) seconds.
+46  01/25  10:17:17 hatsd[1]: COMMIT_BROADCAST_ACK Interval  = (2.000000) seconds.
+47  01/25  10:17:17 hatsd[1]: COMMIT Interval                = (5.000000) seconds.
```

```
+48  01/25  10:17:17 hatsd[1]: PPROCLAIM Interval              = (5.000000) seconds.
+49  01/25  10:17:17 hatsd[1]: NODE_CONNECTIVITY Interval      = (10.000000) seconds.
+50  01/25  10:17:17 hatsd[1]: GROUP_CONNECTIVITY Interval     = (20.000000) seconds.
+51  01/25  10:17:17 hatsd[1]: Default Wait Interval           = (10.000000) seconds.
+52  01/25  10:17:17 hatsd[1]: Fibrillate Count                = 4
+53  01/25  10:17:17 hatsd[1]: MAX_PTC_RETRY 3. MAX_COMMIT_RETRY 3. MAX_ACK_RETRY 3
+54  01/25  10:17:17 - (0) - incorporate_configuration() IPAT disabled on offset 1
```

These are the same set of tunables, but these apply to the second network (offset 1) which is the SP Switch network.

Finally, the daemon has its own tunables specified in a separate stanza.

```
+55  01/25  10:17:17 hatsd[0]: Timer intervals used by daemon *******************
+56  01/25  10:17:17 hatsd[0]: Node Reachability Interval      = (1.000000) seconds.
+57  01/25  10:17:17 hatsd[0]: Refresh Quiesce Interval        = (20.000000) seconds.
+58  01/25  10:17:17 hatsd[0]: Refresh Propagation Interval    = (5.000000) seconds.
+59  01/25  10:17:17 hatsd[0]: Dead Man Switch Disabled
+60  01/25  10:17:17 hatsd[0]: Remote socket send buffer size = 32768.
+61  01/25  10:17:17 hatsd[0]: Remote socket receive buffer size = 65536.
+62  01/25  10:17:17 hatsd[0]: Listening socket send buffer size = 65536.
+63  01/25  10:17:17 - (0) - Hb_Lsock::Listen()                listening to be enabled
+64  01/25  10:17:18 - (0) - init_group()                        node_idx = 0
+65  01/25  10:17:18 hatsd[0]: Type: en0 Address: 192.168.5.150, BROADCAST.
+66  01/25  10:17:18 hatsd[0]: Tunstable      = (917277438.704810) seconds.
+67  01/25  10:17:18 hatsd[0]: TifUnstableSetNCT      = (917277438.704810) seconds.
```

Some are additional tunables for the hats daemon. *Tunstable* in Line 66 is the time-of-day when the instability timer was set. It will expire when the time-of-day reaches *Tunstable + Tunstable_interval*. Tunstable_interval is 10 times the hb frequency. This is the interval where the adapter membership group stays in an "unstable state". No new node membership information is generated for unstable groups. The goal of having the notion of group stability is to prevent Group Services from receiving the intermediate sequence of "node up" events while the system is being booted. If the instability timer works right, then Group Services should receive a single node up event. The instability timer is initially set when the initial singleton group is formed and is reset each time the group membership changes.

TifUnstableSetNCT is related to the previous timer. There are conditions under which adapter groups become indefinitely unstable. In this case, node membership information needs to be updated. There have been cases in the past where a node would die, but TS would not detect this because one of the adapter membership groups was unstable for too long. If TifUnstableSetNCT expires, TS goes ahead and computes node membership even for unstable groups.

The node reachability interval is the interval between an event that causes recomputation of node membership (such as receiving a Group Connectivity message or committing a new adapter group) and actually recomputing node membership.

The *Refresh Quiesce Interval*, from Line 57, is how long to wait between the request for a refresh and the actual change in the daemon's data structures. During the quiesce period, the daemon also sends protocol messages to help propagate to its peers the need for doing refresh. The propagate timer (*Refresh Propagation Interval* in Line 58) tells how often to send these propagation messages.

The quiesce timer is currently defined as 20 times the largest send frequency parameter among all the networks.

The singleton state

After this initialization, the hats daemon starts what is called a *Singleton*. A Singleton is a ring made of a single member. Each daemon will start in this state, where the Group Leader, the Crown Prince (next one in line), the Upstream Neighbor, and the Downstream Neighbor are the same.

You can see this state from the log file as follows:

```
+68  01/25  10:17:18 hatsd[0]: My New Group ID = (192.168.5.150:0x46ac8afe) and is
Unstable.
+69          My Leader is              (192.168.5.150:0x46ac8afe).
+70          My Crown Prince is        (192.168.5.150:0x46ac8afe).
+71          My upstream neighbor is   (192.168.5.150:0x46ac8afe).
+72          My downstream neighbor is (192.168.5.150:0x46ac8afe).
```

We said we were looking into the hatsd’s log file on the CWS; so, there is only one ring where the hats daemon on the CWS is present (the SP Ethernet ring). In any other node with switch connectivity, you should see this stanza duplicated for the switch with the correspondent IP addresses for that adapter (css0).

Also from the previous listing, you see in Line 68 that the state of this Singleton ring is *Unstable* meaning that the membership and topology information for this ring is not ready.

The daemon acting as the Group Leader for this Singleton ring will announce all the other members (well, it will announce to itself) that the ring must be formed and the membership will be distributed. It also will accept local connection from clients (Group Services daemon is the only hats client) as you can see in the following listing:

```
+73  01/25  10:17:20 - (0) - Hb_Lsock::realAccept()           Accepted connection on fd=7 queued
    notification for main thread
+74  01/25  10:17:20 hatsd[0]: Accepted Client Connection on Socket FD=7
+75  01/25  10:17:20 hatsd[0]: Client on Socket File Descriptor 7 is hagsd with PID 14472
+76  01/25  10:17:20 hatsd[0]: Received a HB_GET_CONFIG_INFO request from client hagsd with PID =
    14472.
+77  01/25  10:17:20 hatsd[0]: Received a HB_SUBSCRIBE request from client hagsd with PID = 14472.
+78  01/25  10:17:20 hatsd[0]: Sending Notification packet for [Hb_Config_Group] subscription.
```

```
+79  01/25  10:17:20 hatsd[0]: Subscription name [hagsd_pm_client], type [Hb_All_Nodes_Subscription]
    for event [Hb_All_Events].
+80  01/25  10:17:24 hatsd[0]: Received a HB_SUBSCRIBE request from client hagsd with PID = 14472.
+81  01/25  10:17:24 hatsd[0]: Subscription name [hps0], type [Hb_Network_Subscription] for event
    [Hb_All_Events].
+82  01/25  10:17:24 hatsd[0]: Sending Notification packet for [hps0] subscription.
+83  01/25  10:17:28 hatsd[0]: Group becoming Stable.
+84  01/25  10:17:28 hatsd[0]: Tunstable      = (917277438.704810) seconds.
+85  01/25  10:17:28 hatsd[0]: Sending PTC to new membership:
+86          0 (192.168.5.150:0x46ac8afe)
+87  01/25  10:17:28 hatsd[0]: PTC's sent = 1, PTC's received = 1.
+88  01/25  10:17:28 hatsd[0]: Deleting Current Group, (192.168.5.150:0x46ac8afe), from the NCT.
+89  01/25  10:17:28 hatsd[0]: Sending node [Hb_New_Group] notifications.
+90  01/25  10:17:28 hatsd[0]: Sending Notification packet for [hagsd_pm_client] subscription.
+91  01/25  10:17:28 hatsd[0]: Received a COMMIT BROADCAST request from (192.168.5.150:0x46ac8afe) in
    group (192.168.5.150:0x46ac
8b08).
+92  01/25  10:17:28 hatsd[0]: Other group:
+93          0 (192.168.5.150:0x46ac8afe)
+94  01/25  10:17:28 hatsd[0]: Sending a COMMIT BROADCAST ACK response to (192.168.5.150:0x46ac8afe).
+95  01/25  10:17:28 hatsd[0]: Received a COMMIT request from (192.168.5.150:0x46ac8afe) in group
    (192.168.5.150:0x46ac8b08).
+96  Hash_init: size was 0, making it 257
+97  Hash_init: size was 0, making it 257
+98  01/25  10:17:28 hatsd[0]: Sending a COMMIT ACK response to (192.168.5.150:0x46ac8afe).
+99  01/25  10:17:28 hatsd[0]: Sending adapter [Hb_New_Group] notifications.
+100  01/25  10:17:29 - (0) - check_state_3p()                  group is singleton: start pinging
+101  01/25  10:17:29 hatsd[0]: My New Group ID = (192.168.5.150:0x46ac8b08) and is Stable.
+102          My Leader is              (192.168.5.150:0x46ac8afe).
+103          My Crown Prince is        (192.168.5.150:0x46ac8afe).
+104          My upstream neighbor is   (192.168.5.150:0x46ac8afe).
+105          My downstream neighbor is (192.168.5.150:0x46ac8afe).
```

From Line 75, you can see that hags daemon (hagsd) has subscribed to Topology Services information. It has subscribed to all the events within Topology Services, which will make the hats daemon send notifications to the Group Services daemon for any event it gets generated (events such as new rings, changes in rings topology, and so on).

In Line 83, the Singleton group becomes stable. In Line 85, the daemon sends out the Prepare To Commit (PTC) message to itself. Lines 91 and 94 show that the daemon is preparing to commit its ring with one member. The group ID you see in the listing corresponds to the IP address of the Group Leader for that group (ring), which, in this case, is the same machine (remember that we still are in the Singleton state).

Line 101 shows that, finally, the ring is stable with a single member.

Making up the ring

```
+106  01/25  10:17:33 hatsd[0]: Netmon program returned. Local adapter is DOWN
+107  01/25  10:17:38 - (0) - send_node_connectivity()          CurAdap 0
+108  01/25  10:17:56 hatsd[0]: Netmon program returned. Local adapter is UP
+109  01/25  10:18:03 hatsd[0]: Received a JOIN request from (192.168.5.1:0x46ac8b28).
+110  01/25  10:18:03 hatsd[0]: This is the first JOIN request!
+111  01/25  10:18:03 hatsd[0]: Base group:
+112          0 (192.168.5.150:0x46ac8afe)
```

```
+113  01/25  10:18:03 hatsd[0]: Other group:
+114          0 (192.168.5.1:0x46ac8b28)
+115  01/25  10:18:03 hatsd[0]: Merged group:
+116          0 (192.168.5.1:0x46ac8b28)      1 (192.168.5.150:0x46ac8afe)
+117  01/25  10:18:06 hatsd[0]: Sending PTC to new membership:
+118          0 (192.168.5.1:0x46ac8b28)      1 (192.168.5.150:0x46ac8afe)
+119  01/25  10:18:06 hatsd[0]: PTC's sent = 2, PTC's received = 1.
+120  01/25  10:18:06 hatsd[0]: Received the final PTC response, committing group.
+121  01/25  10:18:06 hatsd[0]: Deleting Current Group, (192.168.5.150:0x46ac8b08), from the NCT.
+122  01/25  10:18:06 hatsd[0]: Received a COMMIT request from (192.168.5.150:0x46ac8afe) in group
(192.168.5.150:0x46ac8b2b).
+123  Hash_init: size was 0, making it 257
+124  Hash_init: size was 0, making it 257
+125  01/25  10:18:06 hatsd[0]: Joining Adapters = 1: 40
+126  01/25  10:18:06 hatsd[0]: Sending a COMMIT ACK response to (192.168.5.150:0x46ac8afe).
+127  01/25  10:18:06 hatsd[0]: Sending adapter [Hb_Join] notifications.
+128  01/25  10:18:06 hatsd[0]: Sending adapter [Hb_New_Group] notifications.
+129  01/25  10:18:06 hatsd[0]: Node Connectivity Message stopped on adapter offset 0.
+130  01/25  10:18:06 hatsd[0]: My New Group ID = (192.168.5.150:0x46ac8b2b) and is Stable.
+131          My Leader is              (192.168.5.150:0x46ac8afe).
+132          My Crown Prince is        (192.168.5.1:0x46ac8b28).
+133          My upstream neighbor is   (192.168.5.1:0x46ac8b28).
+134          My downstream neighbor is (192.168.5.1:0x46ac8b28).
```

Line 106 shows the return from netmon (an external process that monitors the adapters and notifies the hats daemon when they become available or unavailable).

Line 109 shows the first JOIN request from one of the members of this network. It is from node 1 (sp5n01, 192.168.5.1) requesting to join ring 192.168.5.150, which is the highest IP address in this network.

The hats daemon on the CWS will merge this node into the current ring, and then it will send the new membership list as shown by Line 117. Then after acknowledging the new membership list, they will commit the ring with the two members as shown in Line 130.

There are two set of notifications to clients. Lines 127-128 show the ADAPTER notifications for changes in rings membership. These always come with the arrival of a COMMIT message containing the new membership.

Lines 135-138 show the NODE notifications for changes in node reachability. Node reachability is recomputed one second after receiving the COMMIT message.

```
+135  01/25  10:18:07 hatsd[0]: Sending node [Hb_Join] notifications.
+136  01/25  10:18:07 hatsd[0]: Sending Notification packet for [hagsd_pm_client] subscription.
+137  01/25  10:18:07 hatsd[0]: Sending node [Hb_New_Group] notifications.
+138  01/25  10:18:07 hatsd[0]: Sending Notification packet for [hagsd_pm_client] subscription.
```

The hags daemon is being notified about the changes in the group (ring).

You may see that the same will happens for all the nodes in the SP Ethernet network as well as the node in the SP Switch network.

Finally, all the members defined in the machines.lst file should be active in the rings they are listed on. If some nodes are not present, the Group Leader for each ring will send Proclaim packets to those members every five seconds as you can see from the *Proclaim Interval* defined at Line 30 in the log file.

4.1.3 Using the lssrc command

The hats daemon's log file is something you should use in case of problems with Topology Services. For checking status information, the `lssrc` command gives the basic information you need. You probably have used the `lssrc` command before. As you may know, it gives you information about the SRC subsystem. With this command you can check if the subsystem is alive, and also, you may request status information from the daemon itself (if it is programmed to responds to such as request). The following output has been numbered per line in order to better identify the information provided by this command:

```
+1  [sp5en0:/]# lssrc -ls hats.sp5en0
+2  Subsystem         Group            PID     Status
+3   hats.sp5en0      hats             24512   active
+4   Network Name   Indx Defd Mbrs St Adapter ID      Group ID
+5   SPether        [ 0]    5    5  S 192.168.5.150   192.168.5.150
+6   SPether        [ 0]              0x46ac8afe      0x46ac8b83
+7   HB Interval = 1 secs. Sensitivity = 4 missed beats
+8     2 locally connected Clients with PIDs:
+9   haemd(  5540) hagsd( 14472)
+10     Configuration Instance = 917277434
+11     Default: HB Interval = 1 secs. Sensitivity = 4 missed beats
+12     CWS = 192.168.5.150
```

Line 1 is the command we executed. We used the `-ls` flag because we did not just want to know if the subsystem was active. We also want to get the status information from the daemon itself (not all SRC subsystems support the `-l` flag).

Lines 2 and 3 correspond to the standard output you get when you use the `-s` flag only. Therefore, you can see if the subsystem is active[2].

Lines 4 through 6 contain information about the rings this machine is on. In this case, the CWS is on one ring only (SP Ethernet).

Let us take Line 4 and define each one of the terms listed there:

- **Network Name**: It is just a name, SPether or SPswitch.

[2] For SRC, a subsystem is active if the socket connection to it is still open. In some situations, this maybe true even if the daemon is hanging. Active does not necessarily mean that the daemon is working; it means that the daemon is still there, but it may be hanging.

- **Indx**: Index number for the network or group. 0 is SPether; 1 is SPswitch.
- **Defd**: Number of nodes defined in this network (all nodes in a partition plus the CWS if this is the SPether network).
- **Mbrs**: Active members in the ring, the ones who joined the Group Leader.
- **St**: State of the ring. S: Stable, U: Unstable, D: Down.
- **Adapter ID**: This is an unique identifier for the adapter on that ring (network), which is a combination of the IP address and an incarnation number based on the time of the day (number of seconds since the hats subsystem started).
- **Group ID**: This is a unique identifier for the ring. It corresponds to a combination of the IP address of the GL and the time of the day when the group was formed.

Line 7 shows the tunables for the ring (network) right above. Although PSSP does not support different tunables based on networks, TS does, and this entry represents the tunables for this particular network.

Lines 8 and 9 provide information about the clients connected to Topology Services. TS only supports local clients or clients connected through UNIX Domain Sockets (UDS). Although the listing contains two processes (haemd and hagsd), only one is truly a hats client. Event Manager (haemd) is not connected directly to Topology Services, but it uses the Reliable Messaging Library that uses the TS information. See 4.1.4, “Reliable messaging library” on page 207 for more information.

Finally, Lines 10 to 12 show information about the configuration instance (an identifier for the active configuration), the default tunables (which are the ones used by all the networks in PSSP), and the IP address of the CWS.

One of the things to look for when executing this command in different nodes is the Group ID and the configuration instance, which should be the same across nodes in the same partition. Also, the status of the ring is something you should pay attention to. When it says U (Unstable), it means TS is still configuring the ring; so, you should give it more time to finish. Do not start diagnosing problems when the ring is unstable, wait until it says S (Stable) or D (Down).

Let us compare the information we got from the CWS with the one we get from one of the nodes. Let us look at node 1 (sp5n01):

```
[sp5n01:/]# lssrc -ls hats
Subsystem         Group            PID     Status
 hats             hats             16300   active
```

```
Network Name   Indx Defd Mbrs St Adapter ID      Group ID
SPether        [ 0]    5    5  S 192.168.5.1     192.168.5.150
SPether        [ 0]              0x46ac8b28      0x46ac8b83
HB Interval = 1 secs. Sensitivity = 4 missed beats
SPswitch       [ 1]    4    4  S 192.168.15.1    192.168.15.13
SPswitch       [ 1]              0x46ac8ea2      0x46ac8eab
HB Interval = 1 secs. Sensitivity = 4 missed beats
  2 locally connected Clients with PIDs:
haemd(  9916) hagsd( 15244)
  Configuration Instance = 917277434
  Default: HB Interval = 1 secs. Sensitivity = 4 missed beats
  CWS = 192.168.5.150
```

On node 1, there is no need to specify the partition name extension for the subsystem since a node belongs to a single partition at any time.

Since node 1 is connected to two networks (SP Ethernet and SP Switch), we see those two listed in the output of the `lssrc` command, each network with its own tunables (although they are always equal to the default tunables listed at the end of the output if TS is running in a PSSP domain).

As you can see, the Adapter ID is the IP address for the adapter (although TS uses the IP address plus the Instance number) and the Group ID which is the IP address of the Group Leader for that network (ring). The Group Leader in any given network should be the same no matter where you run this command in the system partition. If it is different, then you have a partitioned ring, which is not something you want to deal with.

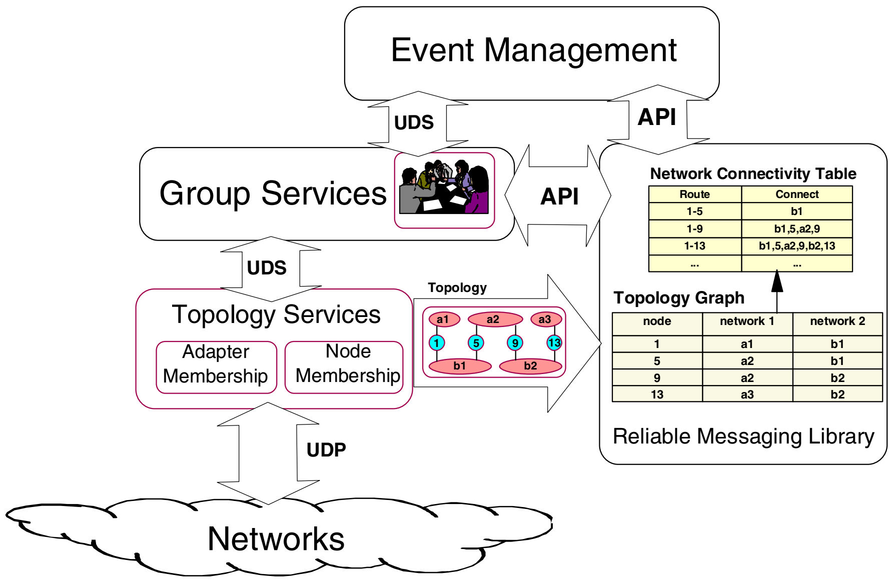
A partitioned ring is generated when two or more sections on a ring have split apart, usually because of connectivity problems. Routing is a very common tale. Each section of the ring "thinks" the other section is dead; so, they form a ring for their own (with the surviving ones). Until the connectivity problem is fixed, you will see more than one Group Leader, one per section.

In our case, everything is fine since we have the same Group Leader for the SP Ethernet on node 1 as we did on the CWS. The switch network (ring) also looks okay with four (4) members defined and the four active in the ring. Node 13 (sp5sw13) is the Group Leader for that ring (things work as designed).

4.1.4 Reliable messaging library

Probably, you will never have to deal with this library if you are not developing distributed applications, but it is part of the RSCT architecture, and it has some implications in the command outputs we have seen so far.

This library is used by Group Services and Event Management for sending and receiving messages from their peers. Actually, each daemon will link the library at execution time, and it will use it for sending and receiving messages. The library instead "subscribes" to topology services for connectivity information, and it maintains a network connectivity table. Figure 126 shows the relationship between the library and the RSCT subsystems.



Route	Connect
1-5	b1
1-9	b1,5,a2,9
1-13	b1,5,a2,9,b2,13
...	...

node	network 1	network 2
1	a1	b1
5	a2	b1
9	a2	b2
13	a3	b2

Figure 126. Reliable messaging library

As you can see from Figure 126, the Reliable Messaging Library is a "client" for Topology Services, that is why we see Event Manager listed as a client even though Event Manager does not connect to Topology Services directly.

4.1.5 Consequences of missing hats

Before discussing the problem of missing hats in some nodes, let us see what happens in the log files when a node goes down. The configuration shown in Figure 124 on page 193 can be logically represented by the following graph:

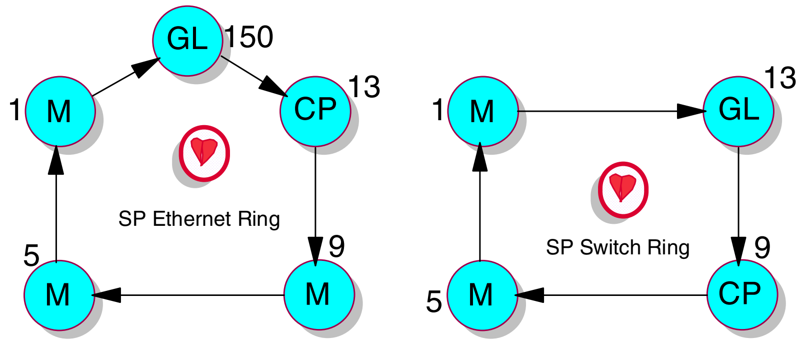


Figure 127. Logical representation of Topology Services rings

As you can see in Figure 127, the ring is formed based on IP addresses with the Group Leader for the SP Ethernet network being the CWS and node 13 for the SP Switch network in this case (highest IP address in these networks).

Let us see a simple case where one of the *Generic* nodes goes down. We will stop Topology Services in node 5, which will simulate a node down.

Who will detect that node 5 is down? It will be node 1, so let us see what happens in the log files of node 1 and the Group Leader.

```
+411  01/25  10:33:01 hatsd[0]: Node Connectivity Message stopped on adapter offset 0.
+412  01/26  10:15:54 hatsd[0]: Node (192.168.5.5:0x46ac8b4b) is dead.
+413  01/26  10:15:54 hatsd[0]: Notifying leader (192.168.5.150:0x46ac8afe) of death.
+414  01/26  10:15:54 hatsd[1]: Node (192.168.15.5:0x46ac8ea0) is dead.
+415  01/26  10:15:54 hatsd[1]: Notifying leader (192.168.15.13:0x46ac8ea2) of death.
+416  01/26  10:15:54 hatsd[1]: Node (192.168.15.5:0x46ac8ea0) is dead.
+417  01/26  10:15:54 hatsd[1]: Notifying leader (192.168.15.13:0x46ac8ea2) of death.
+418  01/26  10:15:54 hatsd[1]: Received a PTC request from (192.168.15.13:0x46ac8ea2) in
group (192.168.15.13:0x46addc2a).
+419  01/26  10:15:54 hatsd[0]: Received a PTC request from (192.168.5.150:0x46ac8afe) in
group (192.168.5.150:0x46addc2a).
```

From the log file in node 1, we can see in Lines 412 and 414 that the daemon in node 1 detects both adapters (Ethernet and Switch) down. Then, in Lines 413 and 415, it notifies the Group Leaders of each ring about the lost. The Group Leaders, in return (Lines 418 and 419), send back a Prepare To Commit (PTC) message to modify the rings.

If we look at the log file of the Group Leader, we will see the following:

SP Ethernet Group Leader (192.168.5.150):

```
+255  01/26  10:15:54 hatsd[0]: Received a DEATH IN FAMILY message from
(192.168.5.1:0x46ac8b28) in group (192.168.5.150:0x46ac8b8
3).
+256  01/26  10:15:54 hatsd[0]: Sending PTC to new membership:
+257          0 (192.168.5.1:0x46ac8b28)      1 (192.168.5.9:0x46ac8b66)
+258          2 (192.168.5.13:0x46ac8b80)     3 (192.168.5.150:0x46ac8afe)
```

SP Switch Group Leader (192.168.15.13):

```
+259  01/26  10:15:54 hatsd[1]: Received a DEATH IN FAMILY message from
(192.168.15.1:0x46ac8ea2) in group (192.168.15.13:0x46ac8e
ab).
+260  01/26  10:15:54 hatsd[1]: Sending PTC to new membership:
+261          0 (192.168.15.1:0x46ac8ea2)     1 (192.168.15.9:0x46ac8ea7)
+262          2 (192.168.15.13:0x46ac8ea2)
```

The nodes receive the new membership list from the leaders. From the SP Switch Group Leader (192.168.15.13):

```
+420  01/26  10:15:54 hatsd[1]: Received a COMMIT BROADCAST request from
(192.168.15.13:0x46ac8ea2) in group (192.168.15.13:0x46ad
dc2a).
+421  01/26  10:15:54 hatsd[1]: Other group:
+422          0 (192.168.15.1:0x46ac8ea2)     1 (192.168.15.9:0x46ac8ea7)
+423          2 (192.168.15.13:0x46ac8ea2)
```

Then, from the SP Ethernet Group Leader (192.168.5.150):

```
+437  01/26  10:15:54 hatsd[0]: Received a COMMIT BROADCAST request from
(192.168.5.150:0x46ac8afe) in group (192.168.5.150:0x46ad
dc2a).
+438  01/26  10:15:54 hatsd[0]: Other group:
+439          0 (192.168.5.1:0x46ac8b28)      1 (192.168.5.9:0x46ac8b66)
+440          2 (192.168.5.13:0x46ac8b80)     3 (192.168.5.150:0x46ac8afe)
```

Finally, the two rings are committed (Lines 452-456 and 457-461), and the local clients are notified (Lines 462-465):

```
+452  01/26  10:15:55 hatsd[0]: My New Group ID = (192.168.5.150:0x46addc2a) and is
Stable.
+453          My Leader is              (192.168.5.150:0x46ac8afe).
+454          My Crown Prince is        (192.168.5.13:0x46ac8b80).
+455          My upstream neighbor is   (192.168.5.9:0x46ac8b66).
+456          My downstream neighbor is (192.168.5.150:0x46ac8afe).
+457  01/26  10:15:55 hatsd[1]: My New Group ID = (192.168.15.13:0x46addc2a) and is
Stable.
+458          My Leader is              (192.168.15.13:0x46ac8ea2).
+459          My Crown Prince is        (192.168.15.9:0x46ac8ea7).
+460          My upstream neighbor is   (192.168.15.9:0x46ac8ea7).
+461          My downstream neighbor is (192.168.15.13:0x46ac8ea2).
+462  01/26  10:15:55 hatsd[0]: Sending node [Hb_Death] notifications.
+463  01/26  10:15:55 hatsd[0]: Sending Notification packet for [hagsd_pm_client]
subscription.
+464  01/26  10:15:55 hatsd[0]: Sending node [Hb_New_Group] notifications.
+465  01/26  10:15:55 hatsd[0]: Sending Notification packet for [hagsd_pm_client]
subscription.
```

The new rings are shown in Figure 128.

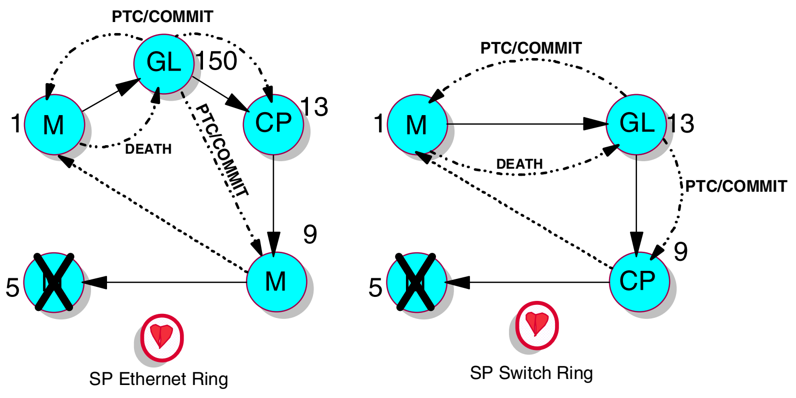


Figure 128. New rings after node 5 down

Now that node 5 is out of the rings, the Group Leaders of both rings will keep sending Proclaim messages to node 5 every five seconds (by default) until node 5 rejoins the rings.

What are the consequences of missing hats on node 5? Well, node 5 is considered down, which means that, although AIX and applications may be up un running on that node, Topology Services "sees" that node down, and it reports it to the upper layers. So, eventually, this information will get to the host responds and Membership resource monitors who will change the resource variables representing host and adapter responds (IBM.PSSP.Response.Host.State, IBM.PSSP.Membership).

The host responds daemon (hrd) running on the CWS subscribes to the membership variable with the local Event Manager daemon. When the change occurs, Event Management notifies the host responds daemon, who then changes the SDR information regarding host responds (host_responds SDR class) for node 5 (host_responds represents en0 connectivity).

So, what we see from the SDR is the following:

```
sp5en0:/var/ha/log>SDRGetObjects host_responds
node_number  host_responds
          1             1
          5             0
          9             1
         13             1
```

As soon as we start Topology Services on node 5, the information gets bubbled up to the resource variables and then to the SDR host_responds class.

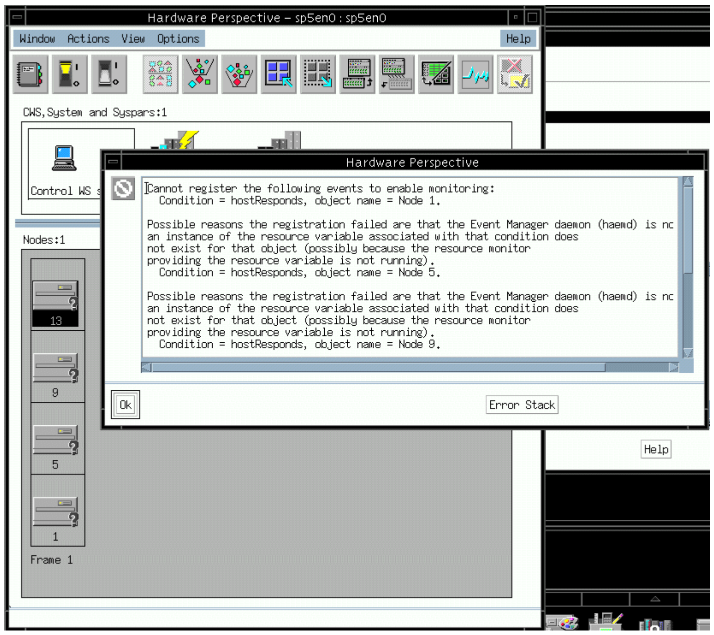
But what happens if Topology Services on the CWS is not running?

From Topology Services view point, the CWS is just node 0 (besides the initial machines.lst file that resides in the SDR), but from the host responds point of view, the CWS is the only node that runs this daemon. So, if hats is not running on the CWS, Group Services and Event Management will not be running either (remember, they are clients of TS); so, the host responds daemon (hrd) will not be able to subscribe with Event Manager for the membership resource variable. This will cause the host responds daemon to wait indefinitely for the Event Manager daemon to accept the connection. As a matter of fact, if you check the host responds daemon with the `lssrc` command, you will see something like:

```
sp5en0:/var/ha/log>lssrc -ls hr.sp5en0
Subsystem         Group            PID     Status
 hr.sp5en0        hr               8204    active
   waiting for Event Manager initialization...
```

This means that the daemon is waiting for Event Manager; so, nobody is updating the SDR with the host responds information, despite the fact that the other nodes and their RSCT stacks (TS, GS and EM) are up and running.

The Hardware Perspective will also be affected by this Topology Services problem. If you try to monitor host responds from this graphical interface, you will get an error message similar to the one shown in Figure 129 on page 213.



```
Cannot register the following events to enable monitoring:
  Condition = hostResponds, object name = Node 1.

Possible reasons the registration failed are that the Event Manager daemon (haemd) is nc
an instance of the resource variable associated with that condition does
not exist for that object (possibly because the resource monitor
providing the resource variable is not running).
  Condition = hostResponds, object name = Node 5.

Possible reasons the registration failed are that the Event Manager daemon (haemd) is nc
an instance of the resource variable associated with that condition does
not exist for that object (possibly because the resource monitor
providing the resource variable is not running).
  Condition = hostResponds, object name = Node 9.
```

Figure 129. Problems with Hardware Perspective if TS is not running

How would this affect applications? The effect should be minimum if applications are not using the host responds SDR bit for checking node availability. However, if they do, as some PSSP subsystems do, the consequences could be more critical.

Since the SDR is not being updated, and some subsystems "trust" the SDR value for host responds, it may cause problems when starting applications or modifying status information. For example, the `Estart` command checks the host_responds bit in the SDR for the nodes that are defined as oncoming primary and oncoming backup primary. It also use this value for distributing the switch topology file to the nodes.

So, in order to avoid problems like this, make sure the subsystems on the CWS are working properly so that the information in the SDR is correct and accessible.

Some thoughts on switch responds

Switch responds (switch_responds) is managed differently. The value in the SDR is updated directly by the switch primary daemon; so, the values in the

SDR are always the right ones. The resource monitor for the IBM.PSSP.Response.Switch.State resource variable takes the information from the SDR and updates the resource variable accordingly. So, as long as the switch primary daemon has access to the SDR (connectivity with the CWS), the values in the SDR for switch_responds should be accurate.

4.1.6 Case study

The single most frequent problem with Topology Services and, in general, with RSCT, is host responds. This case study demonstrates how a host responds problem may include multiple components within RSCT. We will develop this case study in the three sections that cover the core of the RSCT infrastructure. We start with the symptom, which is host responds zero on a particular node. If we were using SP Perspectives, we would have seen a red host responds for the node in question or the following output from the command line:

```
[sp5en0:/]# spmon -G -d
1.  Checking server process
    Process 15348 has accumulated 153 minutes and 36 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       on  on  on  N/A

5.  Checking nodes
---------------------------------- Frame 1 -------------------------------------
Frame  Node    Node         Host/Switch  Key     Env   Front Panel   LCD/LED is
Slot   Number  Type  Power   Responds    Switch  Fail    LCD/LED      Flashing
--------------------------------------------------------------------------------
  1       1    high    on   yes  no      normal   no   LCDs are blank   no
  5       5    high    on    no  no      normal   no   LCDs are blank   no
  9       9    high    on   yes  no      normal   no   LCDs are blank   no
 13      13    high    on   yes  no      normal   no   LCDs are blank   no
```

From the output of the spmon command, we see that the host responds bit for node 5 is zero. We can confirm this by using an SDR command as follows:

```
[sp5en0:/]# SDRGetObjects host_responds
node_number  host_responds
          1            1
          5            0
```

```
            9           1
           13           1
```

There are several ways to approach a problem like this. However, we will use a bottom-up approach in order to provide some techniques to debug problems within the RSCT layers. We start with Topology Services, and then we continue with the other two layers in later sections of this chapter.

The first single command we need to run is the `lssrc` command on the CWS to list the status of the Topology Services subsystem. We issue the following command:

```
[sp5en0:/]# lssrc -ls hats.sp5en0
Subsystem         Group            PID     Status
 hats.sp5en0      hats             10974   active
 Network Name   Indx Defd Mbrs St Adapter ID      Group ID
 SPether        [ 0]    5    4  S 192.168.5.150   192.168.5.150
 SPether        [ 0]              0x401861cc      0x40444b6d
 HB Interval = 1 secs. Sensitivity = 4 missed beats
   1 locally connected Client with PID:
 hagsd( 23312)
   Configuration Instance = 941121985
   Default: HB Interval = 1 secs. Sensitivity = 4 missed beats
   CWS = 192.168.5.150
```

The output from this command seems consistent with the host responds status, given that there are four members in the SPether ring. So, probably the Topology Services daemon is not running on node 5. We also notice something else. There is not an Event Management client listed in the client section. Certainly this is a problem, but we need to solve the Topology Services problem first.

Let us log on on node 5 and verify that the Topology Services daemon is running. We use the following command:

```
[sp5n05:/]# lssrc -ls hats
0513-036 The request could not be passed to the hats subsystem.
Start the subsystem and try your command again.
```

It looks like the daemon is not running. We restart it and check again as follows:

```
[sp5n05:/]# startsrc -s hats
0513-059 The hats Subsystem has been started. Subsystem PID is 18600.
[sp5n05:/]# lssrc -ls hats
Subsystem         Group            PID     Status
 hats             hats             18600   active
 Network Name   Indx Defd Mbrs St Adapter ID      Group ID
```

```
SPether        [ 0]    5    0  D
HB Interval = 1 secs. Sensitivity = 4 missed beats
SPswitch       [ 1]    4    0  D
HB Interval = 1 secs. Sensitivity = 4 missed beats
  1 locally connected Client with PID:
hagsd( 17286)
  Configuration Instance = 941121985
  Default: HB Interval = 1 secs. Sensitivity = 4 missed beats
  CWS = 192.168.5.150
```

According to this output, both rings are down. We then check for any entry in the AIX error log by executing the following command:

```
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
E18E984F   1130171799 P S SRC            SOFTWARE PROGRAM ERROR
C60BB505   1130171799 P S SYSPROC        SOFTWARE PROGRAM ABNORMALLY TERMINATED
12081DC6   1130171799 P S haemd          SOFTWARE PROGRAM ERROR
E18E984F   1130171799 P S SRC            SOFTWARE PROGRAM ERROR
C60BB505   1130171799 P S SYSPROC        SOFTWARE PROGRAM ABNORMALLY TERMINATED
12081DC6   1130171799 P S haemd          SOFTWARE PROGRAM ERROR
E18E984F   1130171799 P S SRC            SOFTWARE PROGRAM ERROR
C60BB505   1130171799 P S SYSPROC        SOFTWARE PROGRAM ABNORMALLY TERMINATED
12081DC6   1130171799 P S haemd          SOFTWARE PROGRAM ERROR
99FA80C7   1130171699 U S hags           SOFTWARE
12081DC6   1130170199 P S hags           SOFTWARE PROGRAM ERROR
E18E984F   1130170099 P S SRC            SOFTWARE PROGRAM ERROR
99FA80C7   1130170099 U S harmld         SOFTWARE
C3E70E5D   1130170099 P S pmand          SOFTWARE PROGRAM ERROR
C3E70E5D   1130170099 P S pmand          SOFTWARE PROGRAM ERROR
C60BB505   1130170099 P S SYSPROC        SOFTWARE PROGRAM ABNORMALLY TERMINATED
E18E984F   1130170099 P S SRC            SOFTWARE PROGRAM ERROR
12081DC6   1130170099 P S haemd          SOFTWARE PROGRAM ERROR
E18E984F   1130170099 P S SRC            SOFTWARE PROGRAM ERROR
12081DC6   1130170099 P S hags           SOFTWARE PROGRAM ERROR
369D049B   1129105299 I O SYSPFS         UNABLE TO ALLOCATE SPACE IN FILE SYSTEM
```

We do not see any error related to Topology Services from this output. The next place to check then is the Topology Services log files. First, we list all the Topology Services log files as follows:

```
[sp5n05:/var/ha/log]# ls -alt hats*
-rw-rw-rw-   1 root     system     12443 Nov 30 17:22 hats.30.171607.sp5en0
-rwxr-xr-x   1 root     system      1056 Nov 30 17:16 hats.sp5en0
-rwxr-xr-x   1 root     system      1056 Nov 30 17:16 hats.sp5en0.1
-rw-rw-rw-   1 root     system     22076 Nov 30 17:00 hats.28.104805.sp5en0
-rwxr-xr-x   1 root     system      1842 Oct 28 10:48 hats.sp5en0.2
```

The first two files are the ones we want to look at. First, we check the hats.sp5en0 file that corresponds to the Topology Services script (hats script). The content is as follows:

```
sp5n05:/var/ha/log]# cat hats.sp5en0
Tue Nov 30 17:16:05 EST 1999 hats
Called with options:
0 dead man switch timers in use.
exec /usr/sbin/rsct/bin/hatsd   -n 5
Tue Nov 30 17:16:06 EST 1999 ==========
---- /var/ha/run/hats.sp5en0/machines.lst ----
-rw-------   1 root     system       523 Nov 30 17:16
/var/ha/run/hats.sp5en0/machines.lst
*InstanceNumber=941121985
*configId=2225957613
*FileVersion=1
*!TS_realm=PSSP
TS_Frequency=1
TS_Sensitivity=4
TS_FixedPriority=38
TS_LogLength=5000
*!TS_PinText
Network Name SPether
Network Type ether
*
*Node Type Address
    0 en0 192.168.5.150
    1 en0  192.168.5.1
    5 en0  192.168.5.15
    9 en0  192.168.5.9
   13 en0  192.168.5.13
Network Name SPswitch
Network Type hps
*
*Node Type Address
    1 css0 192.168.15.1
    5 css0 192.168.15.5
    9 css0 192.168.15.9
   13 css0 192.168.15.13
---- /var/ha/run/hats.sp5en0//AdapAddrMask* ----
---- /var/ha/run/hats.sp5en0//TsTunables ----
---- /var/ha/run/hats.sp5en0//Subnet ----
---- /var/ha/run/hats.sp5en0//Network ----
---- /var/ha/run/hats.sp5en0//dms_loads.out ----
```

From this output, we see indirectly the machines.lst file contents. Something does not look right here. The IP address for the Ethernet adapter (en0) on node 5 does not look right. We confirm this by executing the following command on node 5:

```
[sp5n05:/var/ha/log]# ifconfig en0
en0: flags=e080863<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT>
        inet 192.168.5.5 netmask 0xffffff00 broadcast 192.168.5.255
```

From the daemon's log file, we also get the error message related to the IP address as follows:

```
11/30  17:16:08 prepare_disabled_adapter_for_pings hatsd[0]: 2523-001 My address
missing from the adapter information.
```

So, it looks like there is a problem with the machines.lst file coming from the CWS. But first, we verify name resolution just in case we have a DNS problem. We issue the following commands:

```
[sp5n05:/var/ha/log]# host sp5n05
sp5n05 is 192.168.5.5,  Aliases:   sp5n05.msc.itso.ibm.com
[sp5n05:/var/ha/log]# host 192.168.5.5
sp5n05 is 192.168.5.5,  Aliases:   sp5n05.msc.itso.ibm.com
```

They look fine; so, the problem must be on the CWS; so, we check the machines.lst file stored in the SDR by executing the following command:

```
[sp5en0:/tmp]# SDRRetrieveFile hats.machines.lst /tmp/hats.machines.lst
[sp5en0:/tmp]# cat hats.machines.lst
*InstanceNumber=941121985
*configId=2225957613
*FileVersion=1
*!TS_realm=PSSP
TS_Frequency=1
TS_Sensitivity=4
TS_FixedPriority=38
TS_LogLength=5000
*!TS_PinText
Network Name SPether
Network Type ether
*
*Node Type Address
    0 en0 192.168.5.150
    1 en0  192.168.5.1
    5 en0  192.168.5.15
    9 en0  192.168.5.9
   13 en0  192.168.5.13
Network Name SPswitch
Network Type hps
*
*Node Type Address
    1 css0 192.168.15.1
    5 css0 192.168.15.5
    9 css0 192.168.15.9
   13 css0 192.168.15.13
```

Yes, just as we thought. The machines.lst file on the CWS is wrong. Before we try to fix this problem, we need to verify that the SDR information for node 5 is correct. We issue the following command on the CWS:

```
[sp5en0:/tmp]# splstdata -a
                  List LAN Database Information

node#  adapt           netaddr           netmask           hostname  type  t/r rate
   enet_rate duplex       other_addrs
-----------------------------------------------------------------------------------
    1   css0      192.168.15.1      255.255.255.0   sp5sw01
           NA       NA
         NA     NA       ""
    5   css0      192.168.15.5      255.255.255.0   sp5sw05
           NA       NA
         NA     NA       ""
    9   css0      192.168.15.9      255.255.255.0   sp5sw09
           NA       NA
         NA     NA       ""
   13   css0     192.168.15.13      255.255.255.0   sp5sw13
           NA       NA
         NA     NA       ""
    1    en0       192.168.5.1      255.255.255.0   sp5n01
           bnc       NA
         10   half       ""
    5    en0       192.168.5.15      255.255.255.0   sp5n05
           bnc       NA
         10   half       ""
    9    en0       192.168.5.9      255.255.255.0   sp5n09
           bnc       NA
```

```
            10   half       ""
      13    en0      192.168.5.13      255.255.255.0   sp5n13
              bnc       NA
            10   half       ""
```

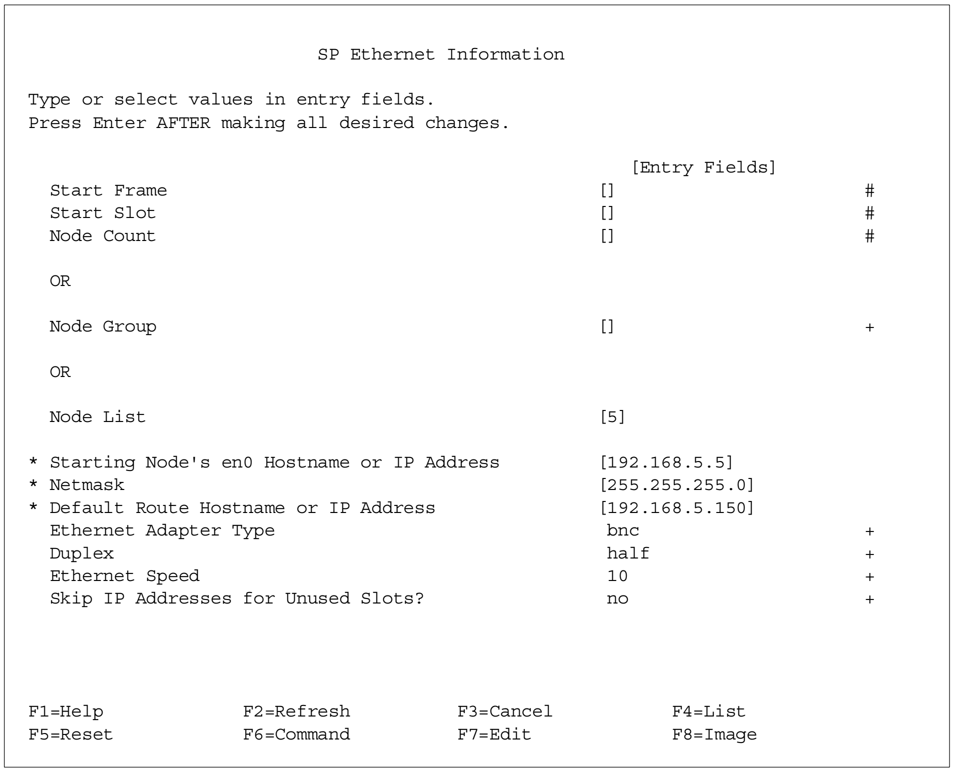
Or, we could use the `SDRGetObjects` command for the Adapter class as follows:

```
[sp5en0:/tmp]# SDRGetObjects Adapter node_number adapter_type netaddr
node_number  adapter_type netaddr
          1 en0          192.168.5.1
          5 en0          192.168.5.15
          9 en0          192.168.5.9
         13 en0          192.168.5.13
          1 css0         192.168.15.1
          5 css0         192.168.15.5
          9 css0         192.168.15.9
         13 css0         192.168.15.13
```

So, the IP address for the en0 adapter on node 5 is wrong. One possible explanation is a name resolution problem. Let us check the name resolution for node 5 as follows:

```
[sp5en0:/tmp]# host sp5n05
sp5n05 is 192.168.5.15,  Aliases:   sp5n05.msc.itso.ibm.com
[sp5en0:/tmp]# host 192.168.5.15
sp5n05 is 192.168.5.15,  Aliases:   sp5n05.msc.itso.ibm.com
```

The problem here is that the sp5n05 name is resolving to a wrong IP address, and this got propagated to the SDR. The easiest way to change the SDR Adapter class is by using SMIT. But first, we need to make sure the sp5n05 name resolves to the right IP address. Let us fix this in the /etc/hosts file or DNS database (whatever method you are using for name resolution). Once we have done this, we can go ahead and change the SDR information as shown in Figure 130 on page 220.



```
                          SP Ethernet Information

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

                                                        [Entry Fields]
  Start Frame                                        []                       #
  Start Slot                                         []                       #
  Node Count                                         []                       #

  OR

  Node Group                                         []                       +

  OR

  Node List                                          [5]

* Starting Node's en0 Hostname or IP Address         [192.168.5.5]
* Netmask                                            [255.255.255.0]
* Default Route Hostname or IP Address               [192.168.5.150]
  Ethernet Adapter Type                               bnc                     +
  Duplex                                              half                    +
  Ethernet Speed                                      10                      +
  Skip IP Addresses for Unused Slots?                 no                      +




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
```

Figure 130. SMIT panel for SDR information

Once the SDR information has been entered, we check the Adapter class as follows:

```
 [sp5en0:/]# SDRGetObjects Adapter node_number adapter_type netaddr
node_number  adapter_type netaddr
          1 en0          192.168.5.1
          5 en0          192.168.5.5
          9 en0          192.168.5.9
         13 en0          192.168.5.13
          1 css0         192.168.15.1
          5 css0         192.168.15.5
          9 css0         192.168.15.9
         13 css0         192.168.15.13
```

This looks much better. So, the only thing to do now is to update the machines.lst file in the SDR. To update the file in the SDR, we just need to stop and restart the Topology Services subsystem on the CWS. We use the following commands:

```
stopsrc -s hats.sp5en0
startsrc -s hats.sp5en0
```

After these two commands are executed, we verify the content of the machines.lst file in the SDR by using the `SDRRetrieveFile` command as follows:

```
[sp5en0:/tmp]# SDRRetrieveFile hats.machines.lst machines.lst
[sp5en0:/tmp]# cat machines.lst
*InstanceNumber=944063551
*configId=2225957613
*FileVersion=1
*!TS_realm=PSSP
TS_Frequency=1
TS_Sensitivity=4
TS_FixedPriority=38
TS_LogLength=5000
*!TS_PinText
Network Name SPether
Network Type ether
*
*Node Type Address
    0 en0 192.168.5.150
    1 en0  192.168.5.1
    5 en0  192.168.5.5
    9 en0  192.168.5.9
   13 en0  192.168.5.13
Network Name SPswitch
Network Type hps
*
*Node Type Address
    1 css0 192.168.15.1
    5 css0 192.168.15.5
    9 css0 192.168.15.9
   13 css0 192.168.15.13
```

The file now looks correct; so, we restart the Topology Services on node 5 and verify that it is working properly as follows:

```
[sp5n05:/]# stopsrc -s hats
0513-044 The hats Subsystem was requested to stop.
[sp5n05:/]# startsrc -s hats
0513-059 The hats Subsystem has been started. Subsystem PID is 18604.
[sp5n05:/]# lssrc -ls hats
Subsystem         Group            PID     Status
 hats             hats             18604   active
 Network Name   Indx Defd Mbrs St Adapter ID      Group ID
 SPether        [ 0]    5    1  U 192.168.5.5     192.168.5.5
 SPether        [ 0]              0x40454536      0x40454536
 HB Interval = 1 secs. Sensitivity = 4 missed beats
 SPswitch       [ 1]    4    1  S 192.168.15.5    192.168.15.5
 SPswitch       [ 1]              0x40454537      0x40454541
 HB Interval = 1 secs. Sensitivity = 4 missed beats
   1 locally connected Client with PID:
 hagsd( 18080)
   Configuration Instance = 944063551
   Default: HB Interval = 1 secs. Sensitivity = 4 missed beats
   CWS = 192.168.5.150
 Daemon is simulating down for 67 sec. Time remaining: 57 sec.
```

The daemon has started; however, the SPether ring is still unstable. We have to wait at least 57 seconds (last line of the `lssrc` output) and check again. Time is up; so, we check again:

```
[sp5n05:/]# lssrc -ls hats
Subsystem         Group            PID     Status
 hats             hats             18604   active
 Network Name   Indx Defd Mbrs St Adapter ID      Group ID
 SPether        [ 0]    5    5  S 192.168.5.5     192.168.5.150
 SPether        [ 0]              0x40454536      0x4045457c
 HB Interval = 1 secs. Sensitivity = 4 missed beats
 SPswitch       [ 1]    4    0  D
 HB Interval = 1 secs. Sensitivity = 4 missed beats
   1 locally connected Client with PID:
 hagsd( 18080)
   Configuration Instance = 944063551
   Default: HB Interval = 1 secs. Sensitivity = 4 missed beats
   CWS = 192.168.5.150
```

Now, Topology Services on node 5 seems to be working fine. We do not worry about the SPswitch ring at this time.

Let us verify host responds for node 5 with the following command from the CWS:

```
[sp5en0:/tmp]# SDRGetObjects host_responds
node_number  host_responds
          1            1
          5            0
          9            1
         13            1
```

Node 5 still does not have host responds. However, we are now sure that this is not a Topology Services problem. Let us see what Group Services has to say about this.

4.2 Group Services

Group Services (GS) is the next layer in the RSCT architecture. It is a distributed subsystem that provides services for synchronization and coordination to clients. Clients connect to Group Services through an API, the Group Services API (GSAPI).

Although you may not work directly with Group Services, it is a critical subsystem that must be up and running in order to get other subsystems to work properly. Subsystems that are clients of GS include: Event Manager, Recoverable Virtual Shared Disk (RVSD), General Parallel File System (GPFS), and indirectly through these subsystems, others will also fail, such as SP Perspectives, Host Responds, and Problem Management.

As you can see, getting Group Services up and running is a must for any RS/6000 SP installation. But, before we start debugging problems with GS, let us define some concepts in order to better understand how Group Services works.

4.2.1 Group Services concepts

Group Services is a middleware. It runs on top of Topology Services (it is a client for Topology Services), and it provides services to upper layers (Event Manager among others). It is made of a daemon (hags daemon, hagsd) and the GSAPI library (libha_gs.a and libha_gs_r.a). The interaction with Group Services is through the GSAPI. There is no command line interface or graphical user interface to Group Services. The GSAPI allows local connections only (through UNIX Domain Sockets).

Figure 131 on page 223 depicts Group Services from a logical point of view.

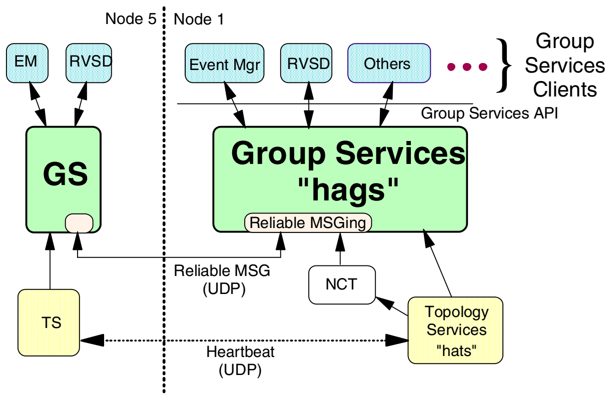


Figure 131. Group Services—Overview

Each hags daemon (hagsd) subscribes to Topology Services locally for connectivity and availability information. The communication between Group Services daemons is through the Reliable Messaging Library.

Group Services is a partition-sensitive subsystem. So, the daemons will communicate within the system partition boundaries. In Group Services, and in RSCT terminology in general, the system partition is called a *domain*. All the RSCT daemons execute and communicate within a domain. Nodes may belong to more than one domain (the CWS belong to all the partitions), and in this case, they run several instances of the RSCT daemons, one per domain.

For example, a node may belong to an SP domain (system partition) and to an HACMP domain (for HACMP 4.3.1 and PSSP 3.1 or later).

The key concept in Group Services is the *group*. A group is a logical association of Group Services clients. Clients will form groups to synchronize and coordinate activities. Group Services provides the plumbing underneath.

Groups have the following characteristics:

- Groups have names (any non-null string).
- Groups have a membership list (a list of the clients belonging to this group).
- Groups have state information (a common state value visible to all members).

Group Services (through the GSAPI) provides mechanisms to create, join, and subscribe to groups, as well as to modify group state information.

Clients using the GSAPI can create or join existing groups. They can also subscribe to existing groups, which means that they are not part of the group (cannot initiate actions to change group information), but they get notifications for changes in the group information. You may think of *subscribers* as read-only members. Members of a group (clients that have join the group) are called *providers*.

Only providers of a group can initiate changes to the group information via *protocols*. Protocols are actions executed by members of a group in order to change the group information.

A key feature of Group Services is to provide a single group namespace across the cluster. Actually, each system partition (domain) is an isolated namespace that defines the complete scope of Group Services. All references to a specific group name within a group namespace will result in those references being directed to the same group. Any reference to that name in another group namespace (domain) refers to a completely independent group.

In order to provide this single group namespace, Group Services defines the concept of a *nameserver* (this is not DNS). A nameserver (NS) is who keeps track of all the group names and membership information across a partition (domain). One of the Group Services daemons is selected as the nameserver at initialization time. The selection method for the nameserver is quite simple:

- When a Group Services daemon starts, it connects to Topology Services, and it requests availability and connectivity information. Topology Services sends back a list of "up" nodes (nodes running Topology Services).
- Based on the input from Topology Services, each daemon finds the lowest-numbered running node in the partition (domain). It then compares it to its own node number and performs one of the following:
  - If the daemon's own node is the lowest-numbered node, it waits for all the other running nodes to nominate it as the Group Leader nameserver.
  - If the daemon's own node is not the lowest-numbered node, it starts sending nomination messages to the lowest-numbered node every five seconds.
- Once all running nodes have nominated the Group Services nameserver-to-be, and a variable-length timer has expired, the nominee sends an insert message to the nodes. All nodes must acknowledge this message. When they have done so, the nominee becomes the established Group Services nameserver, and it sends a commit message to all the nodes.
- At this point, the Group Services domain is established, and requests by clients to join or subscribe to groups are processed.

Note, that this is the case when all nodes are being booted simultaneously, such as at initial system power-on. Often, however, a Group Services daemon is already running in at least one node (for example, the CWS), and the Group Services domain is already established, thus, making the CWS the Group Services nameserver. In this case, the Group Services nameserver will execute insert protocols when nodes start running. The list will be filled on a first-come, first-served basis.

4.2.1.1 Group Services protocols

To understand Group Services protocols, let us use the following example.

Assume that you have a distributed application that uses Group Services for synchronizing and coordinate activities and state data within the application. Let us call this application A.

Application A will run in all the nodes (see Figure 132 on page 226). The application will use a group named Group_A. The first daemon of application A to start will create the group if it does not exists.

Let us assume we start application A in node 5 (it could have been any node, we just picked one).

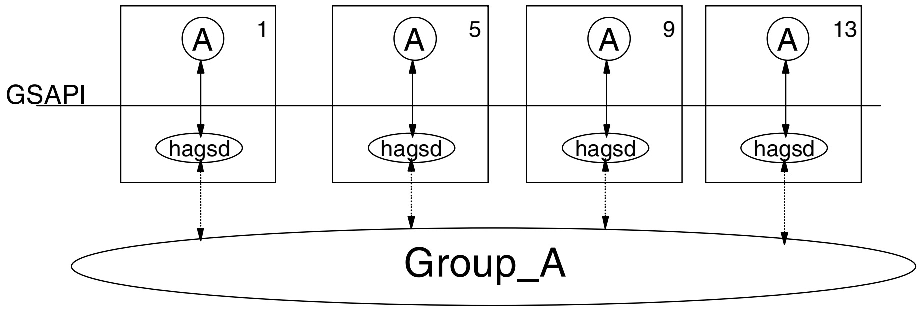


Figure 132. Group Services—An example

The following are the actions carried out by application A and Group Services:

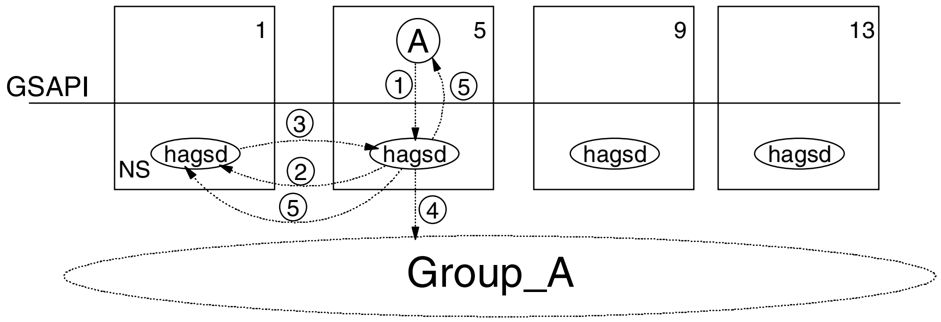


Figure 133. Group Services—What is going on?

1. Application A on node 5 connects to Group Services on that node and makes a join request to Group_A.
2. The Group Services daemon on node 5 does not have a group called Group_A in its tables; so, it contacts the nameserver for more information.
3. Assuming the nameserver is node 1 (it could be any node), the Group Services daemon running there will send back a reply to node 5 stating that no group called Group_A has been created.
4. The Group Services daemon on node 5 receives the message and proceeds to create the group according to the parameters passed by application A's GSAPI call.
5. The Group Services daemon on node 5 informs the nameserver about the newly created group with application A's daemon on node 5 being the

"owner" of the group. Then, the GSAPI library returns the call to application A.

At this point, application A is running on node 5 with a group called Group_A that is only known by the hags daemon on node 5 and the Group Services nameserver. The next step is to start a new application A daemon on a different node (that is the whole idea of a distributed application).

Figure 134 on page 227 depicts the actions carried out when application A starts on another node.

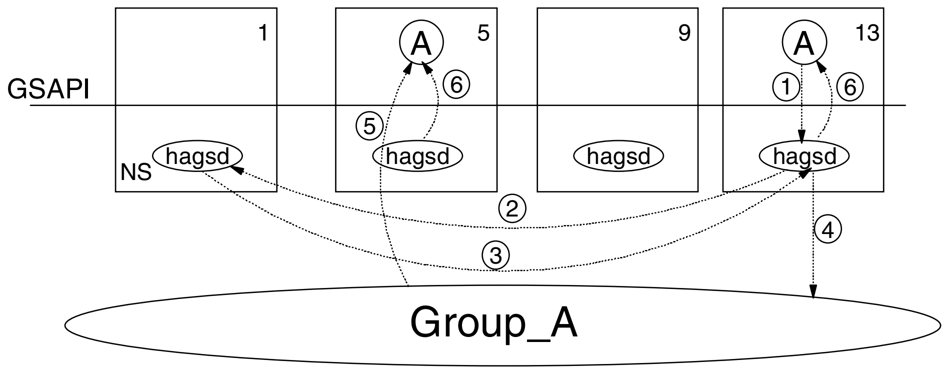


Figure 134. Group Services—Someone is knocking at the door

These are the actions, assuming we start application A on node 13:

1. Application A's daemon on node 13 connects to Group Services and submits a *Join Request* for group Group_A.
2. The Group Services daemon on node 13 does not have a group called Group_A in its tables. It contacts the Group Services nameserver for more information.
3. The nameserver finds Group_A defined and replies to the hags daemon on node 13 with the information about the group including the membership list.
4. The Group Services daemon on node 13 then makes a *proposal* to all members of Group_A for a *Join Protocol*.
5. All members of Group_A (in this case, the daemon running on node 5 only) get the request. Then, all providers have to vote.

   Voting may be carried out in multiple phases. The most common scenario is a 2-phase protocol where members get the request, and they vote *Approve or Reject*. All members have to vote Approve for a proposal to be accepted. A Reject ends the protocol. Members may also vote *Continue,* which causes to initiate a new phase (this is where a multiphase protocol kicks-off).

6. Once the proposal gets approved, Group Services notifies all providers and updates the membership list and state value accordingly. In our case, application A’s daemon running on node 13 has been approved in the group by application A’s daemon running on node 5 and puts it into the membership list for Group_A.

4.2.2 Group Services initialization

Normally, the Group Services daemon is 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.

The SRC subsystem starts the hags script as you can see from the SRC subsystem definition:

```
sp5en0:/>odmget -q subsysname=hags.sp5en0 SRCsubsys

SRCsubsys:
        subsysname = "hags.sp5en0"
        synonym = ""
        cmdargs = "PSSP -p sp5en0 -d /var/ha/log/hags.default.sp5en0"
        path = "/usr/sbin/rsct/bin/hags"
        uid = 0
        auditid = 0
        standin = "/dev/console"
        standout = "/var/ha/log/hags.default.sp5en0"
        standerr = "/var/ha/log/hags.default.sp5en0"
        action = 1
        multi = 0
        contact = 3
        svrkey = 0
        svrmtype = 0
        priority = 20
        signorm = 0
        sigforce = 0
        display = 1
        waittime = 30
        grpname = "hags"
```

The hags script will start the hags daemon after checking some disk space used by the log file (see 4.2.3, “Group Services logs” on page 230 for details). During its initialization, the Group Services daemon performs the following steps:

1. It gets the number of the node on which it is running using the `/usr/lpp/ssp/install/bin/node_number` command. Node 0 is the CWS.
2. It tries to connect to the Topology Services subsystem. If the connection cannot be established because the Topology Services subsystem is not running, it is scheduled to be retried every 20 seconds. This continues until the connection to Topology Services is established. Until the connection is established, the Group Services daemon writes an AIX error log entry periodically, and no clients may connect to the Group Services subsystem.
3. It 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.
4. It establishes the Group Services domain, which is the set of nodes within the SP system partition in which a Group Services daemon is executing.

   At this point, one of the GS daemons establishes itself as the GS nameserver.

   Until the domain is established, no GS client requests to join or subscribe to groups are processed.
5. It enters the main control loop.

   In this loop, the Group Services daemon waits for requests from GS clients, messages from other Group Services daemons, messages from the Topology Services subsystem, and requests from the SRC for status.

To ensure that only one node becomes a GS nameserver, Group Services uses the following protocol:

1. When each daemon is connected to the Topology Services subsystem, it waits for Topology Services to tell it which nodes are currently running in this system partition.
2. Based on the input from Topology Services, each daemon finds the lowest-numbered running node in the domain. The daemon compares its own node number to the lowest-numbered node and performs one of the following:
   - If the daemon's node is the lowest-numbered node, the daemon waits for all other running nodes to nominate it as the GS nameserver.
   - If the daemon's node is not the lowest-numbered node, it sends nomination messages to the lowest-numbered node periodically, initially every five seconds.

3. Once all running nodes have nominated the GS nameserver-to-be and a coronation timer (about 20 seconds) has expired, the nominee sends an insert message to the nodes. All nodes must acknowledge this message. When they do, the nominee becomes the established GS nameserver, and it sends a commit message to all of the nodes.
4. At this point, the Group Services domain is established, and requests by clients to join or subscribe to groups are processed.

Note, that this description is in effect when all nodes are being booted simultaneously, such as at initial system power-on. It is often the case, however, that a Group Services daemon is already running on at least one node (for example, the CWS) and is already established as the domain's GS nameserver. In this case, the GS nameserver waits only for Topology Services to identify the newly running nodes. The GS nameserver will then send the newly running nodes proclaim messages that direct the nodes to nominate it as nameserver. Once those nodes nominate the GS nameserver, the GS nameserver simply executes one or more insert protocols to insert the newly-running nodes into the domain.

4.2.3 Group Services logs

Using the same logic as in Topology Services, Group Services keeps separate log files in the /var/ha/log directory. There are three log files that can be found in the log directory.

The hags.<syspar> file, which contains the standard output and standard error of the hags script, are started from the SRC subsystem. This file is empty most of the time, and when it is not, then there lays your problem. Open it up and see what happens to the hags script. Usually, it is a problem with files or access to SDR information.

Every time the hags script runs, it checks the amount of disk space being used by the Group Services log files. It if exceeds 5MB, it will delete any core file from the run-time directory (/var/ha/run/hags.<syspar>) and any old log file until the disk space being used is less than the maximum of 5 MB (by default).

The second log file is called hags.default.*syspar_nodenum_instnum* where:

nodenum	is the node number on which the daemon is running.
instnum	is the instance number of the daemon.
syspar	is the name of the system partition to which the node belongs.

This file contains trace output from the initial start-up of the daemon.

The third log file is the daemon's log file called hags_*nodenum_instnum.syspar*. This is the log file that contains current information about all the Group Services daemon's activities.

The Group Services daemon limits the size of this to a pre-established number of lines (by default, 5,000 lines). When the limit is reached, the daemon appends the string .bak to the name of the current log file and begins a new log. If a .bak version already exists, it is removed before the current log is renamed.

Most of the time you will not be dealing with these log files though. Group Services provides an excellent set of "undocumented" commands that will help you figure out what is happening inside Group Services. Keep in mind though that these commands will give you instant information, but the log file will give you the history of events.

4.2.4 Group Services commands

Opposite of Topology Services, Group Services provides most of its internal information and behavior through a handful of commands located in /usr/sbin/rsct/bin although the very first command you may want to try is the `lssrc` command with the long list option:

```
[sp5en0:/]# lssrc -ls hags.sp5en0
+1  Subsystem         Group            PID     Status
+2   hags.sp5en0      hags             17524   active
+3   2 locally-connected clients.  Their PIDs:
+4   23510 18992
+5   HA Group Services domain information:
+6   Domain established by node 1.
+7   Number of groups known locally: 2
+8                       Number of   Number of local
+9   Group name          providers   providers/subscribers
+10   cssMembership           4           0           1
+11   ha_em_peers             5           1           0
```

This command displays most of the information you will need for problem determination. The description is as follow:

Lines 1 and 2: This is the standard output from the `lssrc` command without the `-l` option. It tells you whether the subsystem is active or not.

Lines 3 and 4: These lines tell you whether clients are connected locally (through a Unix Domain Socket), and if they are, it gives you the PIDs or process identifiers.

Lines 5 and 6: These lines tell you if the domain has been established. In order words, if Group Services has been able to initialize, and if true, then Line 6 gives you the node number where the Group Services nameserver is running. If Group Services has not been able to initialize, Line 6 will show a "Domain not establish" message, which means that Group Services is not working. Clients who depend on Group Services (Event Management, GPFS, and others) will fail to initialize.

Lines 7 to 11: Finally, these lines give you detailed information about the groups "known" by the Group Services daemon on this machine (where the `lssrc` command is being run). Remember that only groups with local providers or subscribers are "known" to the local Group Services daemon. In this case, the *cssMembership* group has one local subscriber, and the *ha_em_peers* group has one local provider, and they are the same (but you cannot tell from this output).

The `lssrc` command gives you general information about Group Services. If everything is fine, this is all that you need. When things go wrong, you may want to try other commands.

4.2.4.1 The hagsns command

The `hagsns` command provides an SRC-based interface to interrogate the GS daemon as to its complete GS nameserver status. It may vary significantly depending upon the status of the various GS daemons across a system partition.

Assume we have a system partition with four nodes (and the CWS): Partition (CWS) name: sp5en0 (in the output this appears as node_number 0) nodes in partition: 1 (sp5n01), 5 (sp5n05), 9 (sp5n09), and 13 (sp5n13). The following is the output from the `hagsns` command when run on the CWS:

```
[sp5en0:/]# /usr/sbin/rsct/bin/hagsns -s hags.sp5en0
+1  We are: 0.2 pid: 17524 domaindId = 1.0 noNS = 0 inRecovery = 0 CodeLevel = RSCT 1.0
+2  NS::ENsState(6):kCertain protocolInProgress = NS::ENsProtocol(0):kNoProtocol
+3  outstandingBroadcast = NS::ENsBroadcast(0):kNoBcast
+4  Process started on Jan 26 16:09:31, (19d 22:30:45) ago.  HB connection took (0:1:24).
+5  Initial NS certainty on Jan 26 16:12:22, (19d 22:27:54) ago, taking (0:1:26).
+6  Our current epoch of certainty started on Jan 26 16:12:22, (19d 22:27:54) ago.
+7  5 UP nodes: 0 1 5 9 13
```

Line 1: This line specifies that the domain is established and there is a nameserver (noNS=0). If the domain is not established, then the `noNS` flag would be set to one, and the inRecovery bit should be set to one. It means that Group Services is recovering or initilizing; so, you should give it a little bit more time. If the inRecovery bit is set to zero when no domain has been establish, then you have a problem. Refer to 4.2.6, “Case study” on page 245 for an example on how to debug such a problem.

Lines 2 and 3 show that nothing is going on. The domain is established, and there is no protocol or broadcast message in progress.

Lines 4 to 6 give statistic information about the time when the domain was established as well as information about the time it took GS to contact and connect to Topology Services.

Line 7 lists all the nodes reported as up and running from Topology Services. All those nodes should have the Group Services daemon up and running before the domain can be established, hence, Group Services can be operational.

If the same command is run on the nameserver node (which is node 1 in this case), the output will contain some additional information:

```
[sp5n01:/usr/sbin/rsct/bin]# ./hagsns -s hags
We are: 1.0 pid: 15244 domaindId = 1.0 noNS = 0 inRecovery = 0 CodeLevel = RSCT 1.0
NS::ENsState(7):kBecomeNS protocolInProgress = NS::ENsProtocol(0):kNoProtocol
outstandingBroadcast = NS::ENsBroadcast(0):kNoBcast
Process started on Jan 25 10:03:10, (22d 7:29:26) ago.  HB connection took (0:14:51).
Initial NS certainty on Jan 25 10:18:28, (22d 7:14:8) ago, taking (0:0:26).
Our current epoch of certainty started on Jan 26 14:54:33, (21d 2:38:3) ago.
5 UP nodes: 0 1 5 9 13
1.1 ha_em_peers: GL: 1 seqNum: 23 theIPS: 1 13 9 5 0 lookupQ:
2.1 cssMembership: GL: 1 seqNum: 28 theIPS: 1 13 9 5 0 lookupQ:
```

By running the command on the GS nameserver, you get additional information regarding all groups "known" by Group Services.

```
1.1 ha_em_peers: GL: 1 seqNum: 23 theIPS: 1 13 9 5 0 lookupQ:
2.1 cssMembership: GL: 1 seqNum: 28 theIPS: 1 13 9 5 0 lookupQ:
```

This is the list of client groups that have providers join them in this Group Services domain. The output is as follows:

<group identifier>	An internal to Group Services identifier to each group, for example, 1.1 or 2.1.
<group name>	The external name given by the Group Services clients when they created the group, for example, ha_em_peers (Event Management group) or cssMembership (internal Switch group).
GL	An internal to Group Services "Group Leader"[3] used to control the functioning of the group. The

	node_number given is the node that makes decisions about which protocol to run in the group.
seqNum	An internal to Group Services "sequence number" for messages within each group. Each group maintains its own sequence of messages broadcasts. The value shown here is the group's sequence number when it last had a group owner. A group owner updates this field at the GS nameserver when it initially becomes group owner and periodically thereafter (about every 20,000 group messages it sends).
theIPS	An internal to Group Services "list" of nodes that have expressed an interest in a client group. Normally this is because a client process has asked to join or subscribe to a group. A node expresses interest in a group by sending a "lookup" request to the domain's Group Services nameserver and is placed in the *theIPS* list when the Group Services nameserver sends out a response to the lookup request.
lookupQ	An internal to Group Services "list" of nodes that have been sent in a lookup request, and the Group Services nameserver has not yet sent out a response to the lookup. There is only a lookup queue when there is no group owner for the group, which can happen when the request is the first being made for a particular group, or when the group owner's node fails, and a group owner's recovery is in progress. If you do see a queue here that lasts a while, check the group status on the nodes listed in the *theIPS* using the `lssrc -ls hags` command, for example.

4.2.4.2 The hagsmg command

The `hagsmg` command provides the list of nodes in each "meta-group". A meta-group can be thought of as a "shadow" to each client group, with two exceptions:

- The "ZtheNameServerXY" meta-group: This is the "group" used by the GS nameserver itself to manage the domain and should include all active nodes in the system partition.

[3] In this document, we refer to it as a Group Owner to avoid conflicts with the Group Leader definition in Topology Services.

- The "theGROVELgroup" meta-group: This "group" is an internal use only group that never contains any nodes.

```
[sp5en0:/]# /usr/sbin/rsct/bin/hagsmg -s hags.sp5en0
+1  2.1 cssMembership: 1 5 0 13 9
+2  1.1 ha_em_peers: 1 5 0 13 9
+3  0.Nil ZtheNameServerXY: 1.Nil 5.Nil 0.2 13.2 9.1
+4  0.Nil  theGROVELgroup:
```

Similar to hagsns, the first column provides the internal "group_identifier" for each meta-group. The group_name is the external group name with the two exceptions as noted above.

Following the group name is the list of nodes that have been inserted into each meta-group. The first node in the list is the "group owner" for each meta-group. This node controls the operation of the group by choosing the next protocol to be executed.

The list of nodes given by hagsmg should track closely with the list of nodes given by the GS nameserver node response to hagsns, although the order in which the nodes are listed may vary. Also, due to timing differences normal in a distributed system, the group owner indicated by the hagsns output and the group owner indicated by hagsmg may differ, indicating that an old owner had failed, and that Group Services must recover to a new owner for that group.

The nodes are listed in group leader takeover order, which reflects the order in which they joined the group. If node 1 fails, node 5 will takeover as GS nameserver, the domain group leader, as well as group leader for the ha_em_peers group.

4.2.4.3 The hagspbs command

The hagspbs command lists the status of the "pbs" sub-component, an internal component of Group Services. It stands for "Phoenix Broadcast Services". As sample output, the following is from node 1 taken when the domain is established, and there is no activity:

```
[sp5n01:/usr/sbin/rsct/bin]# ./hagspbs -s hags
2.1 cssMembership: HWM 55 LWM 55 weAreTheGL
  pendingAckCount=0 kNotExpectingAcks  pendingRecoverCount=0
        1: HWM=0: lastType1=Nil
        5: HWM=55: lastType1=54
        0: HWM=55: lastType1=54
        13: HWM=55: lastType1=54
        9: HWM=55: lastType1=54
1.1 ha_em_peers: HWM 58 LWM 58 weAreTheGL
  pendingAckCount=0 kNotExpectingAcks  pendingRecoverCount=0
        1: HWM=1: lastType1=Nil
        5: HWM=58: lastType1=57
        0: HWM=58: lastType1=57
        13: HWM=58: lastType1=57
        9: HWM=58: lastType1=57
0.Nil ZtheNameServerXY: HWM 28 LWM 28 weAreTheGL
```

```
  pendingAckCount=0 kNotExpectingAcks  pendingRecoverCount=0
        1.0 kUp needDSM: HWM=1: lastType1=Nil
        5.1 kUp: HWM=28: lastType1=13
        0.2 kUp: HWM=28: lastType1=Nil
        13.2 kUp: HWM=28: lastType1=Nil
        9.1 kUp: HWM=28: lastType1=Nil
0.Nil  theGROVELgroup: HWM Nil LWM Nil
  pendingAckCount=0 kNotExpectingAcks  pendingRecoverCount=0
```

The output lists each group_identifier and group_name. It then lists the "broadcast sequence numbers" for each group, which consists of a HighWaterMark (HWM) and LowWaterMark (LWM) for each group. For example:

```
1.1 ha_em_peers: HWM 58 LWM 58 weAreTheGL
```

In general, these sequence numbers should match for each group on each node that is inserted into that group. Note, however, that due to delays of messages getting across the network(s), some nodes may not have yet seen all messages. Assuming the system partition (domain) eventually quiets down, all nodes should eventually catch up. Note that these sequence numbers will NOT normally match those displayed by the `hagsns` command.

The other lines of output have too many possible permutations to go into now. Suffice it to say that if `lssrc` and or `hagsns` indicate that the Group Services domain is not yet formed or is recovering, you can see some interesting output here. Also, if one or more groups are running a series of protocols (due to node boots and/or failures), you can see a large amount of output here. If a group is hung, or the domain seems hung with a protocol in progress, run this command on the group's owner's node to see which node(s) it may be waiting for.

4.2.4.4 The hagsgr command

All that you want to know about groups is in this command. It has two options, a verbose and a extremely verbose output. The first one is as follow:

```
[sp5en0:/]# /usr/sbin/rsct/bin/hagsgr -s hags.sp5en0
+1    Number of: groups: 5
+2  Group slot # [0] Group name[HostMembership] source[no] target[no] group state[Idle |]
+3  Providers[[0/1][0/5][0/0][0/13][0/9]]
+4  Local subscribers[[10/0]]
+5
+6  Group slot # [1] Group name[ha_em_peers] source[no] target[no] group state[Inserted
|Idle |]
+7  Providers[[1/1][1/5][1/0][1/13][1/9]]
+8  Local subscribers[]
+9
+10  Group slot # [2] Group name[cssRawMembership] source[no] target[no] group state[Idle
|]
+11  Providers[]
+12  Local subscribers[]
+13
+14  Group slot # [3] Group name[enMembership] source[no] target[no] group state[Idle |]
```

```
+15  Providers[[0/0][0/1][0/5][0/13][0/9]]
+16  Local subscribers[[10/0]]
+17
+18  Group slot # [4] Group name[cssMembership] source[no] target[no] group
state[Inserted |Idle |]
+19  Providers[[0/1][0/5][0/13][0/9]]
+20  Local subscribers[[10/0]]
```

The format for Providers listed is the following:

[provider_instance_number / provider_node_number]

- provider_instance_number: This is specified by the client process when it joins a group (via the ha_gs_join() interface).
- provider_node_number: This is the node_number on which the client process is executing.

  Therefore, in group "ha_em_peers", the list: Providers[[1/1][1/5][1/0][1/13][1/9]] specifies that there are five providers joined to the group, and all of them have a provider_instance_number of "1", and they are executing on nodes 1, 5, 9, 13, and 0. If the provider_identifier is unique enough, you may be able to use it to match it to the actual client process using the `hagscl` output. However, this may not be sufficient, and you may need to use the `hagsgr` "long" option.

  The providers are shown in the order in which they joined the group; therefore, the "oldest" provider is listed first, and the "youngest" last.

The format for Subscribers is similar but a bit different:

[socket_file_descriptor / subscriber_node_number]

- socket_file_descriptor is the GS daemon's socket file descriptor that connects it to the client process that subscribed to the group (via ha_gs_subscribe()). Note, that you can use this along with the output from hagscl to determine the pid of the subscriber since a node will display *only* the local subscribers (those actually executing on the node on which the `hagsgr` command is executed).
- subscriber_node_number is the node on which the subscriber is executing, which is always the local node.

The "group state" field indicates the status of that group, and it is a collection of separate states ORed together:

a. If it includes "Not Inserted", then that group is not currently active on this node and should have no providers or subscribers.
b. If it includes "Insert Pending", then it is attempting to become inserted into the group. It has sent the GS nameserver a "lookup" and is awaiting the

response, or, the response is received, and it is awaiting to be inserted into the meta-group. You have to use `hagsns`, `hagsmg`, and/or `hagspbs` to determine the exact step. This should normally be a temporary state.

c. If it includes "Inserted", then that group is currently active on this node, and it may have providers and/or subscribers.

d. If "Inserted" is included, then one of the following:

  - "Idle" indicates the group is not currently running a protocol.
  - "Running Protocol" is simple enough. The group is running a protocol. The `hagsgr` "long" option will allow you to see what this protocol is.
  - "Needs Priming" indicates a node is attempting to become fully active in a group and is waiting to find out the current state (membership and group state value) of the group. This should normally be temporary.

e. Other temporary states include "Waiting for BroadcastSent" and "Resending Requests". The former indicates the node is in the midst of sending a broadcast, the latter that the group's owner has failed, and we are recovering to a new group owner.

It is also possible to specify `hagsgr` for a specific group. The long list option of the `hagsgr` command is extremely verbose. Most of the information displayed with this option is too deep and useful to developers only. However, if you ever are stuck in with a Group Services problem where all the "conventional" methods fail, this command can shed some light into your problem. As an example only, the following output is partial (only the ha_em_peers group information is displayed):

```
[sp5en0:/]# /usr/sbin/rsct/bin/hagsgr -a ha_em_peers -ls hags.sp5en0
+1    Number of: groups: 5
+2  Information for SGroup: Group name[ha_em_peers]
+3  I am *not* Group Leader!  Created by join request.
+4  group state[Inserted |Idle |]
+5  ProtocolToken[71/141]
+6   counts: (prov/localprov/subs) [4/1/0]
+7   delayed join count [0]
+8   protocol counts:  received [0]
+9    dropped(total/DaemonMsg/ProtMgr) [0/0/0]
+10   message counts:  current queued [0] future queued/cumulative [0/0]
+11  Group attributes[{group name:  value:
+12  ha_em_peers
+13  }
+14   batching[Batch both] membership phases[N phases] reflection phases[1 phase]
+15  default vote[Reject] merge control[Don't care]
+16  membership time limit[60] reflection time limit[30]
+17  client version[0] version[0] size[40]
+18
+19  Group state value: [min/max lengths (1/256)] actual length[34]  value:
+20  0x39313732 0x37363239 0x342c3539 0x34333034 0x30302c30 0x004e4f53 0x45435355
0x50504f5
2 0x5400
+21  917276294,59430400,0.NOSECSUPPORT.
```

```
+22
+23  --------------------
+24   Provider list:
+25  SProvider(ProviderId[1/5]conditionalListPosition[-1]
+26  SVSuppMember:  token[0] status[MemberIn ]
+27   supp ptr: 0x0 group ptr: 0x3009c378 groupListPosition: 0 nodeListPosition: 0 Need
Vot
e/Voted Yet[0/0]
+28   [NOT votingParticipant])[end SProvider]
+29
+30  SProvider(ProviderId[1/13]conditionalListPosition[-1]
+31  SVSuppMember:  token[0] status[MemberIn ]
+32   supp ptr: 0x0 group ptr: 0x3009c378 groupListPosition: 1 nodeListPosition: 0 Need
Vot
e/Voted Yet[0/0]
+33   [NOT votingParticipant])[end SProvider]
+34
+35  SProvider(ProviderId[1/9]conditionalListPosition[-1]
+36  SVSuppMember:  token[0] status[MemberIn ]
+37   supp ptr: 0x0 group ptr: 0x3009c378 groupListPosition: 2 nodeListPosition: 0 Need
Vot
e/Voted Yet[0/0]
+38   [NOT votingParticipant])[end SProvider]
+39
+40  SProvider(ProviderId[1/0]conditionalListPosition[-1]
+41  SVSuppMember: [owned by:Client: socketFd[12] pid[21944]] token[0] status[MemberIn ]
na
me[SMemberName: (min/max)length: (1/16)11 value:
+42  ha_em_peers
+43  ]
+44   supp ptr: 0x3009bc98 group ptr: 0x3009c378 groupListPosition: 3 nodeListPosition: 0
N
eed Vote/Voted Yet[0/1]
+45  0x300bc648 [votingParticipant])[end SProvider]
+46
+47  --------------------
+48
+49   Local provider list:
+50  SProvider(ProviderId[1/0]conditionalListPosition[-1]
+51  SVSuppMember: [owned by:Client: socketFd[12] pid[21944]] token[0] status[MemberIn ]
na
me[SMemberName: (min/max)length: (1/16)11 value:
+52  ha_em_peers
+53  ]
+54   supp ptr: 0x3009bc98 group ptr: 0x3009c378 groupListPosition: 3 nodeListPosition: 0
N
eed Vote/Voted Yet[0/1]
+55  0x300bc648 [votingParticipant])[end SProvider]
+56
+57  --------------------
+58
+59   Local subscriber list:
+60  --------------------
+61
+62  Protocol Manager summary information:
+63  Current count: 0
+64  total count: executed/approved/rejected[6/6/0]
+65  failure count: executed/approved/rejected(explicit/implicit)[3/3/0(0/0)]
+66  join count: executed/approved/rejected[3/3/0]
+67  expel count: executed/approved/rejected[0/0/0]
+68  attribute change count: executed/approved/rejected[0/0/0]
+69  leave count: executed/approved/rejected[0/0/0]
+70  state change count: executed/approved/rejected[0/0/0]
```

```
+71  PBM count: executed/approved/rejected[0/0/0]
+72  source reflection count: executed/approved/rejected[0/0/0]
+73  subscription count: executed/approved/rejected[0/0/0]
+74  announcement count: executed/approved/rejected[0/0/0]
+75  --------------------
+76  No transient protocol
+77  --------------------
+78  No currently executing protocol
+79  --------------------
+80  Unsent queue:[No entries]
+81  --------------------
+82  Sent queue:[No entries]
+83  --------------------
+84  Failure queue:[No entries]
+85  --------------------
+86  Join queue:[No entries]
+87  --------------------
+88  Subscribe queue:[No entries]
+89  --------------------
+90  Announcement queue:[No entries]
```

Important points to remember about the output:

a. An indication if this node is, or is not, the owner of the group.

b. The count of providers/local providers and subscribers is given.

c. The group's "protocol token" is shown. This token is used to keep track of "group time" to ensure that all nodes have seen and processed the latest protocol.

d. The group's attributes are listed as they were specified on the ha_gs_join() calls used by the providers to join the group.

e. The group's "group state value" is listed in two formats:

   - Hexadecimal memory dump.
   - Printable characters display normally, non-printable characters displayed as periods (".").

f. Each provider is shown. Local providers (those executing on the node on which the command is executed) are shown with the details of their client process, therefore, allowing you to determine the actual process joined to the group:

   ```
   SProvider(ProviderId[1/0]conditionalListPosition[-1]
   SVSuppMember: [owned by:Client: socketFd[12] pid[21944]] token[0] status[MemberIn ]
   ```

   For remote providers, the listing is simply:

   ```
   SVSuppMember:  token[0] status[MemberIn ]
   ```

   Note, that with 100+ providers, this portion of the output gets rather large. The providers are shown in the order in which they joined the group; therefore, the "oldest" provider is listed first and the "youngest" last.

g. The local providers and local subscribers are then shown.

h. Protocol count summary information is shown for the group. Note, that the summary counts indicate only those protocols for which this node has participated in the group and may, therefore, differ greatly from nodes that have been part of the group for significantly longer or shorter periods of time.

i. Any executing and queued protocols are shown.

Note, that it is often difficult to "capture" a group in the middle of a protocol when everything is functioning properly, as the groups often execute quite quickly.

4.2.4.5 The hagsvote command

Similar to hagsgr, hagsvote has both "short" and "long" options and also supports the -a flag to specify a group name if desired. The hagsvote command only displays interesting information if you catch a group (or groups) in the midst of voting protocols. This is not always easy to do. If no groups are doing anything, you get something like:

```
[sp5en0:/]# /usr/sbin/rsct/bin/hagsvote -s hags.sp5en0
  Number of: groups: 5
Group slot # [0] Group name[HostMembership] voting data:
No protocol is currently executing in the group.
-------------------------------------------------

Group slot # [1] Group name[ha_em_peers] voting data:
No protocol is currently executing in the group.
-------------------------------------------------

Group slot # [2] Group name[cssRawMembership] voting data:
No protocol is currently executing in the group.
-------------------------------------------------

Group slot # [3] Group name[enMembership] voting data:
No protocol is currently executing in the group.
-------------------------------------------------

Group slot # [4] Group name[cssMembership] voting data:
No protocol is currently executing in the group.
-------------------------------------------------

```

The hagsvote output also differs depending whether it is issued on the owner's node for a group or on a non-owner's node. It will display a summary of the collected vote responses if it is on the owner's node as well as a list of nodes that have and have not voted. On non-owner nodes, it can only list the local providers and whether they have voted or not.

The short option displays only summary data; the long option will display data for each provider (on all nodes) and for all providers in the group (on the group owner's node).

The following is the output from the `hagsvote` command when the Event Manager daemon stops on node 1:

```
Number of: groups: 4
Number of: groups: 4
Group name[ha_em_peers] voting data:
GL in phase [1] of an n-phase protocol of type[FailureLeave].
Local voting data:
Local provider count [1] Number not yet voted [0](vote submitted).
Given vote:[Approve vote]Default vote:[No vote value]
ProviderId     Voted?  Failed? Conditional?
[1/1]  Yes     Yes     No
Global voting data:
 Number of nodes in group [5] Number not yet voted [4]
 Given vote:[Approve vote]Default vote:[No vote value]
 Nodes that have voted [1 ]
 Nodes that have not voted [0 5 9 13 ]
```

From the output above, we can see that a *FailureLeave* protocol is being executed by the ha_em_peers group members. There is one local provider who has already voted (vote submitted).

The last two lines gives you a list of nodes (or providers on those nodes) who have not voted yet.

The following is the output from the `hagsvote` command when the Event Manager daemon starts on node 1:

```
Number of: groups: 4
   Number of: groups: 4
 Group name[ha_em_peers] voting data:
 GL in phase [1] of an n-phase protocol of type[Join].
 Local voting data:
 Local provider count [1] Number not yet voted [0](vote submitted).
  Given vote:[Approve vote]Default vote:[No vote value]
 ProviderId     Voted?  Failed? Conditional?
 [1/1]  Yes     No      Yes

 Global voting data:
 Number of nodes in group [5] Number not yet voted [2]
 Given vote:[Approve vote]Default vote:[No vote value]
 Nodes that have voted [1 5 9 ]
 Nodes that have not voted [0 13 ]
```

Similar to the previous output, this time the protocol is *Join*, and they are still voting.

4.2.4.6 The hagscl command

This command allows you to dump out information about each client process currently connected to Group Services. If you remember, the `lssrc -ls hags` command lists the process-IDs (pids) of the connected clients. The `hagscl` command allows you to get additional information.

The "short" option of `hagscl` is not tremendously illuminating:

```
[sp5en0:/]# /usr/sbin/rsct/bin/hagscl -s hags.sp5en0
+1  Client Control layer summary:
+2    Number of clients connected: 2
+3    Cumulative number of clients connected: 2
+4    Total number of client requests: 2
+5    Number of client hash table conflicts: 0
+6
+7  Client: socketFd[10] pid[23510]
+8  Client: socketFd[11] pid[18992]
```

This output shows the cumulative number of clients (here, no clients have left and/or returned; so, it matches the current number.) Also, the last two lines show the GS daemon's internal file descriptors to which each client's UNIX-domain socket connects to the GS daemon and the pid of each client.

The "long" option of `hagscl` is more verbose:

```
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[10] pid[23510]Total number of Clients: 2
 Client initialized: pid: 23510
  uid/gid/version: [0/200/4]
  client directory: [SuppName: length: 23 value:
/var/ha/run/haem.sp5en0

Number of local providers/subscribers: 1/3
Responsiveness information for Client: socketFd[10] pid[23510]
Type[ type[HA_GS_PING_RESPONSIVENESS]] interval[120] response time limit[120]
Checks done/bypassed[14401/0] lastResponse[OK]]
Results(good/bad/late)[14401/0/0]
Membership list:
slot    info
0       [{provider}Member token[0] Client: socketFd[10] pid[23510]ProviderId[1/0
]]
1       [{subscriber}Member token[1] Client: socketFd[10] pid[23510]]
2       [{subscriber}Member token[2] Client: socketFd[10] pid[23510]]
3       [{subscriber}Member token[3] Client: socketFd[10] pid[23510]]
--------------------------------------------------
Client: socketFd[11] pid[18992]Total number of Clients: 2
 Client initialized: pid: 18992
  uid/gid/version: [0/0/4]
  client directory: [SuppName: length: 27 value:
/var/ha/run/hagsglsm.sp5en0

Number of local providers/subscribers: 0/0
Responsiveness information for Client: socketFd[11] pid[18992]
Type[ type[HA_GS_PING_RESPONSIVENESS]] interval[3630] response time limit[10]
Checks done/bypassed[476/0] lastResponse[Not OK]]
Results(good/bad/late)[0/476/0]
Membership list:
slot    info
```

This output expands to describe a number of items:

```
Client: socketFd[10] pid[23510]Total number of Clients: 2 Client initialized: pid: 23510
```

Mostly, just a repeat of what `hagscl -s hags` says, listing pid and socket fd. Additionally, it shows whether or not the client is "initialized" (For example, has the client successfully executed ha_gs_init()?)

```
Number of local providers/subscribers: 1/3
```

How many groups has this client joined/subscribed to. In this case, it has joined one and subscribed to three.

```
Responsiveness information for Client: socketFd[10] pid[23510]
Type[ type[HA_GS_PING_RESPONSIVENESS]] interval[120] response time limit[120]
Checks done/bypassed[14401/0] lastResponse[OK]]
Results(good/bad/late)[14401/0/0]
```

Responsiveness is a periodic check that a GS daemon does to its clients. Here, it echoes the parameters given on ha_gs_init() (type/interval/time limit) and also displays the number of checks done.

```
Checks done/bypassed[14401/0] lastResponse[OK]]
Results(good/bad/late)[14401/0/0]
Membership list:
slot    info
0       [{provider}Member token[0] Client: socketFd[10] pid[23510]ProviderId[1/0
]]
1       [{subscriber}Member token[1] Client: socketFd[10] pid[23510]]
2       [{subscriber}Member token[2] Client: socketFd[10] pid[23510]]
3       [{subscriber}Member token[3] Client: socketFd[10] pid[23510]]
```

This client has joined to one group, and it has subscribed to three other groups. The "ProviderId" is the client's provider identification in its group. This can be matched (somewhat painstakingly) with the `hagsgr` output to determine the groups the client actually cares about.

4.2.4.7 The hagscount command

This command is mostly just interesting to see how "busy" the GS daemons have been. It displays the cumulative counts of various allocations and deallocations of internal data objects. Note, that the number constructed (allocated) will normally be higher than the number destructed (deallocated) since at any time some number of the objects are currently in use. The actual difference varies greatly based on the number of groups, number of providers in the groups, and the number of protocols being currently executed by all of the groups.

```
[sp5en0:/]# /usr/sbin/rsct/bin/hagscount -s hags.sp5en0
+1  Client protocols ([constructed]/[destructed]):
+2  Virtual protocols (SVProtocol) ...... [517]/[517]
+3   Joins (SJoinProtocol) ............. [20]/[20]
+4   Failures (SFailureProtocol) ........ [19]/[19]
+5   State changes (SStateProtocol) ..... [0]/[0]
+6   Provider bcast msg (SPBMProtocol) .. [0]/[0]
+7   Voluntary leaves (SLeaveProtocol) .. [0]/[0]
+8   Expels (SExpelProtocol) ........... [0]/[0]
+9   Change Attributes (SChangeAttrProtocol) [0]/[0]
+10   Reflection (SReflectionProtocol) ... [0]/[0]
```

```
+11   Subscription (SSubscribeProtocol) .. [478]/[478]
+12   Announcement (SAnnouncementProtocol) [0]/[0]
+13  Other client layer objects ([constructed]/[destructed]):
+14  Group attributes (SGroupAttributes) . [20]/[14]
+15  Client "Names" (STSuppNames) ........ [1602]/[1531]
+16   State values (SStateValue) ......... [7]/[2]
+17   Provider msgs (SProviderMessage) ... [0]/[0]
+18   Group names (SGroupName) ........... [1099]/[1050]
+19   Member names (SMemberName) ......... [484]/[475]
+20  Messages from clients (SSuppMessage)  [15329]/[15329]
+21  Client Msg containers (SSuppContainer)[479]/[479]
+22  Client Notifications (SSuppNotice) .. [539]/[537]
+23  Client votes (SVote) ................ [524]/[523]
+24  Messages from daemons (SDaemonMessage)[54]/[54]
+25  PBS Containers ([constructed]/[destructed]):
+26  PB Containers (PBContainer) ......... [134]/[129]
```

4.2.5 Tracing Group Services daemon

The Group Services daemon maintains a trace log where it writes internal trace information in a circular fashion. Normally, the daemon prints a defined set of "informational" messages in the log. However, it is possible to modify the level of tracing.

Note, that the internal tracing is level-based, and it is NOT specific to any one group or other such events. Therefore, if you increase the level of tracing, then the activity for any and all groups will result in increased output into the log, and you will have to search the log to locate information specific to a group if that is what you want.

The default level of tracing is equivalent to that settable by the `tracesoff` command. The `traceson` command has two settings, "short" and "long". It is also possible to use the `hagsctrl` command to modify the trace level:

`hagsctrl -t`	is equivalent to "traceson -l" and will set the GS daemon's internal tracing to trace all events. This will absorb disk space at a frantic rate.
`hagsctrl -o`	is equivalent to "tracesoff" and will set the GS daemon's internal tracing to its default level.

4.2.6 Case study

The host responds problem introduced in 4.2.6, “Case study” on page 245 is still causing some problems with our applications. If we check the SDR with the following command:

```
[sp5en0:/]# SDRGetObjects host_responds
node_number  host_responds
          1             1
          5             0
          9           1
         13           1
```

we see that host responds for node 5 is still zero. However, Topology Services is now working properly; so, it is time for Group Services to get a health check. Let us start with the Group Services subsystem on the CWS. We issue the following command:

```
[sp5en0:/]# lssrc -ls hags.sp5en0
Subsystem         Group            PID     Status
 hags.sp5en0      hags             23314   active
 1 locally-connected clients.  Their PIDs:
 17526
 HA Group Services domain information:
 Domain established by node 13.
 Number of groups known locally: 0
```

From the `lssrc` output, we see that the Group Services domain has been established, which means that Group Services is up an running on the CWS. Actually, this is the only thing that matters for our host responds problem since the host responds daemon gets the information from the Event Management daemon, which, instead, gets it from the Group Services daemon. All of them are running on the CWS.

However, just to make sure that Group Services is working fine on all nodes, we check the MetaGroup information on the nameserver node (node 13) as follows:

```
[sp5n13:/]# hagsmg -s hags
2.1 cssMembership: 13 9 1
1.1 ha_em_peers: 9 13 1
0.Nil ZtheNameServerXY: 13.0 9.0 1.1 0.1 5.2
0.Nil  theGROVELgroup:
```

According to the nameserver, all nodes and the CWS have Group Services are up and running (ZtheNameServerXY group). It looks like the host responds problem is not related to Group Services. However, from the output, we see that the Event Management daemon (ha_em_peers group) has not joined its group on node 5 and on the CWS. Let us discuss about Event Management first before we try to solve this problem.

4.3 Event Management

Event Management (EM) is the upper and most external layer of this cluster technology.

Event Management is a client of Group Services. It uses a group called *ha_em_peers* to maintain common state and membership information across nodes. It also uses the Reliable Messaging Library for inter-daemon communication.

Event Management provides a monitoring service through an application programming interface called EMAPI. Through this API, applications (called EM clients) can request the Event Management subsystem to monitor specific conditions. Event Management monitors those conditions, and it notifies its clients when those conditions are met.

Let us discuss some concepts first so that we can start doing some problem determination on this last RSCT component.

4.3.1 Event Management concepts

The function of the Event Management subsystem is to match information about the state of system resources with information about resource conditions that are of interest to EM clients, which may include applications, susbsystems, and other hardware and software components.

Resource states are represented by resource variables. Resource conditions are represented by expressions.

Resource Monitors are entities (usually programs) that observe the state of specific system resources and transform this state into several resource variables. The resource monitors periodically pass these variables to the Event Manager daemon. The Event Manager daemon applies expressions, which have been specified by EM clients, to each resource variable instance being monitored. If the expression is true, an event is generated and sent to the appropriated EM client. EM clients may also query the Event Manager daemon for the current values of resource variables.

Resource variables, resource monitors, and other related information are contained in several System Data Repository (SDR) object classes. Information stored in these SDR classes is then translated into a form that can be easily used by the Event Management subsystem.

The RSCT ships with several pre-defined resource monitors, most of them related to RS/6000 SP configurations. Figure 135 on page 248 shows the relationship between pre-defined resource monitors and the Event Management daemon.

Resource monitors communicate with the Event Manager daemon through a Resource Monitor API (RMAPI). Through this API, the Event Manager daemon also has some control over Resource Monitors.

The RMAPI is a shared library that is supplied in a non-thread safe version only. The library is located in /usr/lib/libha_rr.a, which is a symbolic link to /usr/sbin/rsct/lib/libha_rr.a.

For detailed information about the RMAPI, see *RSCT: Event Management Programming Guide and Reference*, SA22-7354.

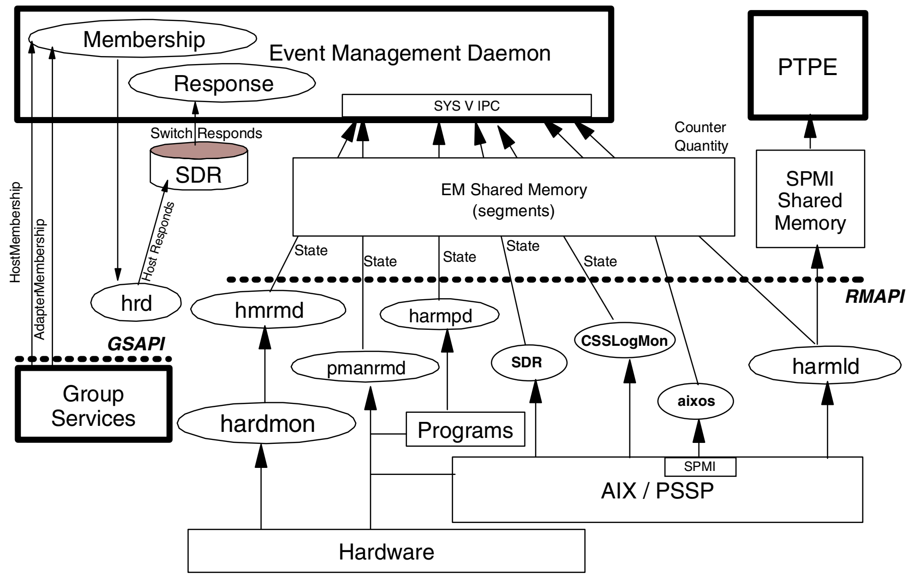


Figure 135. RS/6000 SP Resource Monitors

Resource Monitors are implemented in four different ways:

- *As a daemon*. When EM clients subscribe to Event Management for monitoring resources whose resource variables are supplied for these type of Resource Monitors, Event Management starts the corresponding daemon(s). The Event Management daemon connects to the Resource Monitor through the RMAPI. The connection type is known as a *server* connection.
- *As part of the subsystem that manages the resource*(s). Event Management connects to the Resource Monitor through the RMAPI, but it does not start it. This connection type is know as a *server* connection.

- *As a command*. The command can be used by scripts to supply resource variables to the Event Manager daemon. A command-based Resource Monitor connects to the Event Manager daemon through the RMAPI. The connection type is known as a *client* connection.
- *Internal to Event Manager daemon*. The Event Manager daemon itself performs some resource monitoring functions that subscribe to Group Services information for adapter and host membership. These Resource Monitors maintain a connection type known as *internal*.

RSCT provides seven external Resource Monitors and two internal as shown in Figure 135 on page 248. The following is a brief definition for each one of them:

- IBM.PSSP.harmld

  This monitor supplies resource variables for the CSS, VSD, and LoadLeveler subsystems. This data is also transferred through SPMI to the Performance Monitor subsystem. This is a daemon (harmld) with a connection type of server.

- IBM.PSSP.harmpd

  This monitor provides resource variables that represent the number of processes executing a particular program. These variables can be used to determine whether a particular system daemon is running. This is a daemon (harmpd) with a connection type of server.

- IBM.PSSP.hmrmd

  This monitor provides resource variables obtained from the PSSP hardware monitoring subsystem (hardmon). This is a daemon with a connection type of server.

- IBM.PSSP.pmanrmd

  This monitor supplies the resource variables of the PSSP Problem Management subsystem. This is a command-based resource monitor with a connection type of client (pmand).

- aixos

  This monitor provides resource variables that represent AIX operating system resources. This is a daemon (harmad) with a connection type of server.

- IBM.PSSP.CSSLogMon

  This monitor supplies a resource variable that represents the state of CSS error log entries. This is a command-based resource monitor with a connection type of client.

- IBM.PSSP.SDR

  This monitor provides a resource variable that represents the modification state of SDR classes. This is a command-based resource monitor with a connection type of client.

- Membership

  This monitor supplies resource variables that represent the Host Membership and Adapter Membership states. The Event Manager daemon obtains this information directly from the Group Services subsystem by subscribing to the HostMembership, enMembership, and cssMembership system groups. This is an internal resource monitor.

- Response

  This monitor provides resource variables that represent the information in the host_responds and switch_responds SDR classes. This is an internal resource monitor.

4.3.2 Event Management initialization

Normally, the Event Manager daemon start-up program, haemd_SP, is started by an entry in the /etc/inittab file using the `startsrc` command. If necessary, you can start the start-up program using the `haemctrl` command or the `startsrc` command, directly. The SRC subsystem calls the haem_SP executable as you can see from the SRC definition:

```
sp5en0:/>odmget -q subsysname=haem.sp5en0 SRCsubsys

SRCsubsys:
        subsysname = "haem.sp5en0"
        synonym = ""
        cmdargs = "192.168.5.150"
        path = "/usr/sbin/rsct/bin/haemd_SP"
        uid = 0
        auditid = 0
        standin = "/dev/null"
        standout = "/dev/null"
        standerr = "/dev/null"
        action = 1
        multi = 0
        contact = 3
        svrkey = 0
        svrmtype = 0
        priority = 20
        signorm = 0
        sigforce = 0
        display = 1
```

```
waittime = 120
grpname = "haem"
```

The haem_SP executable performs the following steps:

1. It gets the number of the node on which it is running using the `/usr/lpp/ssp/install/bin/node_number` command. Node 0 is the CWS.
2. It fetches the name of the system partition and the EMCDB version string from the Syspar SDR class. (Recall that one instance of the Event Manager daemon runs on the CWS for each system partition to which the Event Management subsystem was added.) It also fetches the Event Manager daemon remote client communications port number from the SP_ports SDR class. Let us check the Syspar class:

   ```
   sp5en0:/>SDRGetObjects Syspar
   syspar_name  ip_address   install_image syspar_dir   code_version haem_cdb_version
   auth_install auth_root_rcmd auth_methods
   sp5en0       192.168.5.150 default      ""           PSSP-3.1     917276294,59430400,0
   k4           k4           k4:std
   ```

   From the output, we can see the EMCDB version stored in the SDR. The EMCDB gets rebuilt by using the `haemcfg` command, or by removing and re-creating the haem SRC subsystem through the `haemctrl` command. The version number stored in the Syspar class will be compared, by the haem daemon, with the run-time version being used by its peers (if this daemon is not the first one to start).
3. Finally, the haemd_SP executable invokes the Event Manager daemon (haemd), passing the information just collected and any arguments passed to the executable itself. Note, that a new process is not started; the process image is just replaced. This permits the Event Manager daemon to be controlled by the SRC. During its initialization, the Event Manager program performs the following steps:
   a. It 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.
   b. It removes from the registration cache all of 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 the subdirectory.

      Note, that subdirectories for remote clients cannot be removed automatically because the Event Manager daemon cannot determine if remote processes still exist.
   c. It tries to connect to the Group Services subsystem. If the connection cannot be established because the Group Services subsystem is not running, it is scheduled to be retried in five seconds. This continues

until the connection to Group Services is established. Meanwhile, Event Manager daemon initialization continues.

Once the connection with GS is established, the EM daemon will send a join request to the ha_em_peers group (here is where the EM buddies are). Once the join request gets approved, the EM daemon compares the EMCDB version number used by the group with the version it obtained from the Syspar class. If they are different, it will load the correct configuration database from the staged area (/etc/ha/cfg).

By using some of the Group Services commands described in 4.2.4, "Group Services commands" on page 231, you may get the EMCDB version number out of the ha_em_peers. To do that, you can use the following command:

```
/usr/sbin/rsct/bin/hagsgr -a ha_em_peers -ls haem.sp5en0
```

In your particular case, replace the sp5en0 for the system partition you want to query. The output of this command is quite verbose, as we mentioned previously; so, you may want to pipe the output of this command to something, such as more or pg, then you can search for Group State Value. What you should get is something similar to the following:

```
...
Group state value: [min/max lengths (1/256)] actual length[34]  value:
0x39313732 0x37363239 0x342c3539 0x34333034 0x30302c30 0x004e4f53 0x45435355
0x50504f52 0x5400
917276294,59430400,0.NOSECSUPPORT.
...
```

From this partial output, you can see the version number displayed at the last line. This number can be compared to the version number stored in the Syspar class. There is another way that you can get the EMCDB version that the daemon are running. You can use the `lssrc` command as explained in 4.3.4, "Using the lssrc command" on page 256.

d. It enters the main control loop.

   In this loop, the Event Manager daemon waits for requests from EM clients, messages from resource monitors and other Event Manager daemons, messages from the Group Services 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 Group Services.

However, EM client requests, messages from resource monitors, and messages from other Event Manager daemons (called peers) are refused until the Event Manager daemon has successfully joined the daemon peer group (the ha_em_peers group) and has fetched the correct version of the EMCDB.

4.3.2.1 Joining the peer group

After the Event Manager daemon has successfully established a connection with the Group Services subsystem, it tries to join the daemon peer group (ha_em_peers). If this is the first Event Manager daemon to come up in the domain, it establishes the peer group. Otherwise, the other daemons in the peer group either accept or reject the daemon's join request. If an existing peer group member is still recovering from a prior termination of the joining daemon, the join request is rejected. If its join request is rejected, the daemon tries to join again in 15 seconds. This continues until the daemon's join request is accepted.

When it joins the daemon peer group, the Event Manager daemon examines the group state. The group state contains the EMCDB version string.

If the group state is null, the joining daemon proposes that the group state be set to the version string that the daemon has fetched from the SDR. If several daemons try to join the group at about the same time, and the group state is null, then each daemon proposes the group state. When the group is formed, Group Services selects one of the proposals and sets the group state to it. Note, that each daemon is proposing the EMCDB version string that it has fetched from the SDR. Unless the `haemcfg` command has been run at about the same time, the proposed version strings should be identical.

If the group state is not null when it is examined by the joining daemon, a group has already formed, and the daemon does not propose a new group state.

After the daemon has successfully joined the peer group, it compares the EMCDB version string contained in the group state to the version string it fetched from the SDR. If they are different, the version that was fetched from the SDR is replaced by the version in the group state.

An Event Manager daemon is prevented from joining the peer group as long as any other Event Manager daemon, currently in the peer group, is non-responsive to "pings" from the Group Services subsystem. (When an Event Manager daemon successfully joins the peer group, Group Services requests a response from the Event Manager daemon every two minutes. If the daemon does not respond to the request within two minutes, it is considered to be non-responsive. A daemon is also considered to be non-responsive if it does not reply to the join requests of other daemons within one minute.) The Event Manager daemon status, as displayed by the `lssrc` command, indicates if a daemon cannot join the peer group. If this is the case, the em.default.<syspar> file of any other daemon in the peer group should be examined for errors indicating that an Event Manager daemon is non-responsive. If so, and the non-responsive daemon does not terminate itself within a few minutes, perform the User Response specified for the error.

4.3.2.2 Reading the EMCDB

Once the daemon has joined the peer group and has determined the EMCDB version, it reads the run-time EMCDB file from the /etc/ha/cfg directory. If the file does not exist, it is copied from the staging directory on the CWS (/spdata/sys1/ha/cfg).

Once the daemon has read the file, it compares the version string in the EMCDB to the one it fetched (from the SDR or from the group state). If the two version strings do not match, and the daemon has not just copied the EMCDB from the CWS, then it copies the run-time EMCDB from the CWS. If the version strings still do not match, the daemon terminates with an error.

Often, this happens when somebody deletes the EMCDB files from the /etc/ha/cfg directory on the CWS and on a particular node while the remaining nodes are up and running. The files in the staging area are removed (doing some cleaning), and then the `haemcfg` command is run for generating a new EMCDB. In this case, when Event Management is restarted on that node, it will fail because the working group is using the previous files, which have been removed from the CWS and the node. Workaround? Yes, stop the EM daemon on all running nodes within that partition and restart them, or if you cannot do that immediately, copy the EMCDB file from one of the running nodes to the /etc/ha/cfg directory on the node you want to start Event Management. But, remember that by doing the latter, any change to the EMCDB will not be reflected until all the EM daemons are stopped, the group dissolved, and then restarted again.

Whenever it is necessary to copy the EMCDB from the CWS, the EMCDB version string is used to determine how the copy is done. If the EMCDB version string was obtained from the group state, then it is used to copy a back level EMCDB from the staging directory on the CWS. Otherwise, the staging file, /spdata/sys1/ha/cfg/em.<syspar>.cdb, is copied. Note, that back level copies of the EMCDB should be removed from the staging directory only if their version string suffix indicates a time stamp older than the current version string found in the daemon peer group state (the current version string is found in the Event Manager daemon status as displayed by the `lssrc` command).

After the daemon has read and validated the EMCDB, it enables daemon communications. This permits EM clients to send requests to the daemon, resource monitors to connect to the daemon, and peers to send messages to the daemon. At this point, the initialization of the Event Manager daemon is complete.

To copy the EMCDB from the CWS to the /etc/ha/cfg directory, the Event Manager daemon uses the /usr/sbin/rsct/install/bin/haemrcpcdb script. This script uses the `rcp` command to perform the actual copy.

The way in which Event Manager daemons determine the EMCDB version has important implications for the configuration of the subsystem. To place a new version of the EMCDB into production (that is, to make it the run-time version that is used by the Event Management subsystem), you must stop each Event Manager daemon in the domain after the `haemcfg` command is run. Stopping the daemons dissolves the existing peer group. To verify that the group has been removed, use the `hagsgr` command on the Group Services nameserver node to make sure there is no ha_em_peers group defined. Once the existing peer group is dissolved, the daemons can be restarted. As they restart, the daemons form a new peer group. A new EMCDB version string can be submitted as the group state only when a peer group is formed.

4.3.3 Event Management logs

Event Management subsystem uses the AIX error log as the main repository for errors and informational messages. Besides the logging of informational messages and errors, the daemon can log additional and detailed information if tracing is activated.

The tracing function can be activated by using the `haemctrl` command, and it is provided to supply additional problem determination information when it is requested by the IBM Support Center. Normally, tracing should not be turned on because it may degrade Event Management subsystem performance and can consume large amounts of disk space in the /var file system.

The log files are located in the /var/ha/log directory. The following is a brief description of each file:

em.default.<syspar>	contains any error message from the Event Manager daemon that cannot be written to the AIX error log. Normally, all the daemon error messages are written to the AIX error log. This log

	also contains error messages that result from repetitive operational errors, for example, when the Event Manager daemon cannot connect to Group Services, and it retires every five seconds, or when it tries to join the ha_em_peers group every 15 seconds.
em.trace.<syspar>	contains trace output from the Event Manager daemon.
em.msgtrace.<syspar>	contains message trace output from the Event Manager daemon.

The size of the em.default.<syspar> file is examined every two minutes. If the size exceeds 256 KB, the file is renamed with a suffix of last, and a new default file is created. No more than two copies of this file are kept.

The Event Manager also records additional information into the em.defaults.<syspar> log file if it cannot start a resource monitor. The error information includes the name of the resource monitor that could not be started.

If the EM daemon detects an error in the shared memory segment used by the daemon or a resource monitor instance, it creates a dump file containing the first 4096 bytes of the shared memory segment in the /var/ha/run/haem.<syspar> directory. The dump is called rzdump.RM*rmname.rminst.time* (where *rmname.rminst* are the Resource Monitor name and instance, and *time* is a time stamp).

4.3.4 Using the lssrc command

The `lssrc` command with the standard flags gives you status information about the SRC subsystem. For example, we could use the `lssrc` command to see if the Event Management subsystem is active:

```
sp5en0:/>lssrc -s haem.sp5en0
Subsystem         Group            PID     Status
 haem.sp5en0      haem             21266   active
```

This command does not give us much information about the internal status of the subsystem. To get more information, you can use the long list flag (`-l`). By using this flag, the SRC controller will query the daemon for status information, and it will dump it to the screen. The output of this command is little bit long for showing it here in a single piece; so, let us take it in parts. The first part shows status and connectivity information as follows:

```
[sp5en0:/]# lssrc -ls haem.sp5en0
Subsystem         Group            PID     Status
 haem.sp5en0      haem             21266   active

Trace flags set:  None

 Configuration Data Base version: 917276294,59430400,0(SDR)

 Daemon started on 02/19/99 at 13:39:49.314620416
    running 3 days, 0 hours, 18 minutes and 59 seconds
 Daemon connected to group services: TRUE
 Daemon has joined peer group:       TRUE
 Daemon communications enabled:      TRUE
 Peer count:                         3

 Peer group state:
         917276294,59430400,0
```

This first portion of the output gives you a good deal of information. The first lines correspond to the standard output you get without the `-l` flag. So, it tells you if the subsystem is active or inoperative. Then, the status of the trace flag, which is off in this case.

The next line shows the EMCBD version number stored in the SDR, and the last line of this partial output shows the version number used by the ha_em_peers group. You can compare them and see if the daemons are using an EMCDB version different than the one stored in the SDR. This usually happens when you run the `haemcfg` command that creates a new EMCDB, but you have not restarted the EM daemons. Remember that all the EM daemons in the partition (domain) need to be stopped and the group dissolved (ha_em_peers) in order to use the new EMCDB.

The output also tells you how long the daemon has been running:

```
Daemon started on 02/19/99 at 13:39:49.314620416
    running 3 days, 0 hours, 18 minutes and 59 seconds
```

If the daemon was able to connect to Group Services:

```
Daemon connected to group services: TRUE
```

If the daemon was able to join the ha_em_peers group:

```
Daemon has joined peer group:       TRUE
```

And finally, if the daemon has enabled communication with clients:

```
Daemon communications enabled:      TRUE
```

The last line of this stanza gives you the number of providers in the ha_em_peers group. In this case, it is three; so, if we are using the same configuration shown in Figure 124 on page 193, there is one EM daemon missing. How can we know which EM daemon has not started or is able to join the ha_em_peers group? Yes, we can use the `hagsgr` command to get this information:

```
[sp5en0:/]# /usr/sbin/rsct/bin/hsgsgr -a ha_em_peers -s hags.sp5en0
sp5en0:/usr/sbin/rsct/bin>./hagsgr -a ha_em_peers -s hags.sp5en0|more
  Number of: groups: 5
Group name[ha_em_peers] group state[Inserted |Idle |]
Providers[[1/5][1/13][1/9][1/0]]
Local subscribers[]
```

From the output, we can see all the providers for this group. The providers list does not include node 1; so, it means the EM daemon on node 1 is either not running or has been unable to join the ha_em_peers group.

The second portion of the output is related to EM clients and Resource Monitors. The partial output is as follows:

```
Logical Connection Information
  Type   LCID   FD  Node/PID  Start Time
 local      0   11      8204   Fri Feb 19 13:39:50 1999
 local      1   13     20168   Mon Jan 25 10:18:31 1999
 local      2   15     15958   Fri Feb 19 13:39:55 1999
 local      5   16     22598   Fri Feb 19 13:59:17 1999

 Resource Monitor Information
   Resource Monitor Name    Inst  Type   FD    SHMID     PID    Locked
IBM.PSSP.CSSLogMon            0     C    -1        -1      -2  No  00/00
IBM.PSSP.SDR                  0     C    -1        -1      -2  No  00/00
IBM.PSSP.harmld               0     S    -1        -1      -1  No  00/00
IBM.PSSP.harmpd               0     S    -1        -1      -1  No  00/00
IBM.PSSP.hmrmd                0     S    19        -1   23740  No  01/01
IBM.PSSP.pmanrmd              0     C    14        -1      -2  No  00/00
Membership                    0     I    -1        -1      -2  No  00/00
Response                      0     I    -1        -1      -2  No  00/00
aixos                         0     S    12   3407876      -2  No  00/01

 Highest file descriptor in use is 19
```

The Logical Connection Information section lists all the EM clients connected to the daemon. In this case, all EM clients are local; so, no Node ID is displayed.

The section titled Resource Monitor Information list all the information for the internal and external resource monitors. Not all the Resource Monitors will be active at any given time; so, some of the fields display an invalid value. The description of the columns displayed is as follow:

Inst	The Resource Monitor instance number.
Type	The type of Resource Monitor (C: Client, S: Server, I: Internal).
FD	An internal to EM file descriptor.
SHMID	The shared memory segment ID used to communicate with that Resource Monitors. It can be verify with the `ipcs -m` command.
PID	The process ID for Resource Monitors external to the Event Manager daemon.
Locked	The EM daemon maintains two counters: One for start attempts, and one for successful connections. If either of these counters hits the start limit or connect limit, respectively, the RM is locked. The counters are cleared two hours after the first start or connect. For starts, the limit is three. For connects, the limits is three times the number of instances configured for the resource monitor (rmNum_instances in the EM_Resource_Monitor class). For all resource monitors shipped with PSSP, rmNUM_instances is 1.

Once the Event Manager daemon has successfully connected to a Resource Monitor of type *server*, the daemon attempts to reconnect to the resource monitor if it should terminate. The reconnection is attempted at the rate of one per minute. However, reconnection attempts are constrained under the following circumstances:

1. If it is necessary that the daemon start the resource monitor before each reconnection attempt, then after three attempts within two hours, the resource monitor is locked, and no further attempts are made.
2. If the resource monitor is not startable by the Event Manager daemon, then, after about three unsuccessful reconnections within two hours, the resource monitor is locked, and no further attempts are made.

The rationale for locking the resource monitor is that, if it cannot be started and stay running, or successful connections are frequently being lost, then a problem exists with the resource monitor. Once the problem has been determined and corrected, the `haemunlkrm` command can be used to unlock the resource monitor and start it. This command resets the start and connect counters to 0 and also resets the two hour window. Note, that locking does not apply to *client* type resource monitors.

The final portion of the command output shows several internal EM counters and the size of the data memory segment being used by the daemon. The final portion is as follow:

```
Peer Daemon Status
    0 S S     5 I A     9 I A    13 I A

 Internal Daemon Counters
     GS init attempts =         1  GS join attempts =         1
     GS resp callback =      2187  CCI conn rejects =         0
     RMC conn rejects =         0  HR conn rejects  =         0
     Retry req msg    =         0  Retry rsp msg    =         0
     Intervl usr util =         1  Total usr util   =      3343
     Intervl sys util =         3  Total sys util   =      3513
     Intervl time     =     12001  Total time       =  26245048
     lccb's created   =         8  lccb's freed     =         4
     Reg rcb's creatd =        32  Reg rcb's freed  =        29
     Qry rcb's creatd =         3  Qry rcb's freed  =         3
     vrr created      =        32  vrr freed        =        29
     vqr created      =      1047  vqr freed        =      1047
     var inst created =       322  var inst freed   =         0
     Events regstrd   =        32  Events unregstrd =        29
     Insts assigned   =        28  Insts unassigned =        15
     Smem vars obsrv  =         0  State vars obsrv =    104991
     Preds evaluated  =        67  Events generated =        28
     Smem lck intrvl  =         0  Smem lck total   =         0
     PRM msgs to all  =         0  PRM msgs to peer =         0
     PRM resp msgs    =         0  PRM msgs rcvd    =         0
     PRM_NODATA       =         0  PRM_BADMSG errs  =         0
     Sched q elements =        32  Free q elements  =        31
     xcb alloc'd      =       125  xcb freed        =       125
     xcb freed msgfp  =         0  xcb freed reqp   =         0
     xcb freed reqn   =         0  xcb freed rspc   =       102
     xcb freed rspp   =         0  xcb freed cmdrm  =        23
     xcb freed unkwn  =         0
 Daemon Resource Utilization
        User:           0.010 secs    0.008% (last interval)
                       33.430 secs    0.013% (total)
        System:         0.030 secs    0.025% (last interval)
                       35.130 secs    0.013% (total)
        U+S:            0.040 secs    0.033% (last interval)
                       68.560 secs    0.026% (total)

        Date segment size:  1924K
```

Most of the counters displayed here are difficult to explain without getting into really deep detail. However, there are a couple of counters that can be useful for diagnosing problems with this subsystem.

The *GS init attempts* and *GS join attempts* counters give you information about the number of times the EM daemon tried to connect to Group Services (GS init attempts) and the number of times the EM daemon tried to join the ha_em_peers group (GS join attempts). After the daemon has join the ha_em_peers group, Group Services send a callback every two minutes. The response to that callback gets counted in the *GS resp callback* counter. This number should increase by one every two minutes. If not, it means that there is a communication problem between Group Services and the Event Manager daemon.

4.3.5 Tracing Event Management daemon

The tracing function built-in the Event Management daemon can be activated by using the `haemtrcon` command. When activated, the Event Management daemon supplies additional problem determination information when requested by the IBM Support Center. Normally, tracing should not be turned on because it may degrade Event Management subsystem performance and consume large amounts of disk space in the /var file system.

The trace files are located in the /var/ha/log directory. The following is a brief description of each file:

- em.trace.<syspar>: This file contains trace output from the Event Management daemon.
- em.msgtrace.<syspar>: This file contains message trace output from the Event Management daemon.

We strongly recommend not to activated the tracing facility unless instructed by your IBM Support Center to do so. However, there are some additional options you can use to generate or "dump" internal information from the

Event Management daemon that will not cause a performance degradation or consume large amounts of disk space.

There are three optional arguments that can be passed on to the daemon by using the `haemtrcon` command. The syntax for the `haemtrcon` command is as follows:

```
Usage:
    haemtrcon [-h host] [-a argument] -g group_name
    haemtrcon [-h host] [-a argument] -s subsystem_name
    haemtrcon [-h host] [-a argument] -p subsystem_pid
```

The arguments accepted by the daemons for dumping information are shown in Table 5.

Table 5. Arguments for Event Management daemon

Argument	Description
regs	Dumps registered events
dinsts	Dumps registered instances
olists	Dumps observation lists

For example, to see all the events registered with Event Management, we could use the following command:

```
[sp5en0:/var/ha/log]# haemtrcon -a regs -s haem.sp5en0
haemtrcon: the specified trace flags have been set (00000000)
```

This command sends a request to the Event Management daemon to dump all registered events. The daemon will then dump the requested information into the em.trace.<syspar> file in the /var/ha/log directory. The content of the file looks as follows:

```
Trace Started at 11/30/99 14:38:01.509522432


Registered Events:
    0  0x00000000 (    0,    0)     0  IBM.PSSP.Membership.LANAdapter.state  "X==0"
"X==1"
                                       AdapterNum=0 AdapterType=en NodeNum=0
                                       AdapterNum=0 AdapterType=en NodeNum=5
                                       AdapterNum=0 AdapterType=en NodeNum=9
                                       AdapterNum=0 AdapterType=en NodeNum=13
                                       AdapterNum=0 AdapterType=en NodeNum=1
    1  0x00000000 (    0,    0)     2  IBM.PSSP.Response.Host.state  "X==1 && X@P==0"
""
                                       NodeNum=5
                                       NodeNum=9
                                       NodeNum=13
                                       NodeNum=1
    2  0x00010000 (    1,    0)     2  IBM.PSSP.Membership.LANAdapter.state  "X==0 &&
X@P==1"  ""
```

```
                                          AdapterNum=0 AdapterType=css NodeNum=13
                                          AdapterNum=0 AdapterType=css NodeNum=9
                                          AdapterNum=0 AdapterType=css NodeNum=5
                                          AdapterNum=0 AdapterType=css NodeNum=1
    3  0x00000000 (    0,    0)     3  IBM.PSSP.CSSlog.errlog  "X@1 != 0"  ""
                                          No instances currently assigned
    4  0x00000000 (    0,    0)     1  IBM.PSSP.pm.User_state1  "X@0!=X@P0"  ""
                                          No instances currently assigned
    9  0x00000000 (    0,    0)    18  IBM.PSSP.SDR.modification  ""  ""
                                          No instances currently assigned
   10  0x00010000 (    1,    0)    18  IBM.PSSP.SDR.modification  ""  ""
                                          Class=EM_Condition
   14  0x00050000 (    5,    0)    18  IBM.PSSP.SDR.modification  ""  ""
                                          No instances currently assigned
   15  0x00060000 (    6,    0)    18  IBM.PSSP.SDR.modification  ""  ""
                                          No instances currently assigned
   16  0x00070000 (    7,    0)    18  IBM.PSSP.SDR.modification  ""  ""
                                          Class=Node
   17  0x00080000 (    8,    0)    18  IBM.PSSP.SDR.modification  ""  ""
                                          No instances currently assigned
```

As you can see, there several event registered with Event Management, but only few of them have instances currently assigned. An event with no instances assigned is an event known to Event Management but not currently active.

The first three event are the only ones active. We can see that Event Management is monitoring the Ethernet (en) and SP Switch (css) adapters through the Membership resource variable. Also, it is monitoring the Response variable for host in all the nodes.

Let us activate a file system monitor and dump this information again. The result is as follows:

```
Trace Started at 11/30/99 14:51:02.012841216


Registered Events:
    0  0x00000000 (    0,    0)     0  IBM.PSSP.Membership.LANAdapter.state  "X==0"
"X==1"
                                       AdapterNum=0 AdapterType=en NodeNum=0
                                       AdapterNum=0 AdapterType=en NodeNum=5
                                       AdapterNum=0 AdapterType=en NodeNum=9
                                       AdapterNum=0 AdapterType=en NodeNum=13
                                       AdapterNum=0 AdapterType=en NodeNum=1
    1  0x00000000 (    0,    0)     2  IBM.PSSP.Response.Host.state  "X==1 && X@P==0"
""
                                       NodeNum=5
                                       NodeNum=9
                                       NodeNum=13
                                       NodeNum=1
    2  0x00010000 (    1,    0)     2  IBM.PSSP.Membership.LANAdapter.state  "X==0 &&
X@P==1"  ""
                                       AdapterNum=0 AdapterType=css NodeNum=13
                                       AdapterNum=0 AdapterType=css NodeNum=9
                                       AdapterNum=0 AdapterType=css NodeNum=5
                                       AdapterNum=0 AdapterType=css NodeNum=1
    3  0x00000000 (    0,    0)     3  IBM.PSSP.CSSlog.errlog  "X@1 != 0"  ""
```

```
                                          No instances currently assigned
  4  0x00000000 (    0,    0)     1  IBM.PSSP.pm.User_state1  "X@0!=X@P0"  ""
                                          No instances currently assigned
  9  0x00000000 (    0,    0)    18  IBM.PSSP.SDR.modification  ""  ""
                                          No instances currently assigned
 10  0x00010000 (    1,    0)    18  IBM.PSSP.SDR.modification  ""  ""
                                          Class=EM_Condition
 14  0x00050000 (    5,    0)    18  IBM.PSSP.SDR.modification  ""  ""
                                          No instances currently assigned
 15  0x00060000 (    6,    0)    18  IBM.PSSP.SDR.modification  ""  ""
                                          No instances currently assigned
 16  0x00070000 (    7,    0)    18  IBM.PSSP.SDR.modification  ""  ""
                                          Class=Node
 17  0x00080000 (    8,    0)    18  IBM.PSSP.SDR.modification  ""  ""
                                          No instances currently assigned
 18  0x00090000 (    9,    0)    18  IBM.PSSP.aixos.FS.%totused  "X>90"  "X<60"
                                          VG=rootvg LV=hd3
```

As you can see from this new output, there is a new event (18), which is the one we have just activated.

If you want to get more information about this event, you can use one of the other two arguments described in Table 5 on page 262.

For example, the "olists" argument will give you details on the registered monitor as follows:

```
Trace Started at 11/30/99 14:56:45.161388544

Obsv control = 0x200605c8, interval = 60.000000, flags = 0x0000, last obsv = 943991746
874736
  Obsv list = 0x2002b8e8, delay = 20.000000, number of ptr lists elements = 1
    limit = 10000, inst count = 16
      Normal list:
        IBM.PSSP.aixos.FS.%totused
        vector:        VG=rootvg LV=hd3
        API instance ID = 2, RM instance ID = 18465, RM instance number = 0
        current value: 8.072917, raw: 8.072917
        flags:         0003    qcnt:   0

      Immediate list:
```

From this output, you can see that the sample interval for this variable is 60 seconds and that the current value is a little bit over 8 percent. From the instance or vector, we can tell that this is a monitor of the /tmp file system (hd3), and the previous output gave us the condition (X>90) and rearm condition (X<60).

This tracing or dump facility from Event Management is really helpful in situations where events registered through either SP Perspectives, PMAN, or the EMAPI directly, do not seem to be working.

4.3.6 Case study

From previous sections, we learned that the host responds problem is not related to Topology Services or Group Services since both subsystems are working properly on the CWS and nodes.

Event Management is the last component of this RSCT infrastructure, and as we have seen, it provides the resource variables and monitoring capabilities for applications, such as host responds, to get notifications on changes of those resource variables.

Let us check the Event Management subsystem on the CWS since the host responds daemon gets the information from there. We do this by issuing the following command:

```
[sp5en0:/]# lssrc -ls haem.sp5en0
0513-036 The request could not be passed to the haem.sp5en0 subsystem.
Start the subsystem and try your command again.
```

The subsystem is not running; so, we start it and run the `lssrc` command again as follows:

```
[sp5en0:/]# startsrc -s haem.sp5en0
0513-059 The haem.sp5en0 Subsystem has been started. Subsystem PID is 22932.
[sp5en0:/]# lssrc -ls haem.sp5en0
0513-056 Timeout waiting for command response. If you specified a foreign host,
see the /etc/inittab file on the foreign host to verify that the SRC daemon
(srcmstr) was started with the -r flag to accept remote requests.
[sp5en0:/]# lssrc -ls haem.sp5en0
0513-036 The request could not be passed to the haem.sp5en0 subsystem.
Start the subsystem and try your command again.
```

The subsystem died after we restarted it. The next thing to do would be to check for error messages in the AIX error log or in the Event Management log file. Let us first check the AIX error log as follows:

```
[sp5en0:/]# errpt -N haemd -a
LABEL:          HA002_ER
IDENTIFIER:     12081DC6

Date/Time:       Thu Dec  2 09:13:02
Sequence Number: 104
Machine Id:      000509306700
Node Id:         sp5en0
Class:           S
Type:            PERM
Resource Name:   haemd

Description
SOFTWARE PROGRAM ERROR

Probable Causes
SUBSYSTEM

Failure Causes
SUBSYSTEM
```

```
            Recommended Actions
            REPORT DETAILED DATA
            CONTACT APPROPRIATE SERVICE REPRESENTATIVE

    Detail Data
    DETECTING MODULE
    LPP=PSSP,Fn=emd_cdb.c,SID=1.18.1.1,L#=637,
    DIAGNOSTIC EXPLANATION
    haemd(sp5en0): 2521-025 Cannot copy /etc/ha/cfg/em.sp5en0.cdb.941121960,49405465
    6,0 from CWS, return code is 4.
```

This output shows an error on the Event Management subsystem start up program trying to copy the Event Management Configuration Database from the staging area (/spdata/sys1/ha/cfg) to the run-time directory (/etc/ha/cfg).

From the output, we see the EMCDB version number that the start up program is trying to copy is 941121960,494054656,0. So let us check if this version number corresponds to the one stored in the SDR, or if it is the run-time version being used by all the other Event Management daemons. To check the version number in the SDR, we issue the following command:

```
[sp5en0:/]# SDRGetObjects Syspar haem_cdb_version
haem_cdb_version
943999054,965059072,0
```

So, the version number in the SDR is different than the one Event Management on the CWS is trying to use. It must be the version number being used by all the other Event Management daemons. There are several ways you can check this. One way would be to check the ha_em_peers group information as follows:

```
[sp5en0:/]# hagsgr -ls hags.sp5en0 -a ha_em_peers|grep -p "Group state"
Group state value: [min/max lengths (1/256)] actual length[35]  value:
0x39343131 0x32313936 0x302c3439 0x34303534 0x3635362c 0x30004e4f 0x53454353
0x5550504f 0x525400
941121960,494054656,0.NOSECSUPPORT.
```

Another way would be to log on to one of the nodes where Event Management is running and issue the `lssrc` command as follows:

```
[sp5n13:/]# lssrc -ls haem|grep "version"
 Configuration Data Base version: 941121960,494054656,0(SDR)
```

As you can see, the Event Management subsystem start up program on the CWS is trying to copy this configuration database from the staging area (/spdata/sys1/ha/cfg). However, for some reason, it could not copy it. Most likely, the configuration database file was removed.

The easiest way out of this problem is to stop all the Event Management daemon on this partition, then check that the ha_em_peers group gets disolved, and then restart the daemons everywhere.

We stop the Event Management daemons as follows:

```
[sp5en0:/]# dsh -a 'stopsrc -s haem'
sp5n01: 0513-044 The haem Subsystem was requested to stop.
sp5n05: 0513-004 The Subsystem or Group, haem, is currently inoperative.
sp5n09: 0513-044 The haem Subsystem was requested to stop.
sp5n13: 0513-044 The haem Subsystem was requested to stop.
```

Then, we check that the ha_em_peers group was disolved as follows:

```
[sp5en0:/]# hagsgr -ls hags.sp5en0 -a ha_em_peers|grep -p "Group state"
  Number of: groups: 3
Information for SGroup: Group name[ha_em_peers]
I am *not* Group Leader!  Created by join request.
group state[Inserted |Idle |]
ProtocolToken[27/52]
 counts: (prov/localprov/subs) [0/0/0]
 delayed join count [0]
 protocol counts:  received [0]
  dropped(total/DaemonMsg/ProtMgr) [0/0/0]
 message counts:  current queued [0] future queued/cumulative [0/0]
No established attributes.
Group state value: [min/max lengths (1/256)] actual length[4]  value:
0x00000000
....
```

This output shows that there are no providers on this group and the group state value is zero. We then start the daemon on the CWS and on every node as follows:

```
[sp5en0:/]# startsrc -s haem.sp5en0
0513-059 The haem.sp5en0 Subsystem has been started. Subsystem PID is 12292.
[sp5en0:/]# dsh -a 'startsrc -s haem'
sp5n01: 0513-059 The haem Subsystem has been started. Subsystem PID is 14250.
sp5n05: 0513-059 The haem Subsystem has been started. Subsystem PID is 10460.
sp5n09: 0513-059 The haem Subsystem has been started. Subsystem PID is 16514.
sp5n13: 0513-059 The haem Subsystem has been started. Subsystem PID is 17122.
```

Finally, we check the Event Management subsystem information as follows:

```
[sp5en0:/]# lssrc -ls haem.sp5en0|more
0513-036 The request could not be passed to the haem.sp5en0 subsystem.
Start the subsystem and try your command again.
```

It looks like the Event Management daemon died again. We check for errors as follows:

```
[sp5en0:/]# errpt -N haemd -a
LABEL:          HA002_ER
IDENTIFIER:     12081DC6

Date/Time:       Thu Dec  2 09:52:13
Sequence Number: 120
Machine Id:      000509306700
Node Id:         sp5en0
Class:           S
Type:            PERM
Resource Name:   haemd

Description
SOFTWARE PROGRAM ERROR

Probable Causes
```

```
SUBSYSTEM

Failure Causes
SUBSYSTEM

        Recommended Actions
        REPORT DETAILED DATA
        CONTACT APPROPRIATE SERVICE REPRESENTATIVE

Detail Data
DETECTING MODULE
LPP=PSSP,Fn=emd_cdb.c,SID=1.18.1.1,L#=637,
DIAGNOSTIC EXPLANATION
haemd(sp5en0): 2521-025 Cannot copy /etc/ha/cfg/em.sp5en0.cdb from CWS, return c
ode is 4.
```

This error is somewhat different than the previous. We check the staging area for any permissions problem as follows:

```
[sp5en0:/]# ls -l /spdata/sys1/ha/cfg
total 0
```

This is our problem. Even the version in the SDR was removed from the directory. One way of fixing this problem is to recreate the Event Management subsystem on the CWS. However, we will use a shortcut to rebuild the database. We first stop all the Event Management daemons as follows:

```
[sp5en0:/]# dsh -a 'stopsrc -s haem'
sp5n01: 0513-004 The Subsystem or Group, haem, is currently inoperative.
sp5n05: 0513-004 The Subsystem or Group, haem, is currently inoperative.
sp5n09: 0513-004 The Subsystem or Group, haem, is currently inoperative.
sp5n13: 0513-004 The Subsystem or Group, haem, is currently inoperative.
```

They were not running since there is no file left in the /spdata/sys1/ha/cfg directory. We then issue the following command on the CWS to rebuild the configuration database:

```
[sp5en0:/]# haemcfg
haemcfg: Reading Event Management data for partition: sp5en0.
haemcfg: Created EMCDB file: /spdata/sys1/ha/cfg/em.sp5en0.cdb Version:
944150447,703115008,0.
```

Then we start the Event Management daemons everywhere as follows:

```
[sp5en0:/]# startsrc -s haem.sp5en0
0513-059 The haem.sp5en0 Subsystem has been started. Subsystem PID is 25160.
[sp5en0:/]# dsh -a 'startsrc -s haem'
sp5n01: 0513-059 The haem Subsystem has been started. Subsystem PID is 18366.
sp5n05: 0513-059 The haem Subsystem has been started. Subsystem PID is 10480.
sp5n09: 0513-059 The haem Subsystem has been started. Subsystem PID is 16528.
sp5n13: 0513-059 The haem Subsystem has been started. Subsystem PID is 16898.
```

Finally, we check the Event Management subsystem by using the `lssrc` command as follows:

```
Subsystem         Group            PID     Status
 haem.sp5en0      haem             25160   active

Trace flags set:  None
```

```
  Configuration Data Base version: 944150447,703115008,0(SDR)

  Daemon started on 12/02/99 at 11:01:24.525047296
     running 0 days, 0 hours, 1 minutes and 32 seconds
  Daemon connected to group services: TRUE
  Daemon has joined peer group:       TRUE
  Daemon communications enabled:      TRUE
  Peer count:                         3

  Peer group state:
          944150447,703115008,0
 ...
```

It looks fine now. However, it seems that one node is missing from the Peer count value. The Peer count value represents the number of daemons currently in the ha_em_peers group beside this one (the daemon responding to the `lssrc` command). To investigate this, we query Group Services to get the providers list for the ha_em_peers group as follows:

```
[sp5en0:/]# hagsgr -s hags.sp5en0 -a ha_em_peers
  Number of: groups: 5
Group name[ha_em_peers] group state[Inserted |Idle |]
Providers[[1/0][1/1][1/13][1/9]]
Local subscribers[]
```

From the output, we see that node 5 is missing from the providers list. Let us check the Event Management subsystem on node 5. We issue the following command on node 5:

```
[sp5n05:/]# lssrc -ls haem
0513-036 The request could not be passed to the haem subsystem.
Start the subsystem and try your command again.
```

It is not running. We check for errors as follows:

```
[sp5n05:/]# errpt -N haemd -a
LABEL:          HA002_ER
IDENTIFIER:     12081DC6

Date/Time:       Thu Dec  2 11:02:08
Sequence Number: 75
Machine Id:      00010023A400
Node Id:         sp5n05
Class:           S
Type:            PERM
Resource Name:   haemd

Description
SOFTWARE PROGRAM ERROR

Probable Causes
SUBSYSTEM

Failure Causes
SUBSYSTEM

        Recommended Actions
        REPORT DETAILED DATA
        CONTACT APPROPRIATE SERVICE REPRESENTATIVE
```

```
Detail Data
DETECTING MODULE
LPP=PSSP,Fn=emd_cdb.c,SID=1.18.1.1,L#=637,
DIAGNOSTIC EXPLANATION
haemd: 2521-025 Cannot copy /etc/ha/cfg/em.sp5en0.cdb from CWS, return code is 3
.
```

This is a similar problem we had on the CWS. We then check the Event Management subsystem start up program error log file as follows:

```
sp5n05:/var/ha/log]# cat em.default.sp5en0
rcmdtgt:  2502-052 Error getting service ticket for rcmd.sp5n05@SP5EN0
2504-075 Can't write Kerberos ticket file.
rcmdtgt:  2502-052 Error getting service ticket for rcmd.sp5n05@SP5EN0
2504-075 Can't write Kerberos ticket file.
rcmdtgt:  2502-052 Error getting service ticket for rcmd.sp5n05@SP5EN0
2504-075 Can't write Kerberos ticket file.
rcmdtgt:  2502-052 Error getting service ticket for rcmd.sp5n05@SP5EN0
2504-075 Can't write Kerberos ticket file.
rcmdtgt:  2502-052 Error getting service ticket for rcmd.sp5n05@SP5EN0
2504-075 Can't write Kerberos ticket file.
rcmdtgt:  2502-052 Error getting service ticket for rcmd.sp5n05@SP5EN0
2504-075 Can't write Kerberos ticket file.
rcmdtgt:  2502-052 Error getting service ticket for rcmd.sp5n05@SP5EN0
2504-075 Can't write Kerberos ticket file.
rcmdtgt:  2502-052 Error getting service ticket for rcmd.sp5n05@SP5EN0
2504-075 Can't write Kerberos ticket file.
rcmdtgt:  2502-052 Error getting service ticket for rcmd.sp5n05@SP5EN0
2504-075 Can't write Kerberos ticket file.
```

Well, this seems a Kerberos problem. However, given what we learned on Chapter 3, “Security environment” on page 167, tickets are cached and stored in /tmp by default; so, we check /tmp space as follows:

```
[sp5n05:/var/ha/log]# df -k
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4             8192      3476   58%     1041    26% /
/dev/hd2           307200     21396   94%    10397    14% /usr
/dev/hd9var         32768     23440   29%      353     5% /var
/dev/hd3            32768         0  100%      112     2% /tmp
/dev/hd1             4096      3920    5%       18     2% /home
```

/tmp if full. We clean up some of the mess in /tmp, and then we restart the Event Management daemon on node 5. We then check the daemon on node 5 as follows:

```
Subsystem         Group            PID     Status
 haem             haem             12212   active

Trace flags set:  None

 Configuration Data Base version: 944150447,703115008,0(SDR)

 Daemon started on 12/02/99 at 11:15:57.785500639
    running 0 days, 0 hours, 0 minutes and 10 seconds
 Daemon connected to group services: TRUE
 Daemon has joined peer group:       TRUE
 Daemon communications enabled:      TRUE
 Peer count:                         4
 Peer group state:
        944150447,703115008,0
```

So, we have solved the Event Management problem. All the RSCT subsystems are up and functional. But, what about host responds? Let us check host responds as follows:

```
[sp5en0:/]# SDRGetObjects host_responds
node_number  host_responds
          1            1
          5            0
          9            1
         13            1
```

Still, no host responds for node 5. However, since host responds is a client of Event Management, we need to restart the host responds daemon on the CWS after we have restarted the Event Management daemon. We issue the following command on the CWS:

```
[sp5en0:/]# startsrc -s hr.sp5en0
0513-059 The hr.sp5en0 Subsystem has been started. Subsystem PID is 23916.
```

And then, we check the host responds daemon as follows:

```
[sp5en0:/]# lssrc -ls hr.sp5en0
Subsystem         Group            PID     Status
 hr.sp5en0        hr               23916   active
   Nodes_known = 4, up = 4, clients = 0, mode = heartbeat
   bind = 192.168.5.150
```

It looks fine. We then check the host responds bit in the SDR as follows:

```
[sp5en0:/]# SDRGetObjects host_responds
node_number  host_responds
          1            1
          5            1
          9            1
         13            1
```

We have solved the host responds problem.

For a listing of other host responds problems, refer to Appendix D, “43 reasons why host responds is red” on page 429.

Chapter 5. Diagnosing SP Switch problems

In this chapter, we describe how to diagnose switch-related problems. This chapter is divided into concepts, methodology, and examples. We first describe, in brief, the components of the SP Switch and how they work together. Then, we develop a methodology for problem determination. Finally, we offer real-life case studies to illustrate the concepts and methodologies.

The scope of the chapter does not include the detailed operation of the SP Switch. For details about the SP Switch, see *Understanding and Using the SP Switch*, SG24-5161.

5.1 SP Switch system

For successful problem determination of the SP Switch, the system administrator needs to understand what hardware and software components are involved in an SP Switch system. Also, it is necessary to understand how the functional units interact with each other. In this section, we first describe the hardware components of the switch, and then we discuss how the switch is started.

5.1.1 SP Switch board

In an SP frame, there is a switch board, which is the basic module of the SP Switch. The components of the switch board are the switch chip, the supervisor card, power supplies, and the cooling fans. In addition, there are a number of hardware components that create the complete switch board. From outside the switch looks like a black box with 32 ports. Out of these, 16 ports are used for connecting nodes, and the other 16 ports are used for connecting other switch boards in different frames. So, it is possible to create bigger networks using several interconnected switch boards.

The heart of the switch board is the switch chip. There are eight switch chips on a switch board. Each switch chip is interconnected with other chips through bi-directional redundant paths. There are eight ports on a switch chip for sending and receiving data simultaneously. Figure 136 on page 274 shows the logical representation of a switch chip.

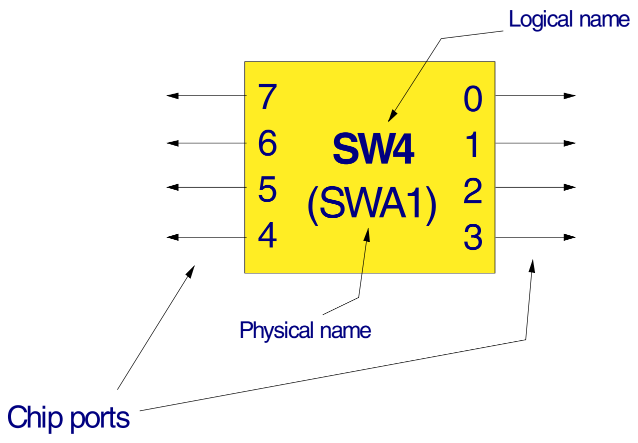


Figure 136. Logical diagram of a switch chip

These switch chips are divided into two interconnected stages; so, four chips are used to communicate with nodes, and the other four are used to extend the network to other switch boards on different frames. A logical layout of the chips on the switch board is illustrated in Figure 137 on page 275

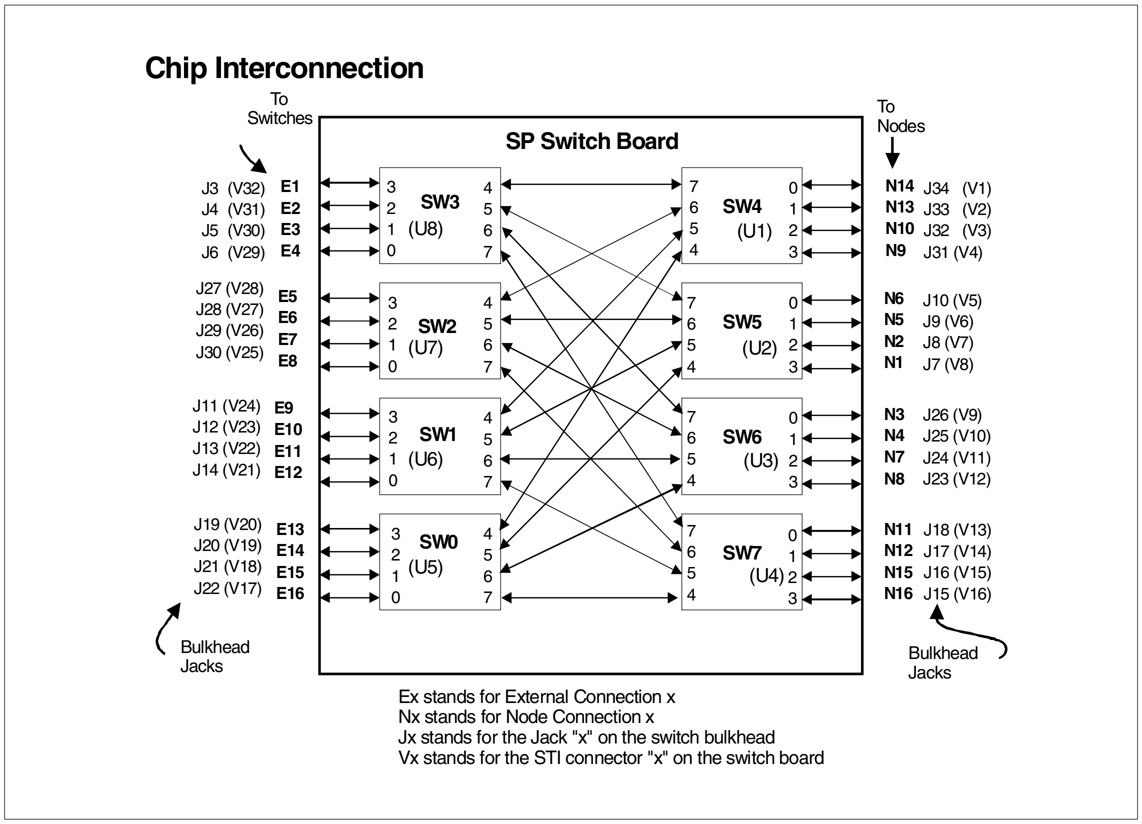


Figure 137. SP Switch board logical layout

The switch chip is responsible for routing data from one link to another. The internal structure of the switch chip is able to route data among all its output ports in parallel. The incoming data bytes are sent to the next network stages as soon as possible. If two data flows traverse the same chip in parallel, and must use the same port, one of them is stopped. For this reason, there are queueing facilities available on the chips. Other than routing data, the switch chip is designed to detect and isolate network and chip failures to maintain reliability. If a link failure occurs, the switch chip first handles it and takes the recovery action to resume normal link operation.

The function of the switch supervisor card is to monitor the hardware environment of the switch board constantly. It also receives the configuration messages from the frame supervisor card. The switch supervisor card, in turn, is connected to the frame supervisor card to provide the following functions on the board:

- Board clock selection: The SP Switch is a synchronous system. It is required to maintain clock synchronization across all the switch chips, switch adapters, and switch boards. Through the supervisor card, we can configure which internal oscillator will be used as a clock signal.
- Board status monitoring: The supervisor card provides the value of the variables that can be displayed using the `spmon` and `hmmon` commands.
- Board configuration sensing: The supervisor card detects whether an entry level switch with eight input ports and eight output ports or a full switch with 32 input ports and 32 output ports is in use.
- Fan rotation sensing: Five cooling fans are monitored by the supervisor card to ensure proper cooling for normal operation. If the cooling is not adequate, the supervisor card will turn off the power of the switch board assembly.
- Monitoring the power supplies: Power supplies are also monitored and controlled by the supervisor card.
- Board level sensing: There are different levels of switch boards available. The supervisor card is able to detect the board level. This information is important for level-specific software.

5.1.2 SP Switch adapter

This adapter is located in each node. It is the device through which data is passed from node to switch chips on the switch board. Each switch adapter is connected to one of the switch ports. There are different kinds of adapters depending on the bus type of the RS/6000. Figure 138 on page 277 shows the logical internal structure of a TB3 adapter.

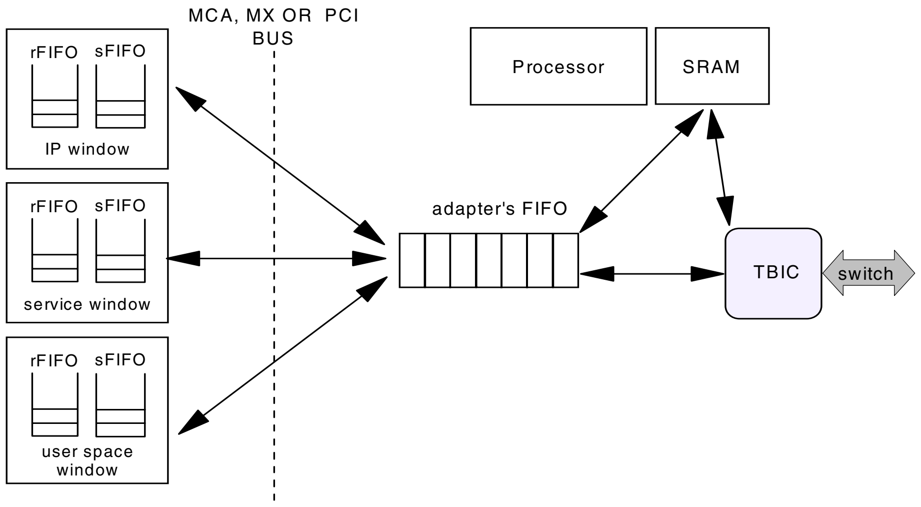


Figure 138. Logical structure of an SP Switch adapter

On an RS/6000 node, data is sent and received through some memory areas known as *windows*. If User Space protocol is used for data transfer, then an application puts the data on windows. The switch adapter microcode collects the data from the windows using the DMA engine of the adapter. After that, another DMA engine on the adapter moves data to and from the FIFO and TBIC chip. The TBIC chip on the switch adapter is responsible for sending and receiving data from the switch chip.

The microcode is responsible for providing the correct routing information for outgoing and incoming data packets. It uses a set of tables, which contains routes to every node in the system. This routing table is stored in the SRAM of the adapter. Based on the information provided by the primary node, the *fault service daemon* creates the routing tables and updates it continuously for the adapter.

For detailed description of the switch adapter internal structure, see *Understanding and Using the SP Switch*, SG24-5161.

5.1.3 SP Switch initialization

Now, we describe how the PSSP software switch components initialize the switch. The SP Switch initialization can be divided into three steps:

1. The switch adapter gets configured on the nodes.
2. The fault service daemon starts up on the nodes.
3. The `Estart` command is issued from the Control Workstation (CWS) to start the switch, or the switch admin daemon starts the switch automatically.

There are two boot phases when a node is powered on or rebooted. During the boot phase 2, `cfgmgr` executes the `cfgtb3` configuration method. The `cfgmgr` command takes the following actions:

- The css0 device is defined in the ODM CuDv class.
- The switch adapter's physical location is verified.
- The /dev/css0 file is created in the /dev directory.
- The device driver cssdd3 is loaded.
- Adapter-dependent microcode is loaded.
- The adapter device becomes available in the ODM CuDv class.
- Power-on Self Test (POST) is run for the adapter.

After the successful completion of the configuration of the switch adapter, the ODM CuAt class is updated. Figure 139 shows output of the command `odmget -q attribute=adapter_status CuAt`. In this figure, the adapter_status value is css_ready, indicating that the adapter is configured successfully.

```
# odmget -q attribute=adapter_status CuAt
CuAt:
        name = "css0"
        attribute = "adapter_status"
        value = "css_ready"
        type = "R"
        generic = "D"
        rep = "s"
        nls_index = 10
```

Figure 139. ODM value after the SP Switch adapter is configured

Updating the ODM's CuAt class is an intermediate step. When the network becomes available, the ODM value is used to update the SDR switch_responds class. If the configuration is successful, then the value of the adapter_config_status field in the switch_responds class is set to css_ready. Figure 140 on page 279 shows the output of `SDRGetObjects switch_responds`. In this figure, the values under the column adapter_config_status are all css_ready, indicating that switch adapters on all the nodes are properly configured.

```
[sp5en0:/] SDRGetObjects switch_responds
node_number switch_responds autojoin     isolated
adapter_config_status
          1           1           1           0 css_ready
          5           1           1           0 css_ready
          9           0           1           1 css_ready
         13           0           1           1 css_ready
```

Figure 140. SDR value of the adapter_config_status attribute

Once the adapter is configured, the next step is to start the fault service daemon on the nodes. The *rc.switch* script is executed from the inittab file, and it starts the fault service daemon, which is commonly known as *Worm*.

Worm performs different roles in supporting the switch. The role that the Worm daemon assumes will define the type of node it is running on. These are:

- Primary Node: The Worm daemon functions are as follows:
  - Process switch commands.
  - Initiate switch recovery for node and link failure.
  - Scan the switch network periodically to protect the Worm database against error/status packets.
  - Detect and recover from the loss of primary backup node.
  - Distribute the topology file.
- Secondary Node: The Worm daemon functions of secondary nodes are as follows:
  - Build the optimal database based on topology file.
  - Modify the database based on information received from the primary node.
  - Call the route table generator to build the routes from this node to all other nodes.
  - Tell the switch protocol (IP and User Space) which destinations are (or are not) available.
  - Load the new routes down to the adapter.
- Primary Backup Node: The Worm daemon functions of the primary backup node are as follows:
  - Detect the loss of the primary node and initiate takeover.
  - Function as any secondary node.

The fault service daemon, besides putting entries in the AIX error log, generates several log files, such as daemon.stderr, daemon.stdout, worm.trace, fs_daemon_print.file, cable_miswire, out.top, and flt. The flt file is the most important file generated by the daemon. It contains a summary of major events and errors encountered by the daemon.

The final step is issuing the `Estart` command from the CWS. The `Estart` command, at the beginning, does some initial checking as follows:

- The command checks the hardware clock synchronization for all the nodes.
- It confirms if the oncoming primary and oncoming primary backup nodes are up.
- It checks if the fault service daemon is running on both the oncoming primary and the backup nodes.
- It checks if the oncoming primary node is fenced or not.

Important

A new switch admin daemon, named *cssadm*, was introduced in PSSP 3.1 to automate the switch start-up process. This daemon monitors the nodes and the switch adapter events in all partitions and responds with an automatic `Estart` whenever required. In PSSP 3.1 or later, there is no need for manual intervention or writing a script to issue the `Estart` command. This new daemon runs on the CWS.

After the initial checks are completed, the `Estart` command starts another script called `Estart_sw` on the oncoming primary node to distribute the *switch* topology file to the nodes. Each time the `Estart` command runs on the CWS, it matches the number of nodes with switch adapters to the value of the num_nodes_success attribute in the switch_partition SDR class. If the values do not match, the topology file is distributed to all the nodes.

The `Estart` command uses the `pcp` command to distribute the topology file to the nodes. The distribution process creates a log file on the primary node called /var/adm/SPlogs/css/dist_topology.log.

After the topology file is distributed to all nodes, the message `Switch initialization started on <primary node>` is displayed on the CWS.

On the primary node, the fault service daemon first initializes the switch and creates the act.top.<pid> file. The `Estart_sw` script uses the act.top.<pid> file to update the SDR. From the content of the act.top.<pid> file, the `Estart_sw` script updates the SDR with the current primary and primary backup nodes and the number of uninitialized links. Then, it renames the act.top.<pid> file to topology.data.

After the topology.data file is created, Worm code is executed by the fault service daemon on the primary node to find out the actual topology of the switch fabric. Then, the Worm daemon initializes all the functional chips and nodes with run time parameters. At this point in time, the Worm uses the *expected topology* file to discover the switch chips and nodes, those that are part of the switch fabric. In this stage, if Worm on the primary node fails to contact a chip or a node, it considers the link to that component to be faulty and the corresponding chip port is disabled. Wrap-plugs that should not be there are detected. While finding the actual topology file, Worm also detects whether or not any miswire is present in the switch fabric.

Once the actual topology of the network is known to the Worm, it generates the *out.top* file in the primary node. The out.top file contains the expected topology file with annotations describing the status of links and devices as detected by the Worm. After creating the out.top file on the primary node, the Worm sends information about the links that were found to the nodes. The nodes use that information to build the out.top file by updating the expected topology file. The update will fail, and the node will be fenced if the expected topology file does not exist on that node.

When the secondary node manages to create the out.top file, it sends an acknowledgment to the primary node. If the primary node does not receive the acknowledgment from the node, it knows that there is some problem.

Now, all the participating nodes have the actual topology file; so, they generate the four best routes between each pair of nodes. Worm on the primary node sends a “load Routes” message to all other nodes. Each node then loads the routes into the adapter and sends another acknowledgment back to the primary node. This completes the switch initialization.

Here, we described the switch initialization process from a very high level. For detailed information on switch initialization, see *Understanding and Using the SP Switch*, SG24-5161.

5.2 Prerequisites for successful SP Switch operation

During the installation phase of the CWS, some files and certain tuning parameters are set for the initialization of the switch. If the configuration files and tuning parameter values are not set correctly, it is not possible to start the switch. Also, in the course of time, if the initial settings get changed or the files become corrupted, switch failure problem determination becomes difficult. Therefore, before starting problem determination, it is important to check if all these settings are correct.

5.2.1 Verification of SP Switch software installation

There is a script called `CSS_test` located in the /usr/lpp/ssp/bin directory. If this script is run from the CWS, it checks the fileset level for all the nodes. It also checks the consistency of the fileset.

In addition, the `CSS_test` script checks the following:

- It uses `ping` to check the Ethernet interface of all the nodes and reports if any problem is found.
- It checks if there is valid Kerberos ticket for all the nodes.
- It uses `ping` to check the switch IP address and, in case of a problem, reports the failure.

This script can be used as the first problem determination tool to make sure the fileset is not broken, Kerberos is functional, and nodes are reachable over the Ethernet interface. We can run the script any time without interrupting the normal operation of the system.

Figure 141 on page 283 shows the output of `CSS_test` running on an SP with four high nodes.

```
[sp5n09:/usr/lpp/ssp/bin]# CSS_test

============================================================================
CSS_test:  CSS Installation Verification Test started on
                   Wed Mar 3 17:05:15 EST 1999.
----------------------------------------------------------------------------
CSS_test:  Beginning Ethernet IP test of the nodes in partition sp5en0.
----------------------------------------------------------------------------
CSS_test:  Show LPP ssp.basic installation levels:  (lslpp -Lq ssp.basic)

        -Node-          -LPP Name-  -Installation Level-
        sp5n01:         ssp.basic       3.1.0.0
        sp5n05:         ssp.basic       3.1.0.0
        sp5n09:         ssp.basic       3.1.0.0
        sp5n13:         ssp.basic       3.1.0.0
        sp5n09:         ssp.basic       3.1.0.0

CSS_test:  Show LPP ssp.css installation levels:  (lslpp -Lq ssp.css)

        -Node-          -LPP Name-  -Installation Level-
        sp5n01:         ssp.css         3.1.0.0
        sp5n05:         ssp.css         3.1.0.0
        sp5n09:         ssp.css         3.1.0.0
        sp5n13:         ssp.css         3.1.0.0
        sp5n09:         ssp.css         3.1.0.0

CSS_test:  Show inconsistent ssp.css files for relevant nodes:  (lppchk)

 -Node-         -File Name-                     -Actual Size-   -Expected Size-

----------------------------------------------------------------------------
CSS_test:  Beginning Switch IP test of the nodes in partition sp5en0.
CSS_test:  Following nodes failed Switch IP test:
----------------------------------------------------------------------------
CSS_test:  CSS Installation Verification Test completed on
                   Wed Mar 3 17:05:23 EST 1999.
```

Figure 141. Results of CSS_test

5.2.2 Verification of the SP Switch topology file

The switch topology file is the textual representation of the SP Switch network layout, and during the installation or reconfiguration, this file must be selected from the /etc/SP directory. This directory only contains links for the files located in /spdata/sys1/syspar_configs.

The selection of the topology file depends on how many node switch boards (NSBs) and intermediate switch boards(ISBs) are there in the frames. The naming convention for the switch topology file is as follows:

```
expected.top.<NSBnum>nsb.<ISBnum>isb.type
```

If there is only one SP frame with a single switch board, the selected topology file should be *expected.top.1nsb.0isb.0*. During the installation of the CWS, the topology file must be annotated and stored in the SDR using the `Eannotator` command (or `smitty annotator`).

The `Etopology` command saves the topology file into the SDR. To check the name of the topology file, issue the `SDRGetObjects` command as follows:

```
SDRGetObjects Switch_partition topology_filename
```

```
[sp5en0:/etc/SP] SDRGetObjects Switch_partition topology_filename
topology_filename
expected.top.last.3
```

Figure 142. SDRGetObjects command for topology_filename attribute

Figure 142 shows the name of the topology file saved in the SDR of our CWS. On the nodes, make sure that the same topology file exists in the /etc/SP directory.

The `Estart` command distributes the topology file to the nodes using the `pcp` command and copies the file to the /etc/SP directory as *expected.top.out.<num>*. It is a good practice to make sure the right topology file is stored in the SDR and to the nodes.

To verify that the topology file in the SDR is correct. It should be read out of the SDR by issuing the `Etopology` command as follows:

```
Etopology -read <output file>
```

The `Etopology` command with `-read` option reads the switch topology from the SDR and places it into the specified file. You will now be able to read that file and check that the information is an accurate representation of the installed hardware setup.

If changes to the switch topology file are required, remember to place them back into the SDR after the changes have been completed by using the `Eannotator` command.

5.2.3 Verification of the SP Switch clock topology file

The clock topology file is the file that identifies the master switch board and determines how the clock is distributed to all other switch boards. The master switch board generates the synchronous clock using an internal oscillator that drives the whole switch fabric. Each of the other boards, called *slave boards*, gets the clock directly or indirectly from the master board through a switch-to-switch cable.

The clock topology file is also selected during the time of installation depending on the hardware configuration of the system. The `Eclock` command is used to specify the clock topology file and store the distribution file in the Switch SDR class. To verify that the right clock settings are stored in the SDR, we can run the following command:

```
SDRGetObjects Switch switch_number clock_input clock_source
```

Figure 143 shows the output for our four-node SP system.

```
[sp5en0:/] SDRGetObjects Switch switch_number clock_source clock_input
switch_number clock_source clock_input
          1            0 0
```

Figure 143. Examples of SDRGetObjects for clock source on a single board

To validate the clock settings stored in the SDR, we run the `spmon -G -d` command to read the frame information. An example output for the same system is shown in Figure 144. We can see that the value of clock_source of Figure 143 matches that of Figure 144.

```
4. Checking frames

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

Figure 144. Examples of spmon output for validating the clock source

You can also generate a clock topology file out of the information stored in the SDR. To generate a clock topology file, we use:

```
Eclock -c /tmp/Eclock.top
```

This command will create a clock topology file called Eclock.top and place it in the /tmp directory.

5.2.4 Verification of primary and primary backup

We already mentioned in 5.1.3, “SP Switch initialization” on page 277 that the `Estart` command needs one of the nodes to be selected as the primary node to initialize the switch fabric. For reliability, and to avoid having a single point of failure, we also need to select another node as the primary backup node.

To check if the primary and the primary backup are selected, run the `Eprimary` command without any option. Figure 145 shows the sample output of the `Eprimary` command.

```
1       - primary
1       - oncoming primary
5       - primary backup
5       - oncoming primary backup
```

Figure 145. A sample of Eprimary output

If the primary or the primary backup field is none, select those nodes as follows:

```
Eprimary <node Number> -backup <node number>
```

Also, remember that until the next `Estart` is issued, the nodes specified in the `Eprimary` command are referred to as the oncoming primary and the oncoming primary backup.

5.2.5 Verifying the inittab entry

The local /etc/inittab for each node should contain an entry for the rc.switch file similar to the following:

```
fsd:2:once:/usr/lpp/ssp/css/rc.switch
```

This can be checked using the `dsh` command, as follows, for all nodes:

```
dsh -w <reliable_hostname> grep rc.switch /etc/inittab
```

Attention

Prior to PSSP 3.1, setting up applications to use the switch network must be done by writing a separate script to start the switch and checking that it is available before starting the application. In PSSP 3.1 or later releases, setting up applications to use the switch network takes just two steps:

1. Change the rc.switch entry in the /etc/inittab from once to wait.

   ```
   fsd:2:wait:/usr/lpp/ssp/css/rc.switch
   ```

2. Put the application startup script after rc.switch.

Remember this option is only available on PSSP 3.1 or later releases.

5.2.6 Verify the System Data Repository (SDR)

To verify that the SDR is installed and operating correctly, you can run `SDR_test` on the CWS. It is run either through SMIT or from the command line.

To verify the SDR from the command line, enter:

```
/usr/lpp/ssp/bin/SDR_test
```

Figure 146 shows the message displayed when `SDR_test` runs successfully on the CWS.

```
sp5en0:/etc/SP] SDR_test
SDR_test: Start SDR commandline verification test
SDR_test: Verification succeeded
```

Figure 146. SDR_test results

Next, log in to 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, while value of 0 indicates a node is non-operational.

The *SDR_dest_info* file on all nodes should be checked to see if the file has a correct entry for the IP address and the hostname associated to the default and primary partitions. Figure 147 shows a sample of the SDR_dest_info file.

```
# more /etc/SDR_dest_info
default:192.168.5.150
primary:192.168.5.150
nameofdefault:sp5en0
nameofprimary:sp5en0
```

Figure 147. A sample of the SDR_dest_info file

5.3 Type of SP Switch failures

Switch failure may occur because of a faulty switch chip, switch adapter, cable, incorrect or corrupted configuration files, or broken software. For quick problem determination to reduce the down time, the best approach is to understand what component is failing.

Use the summlog file on the CWS to get a global picture of the switch fabric problem. See 5.4.2.1, “The summlog file” on page 296 for more about the summlog file.

Important

Be aware that the summlog log file will contain entries for nodes running at PSSP 3.1 or later. Nodes running previous PSSP versions do not send AIX error log information.

If it is suspected that the failing component is the switch chip, log in to the primary node. If the node adapter failure is suspected, log in to the suspected node and check the AIX error log as follows:

```
errpt |more
```

Figure 148 on page 288 shows the sample error report on a primary node. The resources name (Res Name) in the error log should give you an indication of how the failure occurred. Also, the class column (C) tells if it is a hardware error or a software error.

```
# errpt|more
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
89F34F83   1006212198 P S Worm           Switch Fault Service Daemon
Terminated
D84D68A7   1006212198 T U Worm           Estart failed
EC771C6B   1006154998 T S css           Switch daemon SDR communications
failed
A388E1C3   1006154998 I S Worm           Links not initialized during
Estart
B1531818   1006154998 I S Worm           Nodes not initialized during
Estart
89F34F83   1006153898 P S Worm           Switch Fault Service Daemon
Terminated
D84D68A7   1006153898 T U Worm           Estart failed
89F34F83   1006153098 P S Worm           Switch Fault Service Daemon
Terminated
D84D68A7   1006153098 T U Worm           Estart failed
```

Figure 148. A sample error report output

For a complete list of error labels and their descriptions, refer to *IBM Parallel System Support Program for AIX Diagnosis Guide*, GA22-7350. If a hardware error is detected, you should contact your local hardware support for hardware diagnosis.

If the problem is not obvious from the error log analysis, or you think that the problem is due to software components, follow one of the problem determination flow charts we have developed from our experience.

In most of the cases we have seen, there are two types of problems related to the switch: Single node failure or multiple nodes failure. We have flowcharts for both type of problems. The flowcharts also guide you to the specific sections mentioned in Table 6.

Table 6. Specific problems related to the SP Switch

Problem description	**Sections**
Switch adapter configuration problem	See 5.4.4, "Diagnosing SP Switch adapter problems" on page 301.
The `Eunfence` command failure	See 5.4.5, "Diagnosing Eunfence command problems" on page 304.
The `Estart` command failure	See 5.4.6, "Diagnosing Estart problems" on page 306.
Clock synchronization problems	See 5.4.7, "Diagnosing clock synchronization problems" on page 314.

5.3.1 Flowchart for single node switch failure

Figure 149 on page 290 and Figure 150 on page 291 show the flowchart for a single node switch failure problem. The flow chart starts from very basic checking and proceeds through step-by-step checking of all the probable contributing factors for switch problems.

In a problem situation, the flowchart can be followed from the very beginning, referring to the sections mentioned in the flowchart for answers to the questions asked within the chart itself. Another approach is to use the flow chart as a quick checklist so that nothing is left untried before contacting the IBM Support Center.

The assumption for the flowchart is that the switch is up and running, but suddenly a switch failure occurs on a node, or that a new node is being installed in the frame but the switch will not come up on this new node. The flow chart helps to bring the switch up on that node.

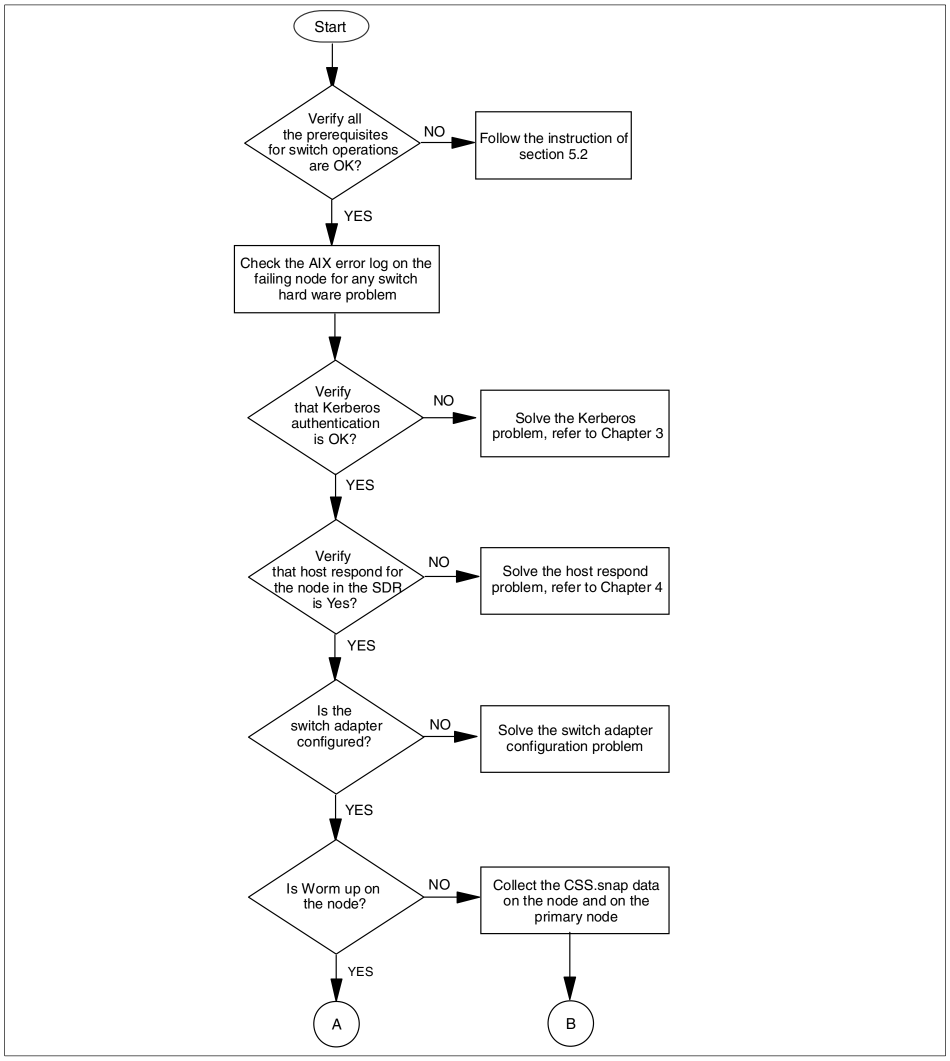

Start
Verify all the prerequisites for switch operations are OK?
NO
Follow the instruction of section 5.2
YES
Check the AIX error log on the failing node for any switch hard ware problem
Verify that Kerberos authentication is OK?
NO
Solve the Kerberos problem, refer to Chapter 3
YES
Verify that host respond for the node in the SDR is Yes?
NO
Solve the host respond problem, refer to Chapter 4
YES
Is the switch adapter configured?
NO
Solve the switch adapter configuration problem
YES
Is Worm up on the node?
NO
Collect the CSS.snap data on the node and on the primary node
YES
A
B

Figure 149. Single node switch failure flowchart (Part 1 of 2)

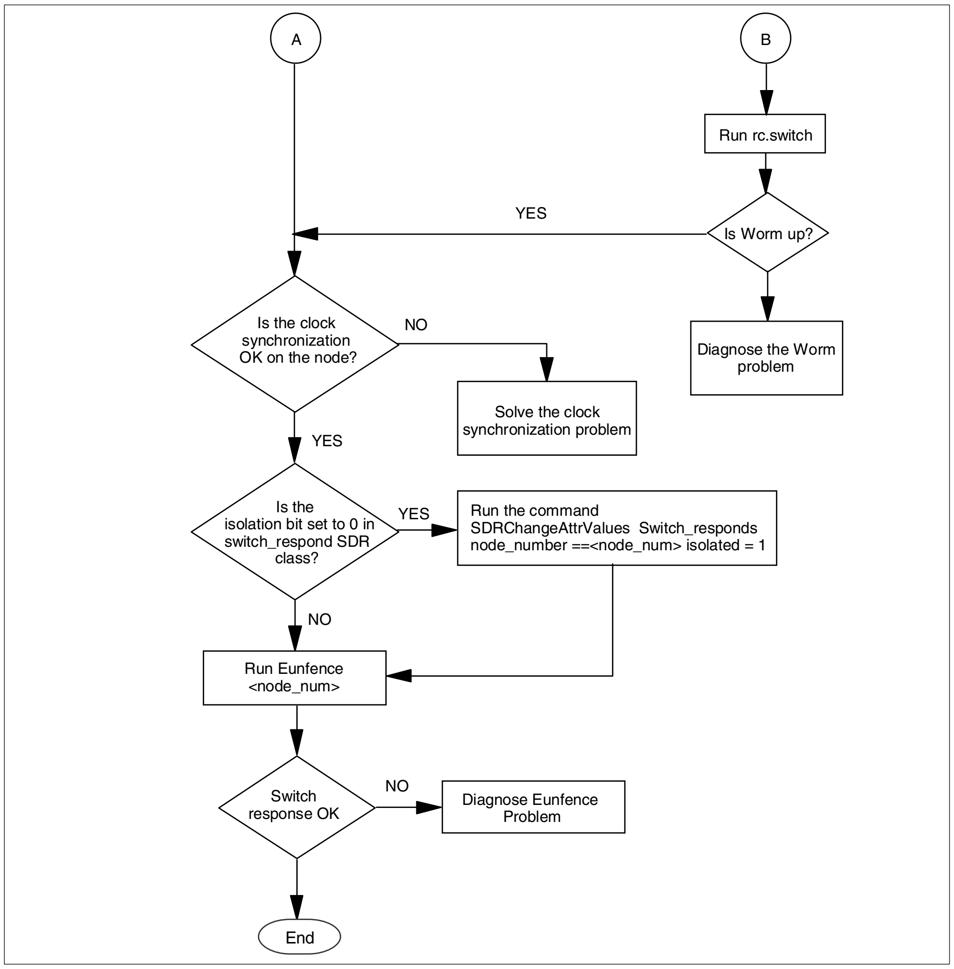

A
B
Run rc.switch
YES
Is Worm up?
Is the clock synchronization OK on the node?
NO
Diagnose the Worm problem
Solve the clock synchronization problem
YES
Is the isolation bit set to 0 in switch_respond SDR class?
YES
Run the command SDRChangeAttrValues Switch_responds node_number ==<node_num> isolated = 1
NO
Run Eunfence <node_num>
Switch response OK
NO
Diagnose Eunfence Problem
End

Figure 150. Single node switch failure flowchart (Part 2 of 2)

5.3.2 Flowchart for multiple nodes switch failure

Figure 151 to Figure 153 on page 294 show the flowcharts for problem determination of multiple node switch failure. These flowcharts can be used as a quick checklist by an experienced system administrator, or it can be followed step-by-step to solve the switch problems.

The assumption for the flowchart is that the switch is up and running, but some changes caused the switch to stop functioning properly on some of the nodes. Another situation that can lead to a multiple node switch problem at the time of installation is that something goes wrong causing some of the nodes to not be up with the switch. In either case, the flowchart helps to resolve the problem.

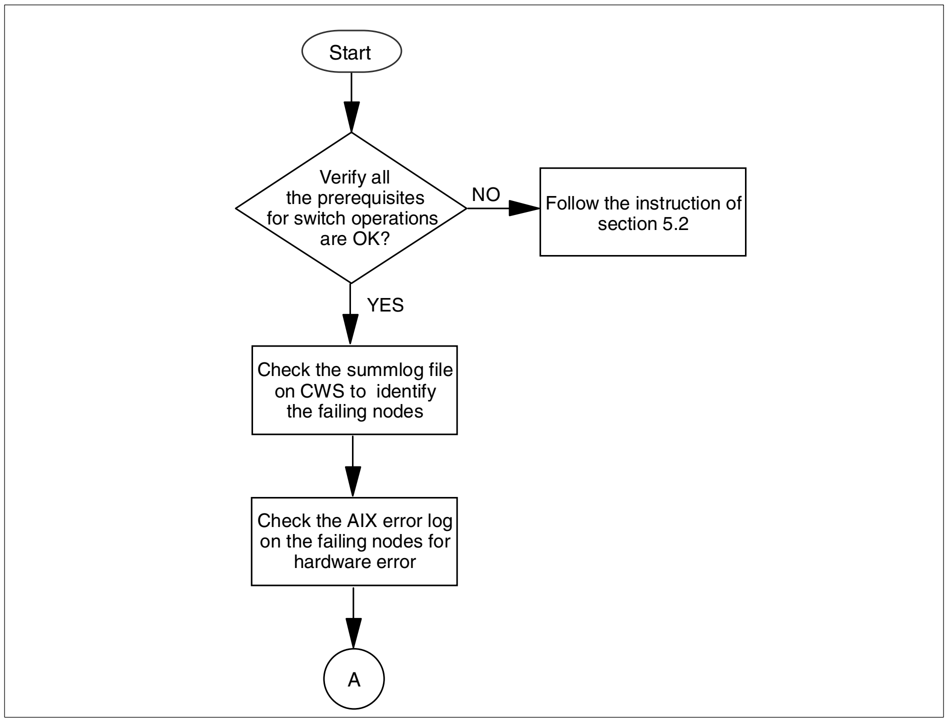

Start
Verify all the prerequisites for switch operations are OK?
NO
Follow the instruction of section 5.2
YES
Check the summlog file on CWS to identify the failing nodes
Check the AIX error log on the failing nodes for hardware error
A

Figure 151. Multiple nodes switch failure flowchart (Part 1 of 3)

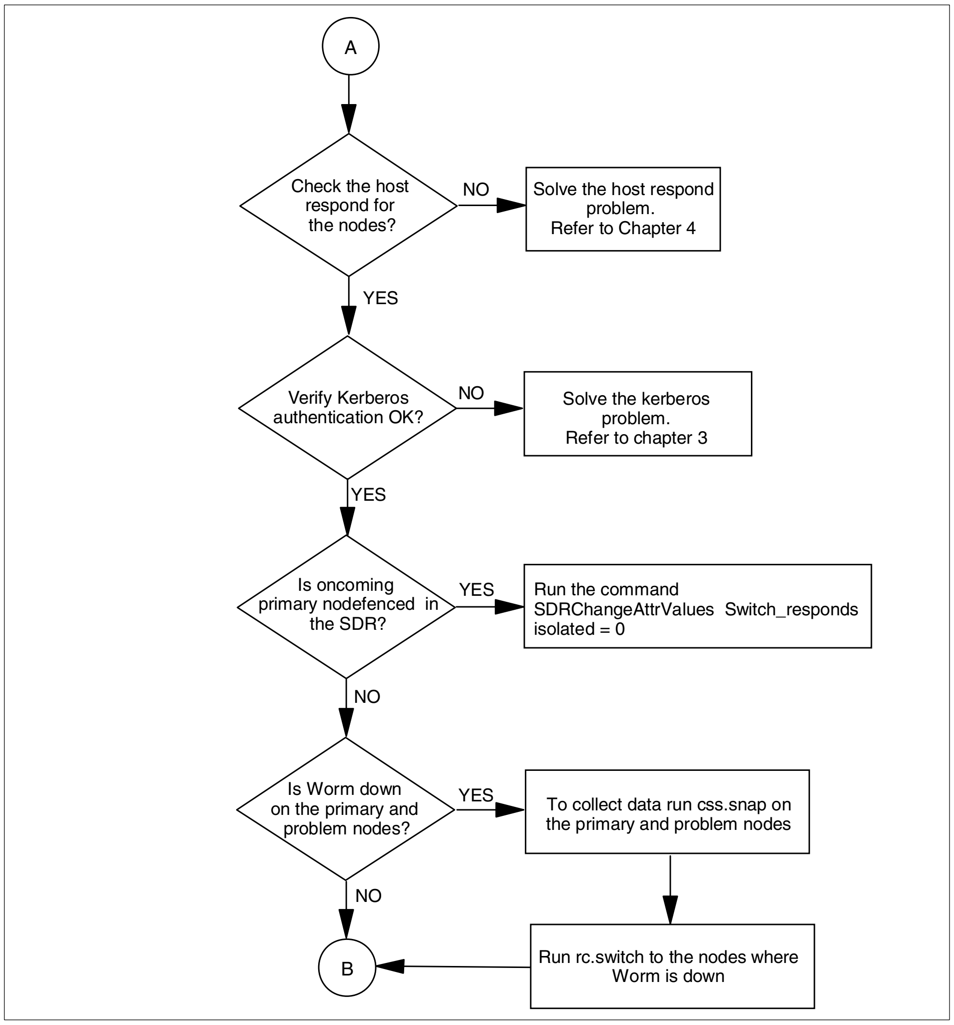

A
Check the host respond for the nodes?
NO
Solve the host respond problem. Refer to Chapter 4
YES
Verify Kerberos authentication OK?
NO
Solve the kerberos problem. Refer to chapter 3
YES
Is oncoming primary nodefenced in the SDR?
YES
Run the command SDRChangeAttrValues Switch_responds isolated = 0
NO
Is Worm down on the primary and problem nodes?
YES
To collect data run css.snap on the primary and problem nodes
NO
Run rc.switch to the nodes where Worm is down
B

Figure 152. Multiple nodes switch failure flowchart (Part 2 of 3)

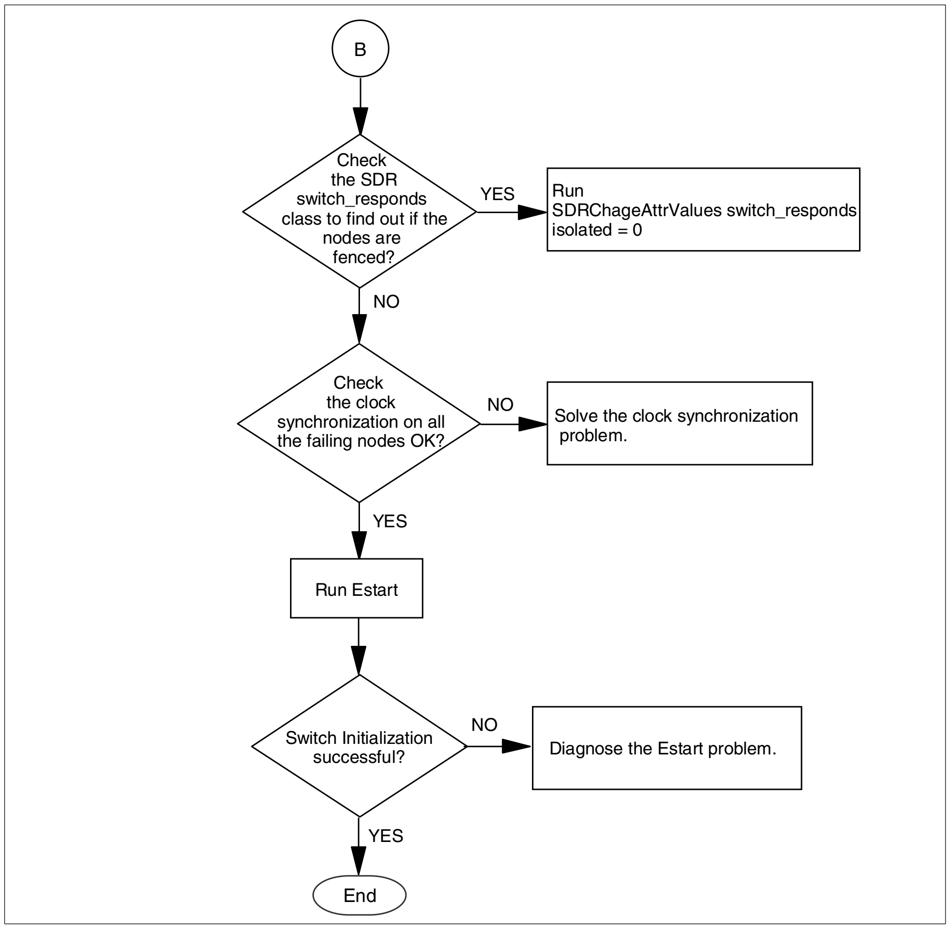

B
Check the SDR switch_responds class to find out if the nodes are fenced?
YES
Run SDRChageAttrValues switch_responds isolated = 0
NO
Check the clock synchronization on all the failing nodes OK?
NO
Solve the clock synchronization problem.
YES
Run Estart
Switch Initialization successful?
NO
Diagnose the Estart problem.
YES
End

Figure 153. Multiple nodes switch failure flowchart (Part 3 of 3)

5.4 Diagnosing switch commands and daemon Failures

Here, we describe how to collect data and how to solve the most common switch command failures. Some of the log files are also discussed to show how they can be used for problem determination.

5.4.1 Collecting SP Switch related data

The css.snap script collects log files created by switch support code, such as device drivers, the Worm, diagnostic outputs, and so on. The command is automatically called from the fault service daemon when certain serious errors are detected. It can also be issued from the command line when a switch or adapter related problem is indicated as follows:

```
#/usr/lpp/ssp/css/css.snap -s
```

We recommend you use the `css.snap` script with the `-s` option; otherwise, it might have a disruptive effect when used on a running system. If `css.snap` is used without any option, it is advisable to run the `rc.switch` script in order to reset or reload the switch adapter microcode and eliminate the residual effects of this utility.

The `css.snap` script should run on the primary node and also on the failing nodes. The script will create a compressed tar file in the /var/adm/SPlogs/css directory. The naming convention for the file is css.snap.<date-time>.tar.Z.

Under normal circumstances, the following files are collected:

- cable_miswire
- cable_miswire.old
- core
- css.snap.log
- daemon.stderr
- daemon.stdout
- dtbx.trace
- dtbx.failed.trace
- errpt.out
- flt
- fs_daemon_print.file
- netstat.out
- out.top
- rc.switch.log
- regs.out
- router.log
- scan_out.log
- scan_save.log
- tb_dump.out
- vdidl.out

- worm.out

We recommend that, if you experience a switch problem, you first run the `css.snap` script to collect the data and then start your problem determination procedure. The reason for this recommendation is that if you start issuing switch commands and *then* take the snapshot, the logs may not have the information of the original problem. Refer to 1.1, “Methodologies” on page 1 for more information on problem determination methodologies.

Before running the script, make sure that the /var file system has enough free space. For this script to run successfully, it needs at least 20 MB of free space in the /var file system.

5.4.2 SP Switch logs

In this section, we describe some of the logs collected by the `css.snap` script. Other logs are described in later sections where we discuss the detailed procedures for diagnosing the failures of different commands.

5.4.2.1 The summlog file

The summlog file is located on the CWS in the /spdata/sys1/ha/css directory, and though it is not collected by the `css.snap` script, this should be the starting point for problem determination. (Note, that it is only available for nodes running PSSP 3.1 or later.)

This log allows us to identify the nodes that are participating in an error event, and it shows the failure symptom on each node (see Figure 154 on page 296). The information of interest is as follows:

- The third column in the report shows whether or not css.snap dump is initiated.
- The sixth column in the report shows the error label of the event.

```
100213151998 sp5n05 N sp5en0 128 SP_SW_SDR_FAIL_RE
100213291998 sp5n13 Y sp5en0 153 TB3_LINK_RE
100213311998 sp5n13 N sp5en0 154 TB3_TRANSIENT_RE
100213341998 sp5n05 N sp5en0 130 SP_SW_SDR_FAIL_RE
100213341998 sp5n13 N sp5en0 155 HPS_FAULT6_ER
```

Figure 154. Sample of the summlog file on the CWS

Figure 154 shows an example of the summlog file when a problem occurred in one of the nodes of our four-node SP system. The second record shows that a css.snap was taken, and the last record shows that an `HPS_FAULT6_ER` is detected on node 13.

Refer to Table 7 for an explanation of the error labels in this figure. Refer to *PSSP Diagnosis Guide*, GA22-7350 for a complete description of the error labels.

Table 7. Explanation of Error Labels of summlog File

Error label	**Description**
SP_SW_SDR_FAIL_RE	The switch daemon failed to communicate with the SDR. Probable causes are an Ethernet overload, excessive SDR traffic, SDR daemon is down on the CWS.
TB3_LINK_RE	A switch adapter link outage occurred. Probable causes are, the node is fenced, on a loose, disconnected, or faulty cable.
TB3_TRANSIENT_RE	A switch adapter transient error occurred. A loose, disconnected, or faulty cable is the cause of the problem.
HPS_FAULT6_ER	The fault_service_Worm_RTG_SP daemon was terminated. Probable causes are bad switch adapter or missing external clock signal.

5.4.2.2 The flt file

This file is located on all nodes. However, useful information is logged in the primary or primary backup's flt files under the /var/adm/SPlogs/css directory.

The purpose of the flt file is to log hardware error conditions on the switch as well as recovery actions taken by the fault service daemon and general operations that alter the switch configuration.

Information that can be found in the flt file includes:

- Disabled switch chip, nodes, and ports.
- Switch initialization error status.
- Failures in service packet broadcasts.
- Switch initialization `(Estart)` as a recovery action.
- Switch initialization `(Estart)`as a command.
- Primary node takeover.
- Fence operations.
- Unfence operations.
- Primary node switch port is disabled.

- Node personality changes.
- Route generation.
- fault_service signal (SIGBUS, SIGTERM, SIGDANGER).
- Phase 2 of switch initialization being retrieved.
- Diagnosing Worm problems.

Figure 155 is an extract from the flt file on the primary node. The file is created by issuing the `Estart` command. The figure shows that the fault service daemon detected some problem on switch chip 5 at port 3. It tried to resolve the problem, but the recovery actions failed. As a result, it disabled the port 3 on switch chip 5 and also turned off the switch_responds bit in the SDR. This is an example of an unfence operation.

```
(i) 10/04/98 09:56:42: 2510-744 SP Switch error recovery initiated.
(i) 10/04/98 09:56:42: 2510-793 First Error Capture Register = 000000.
(i) 10/04/98 09:56:42: 2510-741 Second Error Capture Registers =
10000000 00000010 00000000 00000000 00000010 000000
(i) 10/04/98 09:56:42: 2510-740 Packet Sequence Number = 192 Switch
Time-Of-Day = 0x800003818ec959a8
(i) __Date__ __Time__ _Msgid__ Reg ___Location___ Ch Po
__Type_of_SECR_Error__
(n) 10/04/98 09:56:42: 2510-767 SEC E01-S17-BH-J07 5 3 Recv Link Sync
Failure
(n) 10/04/98 09:56:42: 2510-760 SEC E01-S17-BH-J07  5 3 Incorrect EDC
(n) 10/04/98 09:56:42: 2510-778 SEC E01-S17-BH-J07  5  3 Send Link Sync
Failure
(i) 10/04/98 09:56:42: 2510-759 Error count threshold has been
exceeded, initiating recovery action(s).
(n) 10/04/98 09:56:42: 2510-743 Disabling port 3 (jack 7) of chip 5 on
the switch in slot 17 of frame 1
(n) 10/04/98 09:56:42: 2510-749 Turning off switchResponds bits for
node 4 in the SDR
```

Figure 155. Example of the flt file

The log indicates the failing node is node 5. Therefore, to find the root cause of the failure, we can log in to node 5 and also look into other logs. Sometimes decoding the “error capture register” in the flt file should give you some idea of what type of errors the switch chip is detecting. Usually, only IBM software support people are able to decode these register values.

5.4.2.3 The out.top file

The fault service daemon reads the topology file and writes the link information to the /var/adm/SPlogs/css/out.top file on every node. The out.top file looks similar to a switch topology file except for the additional comments on lines where either the device or the link is not operational. These additional comments are appended to the file by the fault service daemon to reflect the current link status on the system.

Figure 156 shows the successful entry for node 5 in our four-node SP system. There should be no error messages for the installed nodes in the frame.

```
s 15 3 tb3 4 0           E01-S17-BH-J7 to E01-N5
```

Figure 156. Example of successful entry in the out.top file

Figure 157 shows the failing entry for the same node 5 when we had a switch problem with this node. The failing entry has a comment starting with a return code -4. Refer to Chapter 18 “Diagnosing Switch Problems” in *PSSP: Diagnosis Guide*, GA22-7350 for a complete description of these codes. The guide also lists the action to be taken based on the return code.

```
s 15 3 tb3 4 0          E01-S17-BH-J7 to E01-N5   -4 R: device has been
removed from network - faulty (link has been removed from network -
fenced)
```

Figure 157. Example of a failing entry in the out.top file

Figure 157 is read as follows:

- Switch chip 5, port 4, is connected to switch node number 5. The switch is located in frame E01 slot 17. Its bulkhead connection to the node is jack 7.
- -4 R refers to the device status of the right-side device, tb3 4.
- 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”.

5.4.3 The cable_miswire file

The cable_miswire file is also created when we run the `Estart` command on the CWS. If you suspect there is a cable miswire in the system, we suggest you check to see if a cable_miswire file has been created on the primary node.

As an example, to simulate the cable miswire we interchanged the switch cable of node 1 and node 5 in our four-node SP. We ran the `Estart` command to bring the switch up. The switch was up and was running, but each time we issued the `Estart` command, a cable miswire file was created on the primary node.

Figure 158 shows the cable miswire, problem for device ID 0 (node 1) and device ID 4 (node 5). In the figure, the message ID 2510-796 indicates that, according to the expected topology file, device ID 0 is connected to switch port 1, but actually it should be connected to switch port 3.

Similarly, device ID 4 is connected to switch port 3, but it should be connected to switch port 1. The Object Database Manager (ODM), while configuring the switch, identifies which port the node switch adapter is connected to. Later on, the Worm daemon always compares the value of the ODM with the expected topology file, and if it finds any mismatch, it reports the problem in the cable_miswire file.

```
(i) print_the_time_miswire: The date and time  = Mon Oct  5 16:16:14
1998
ProcessNodeMiswire: 2510-745 Node Miswire detected for Device id -
Expected = 0 Actual = 0.
ProcessNodeMiswire: 2510-794 Node Miswire detected for switch number -
Expected = 1 Actual = 1.
ProcessNodeMiswire: 2510-795 Node Miswire detected for switch chip -
Expected =5 Actual = 5.
ProcessNodeMiswire: 2510-796 Node Miswire detected for switch port -
Expected =1 Actual = 3.
ProcessNodeMiswire: Connection line:  s 15 1  tb3 0 0
E01-S17-BH-J9 to E01-N1

ProcessNodeMiswire: 2510-745 Node Miswire detected for Device id -
Expected = 4 Actual = 4.
ProcessNodeMiswire: 2510-794 Node Miswire detected for switch number -
Expected= 1 Actual = 1.
ProcessNodeMiswire: 2510-795 Node Miswire detected for switch chip -
Expected =5 Actual = 5.
ProcessNodeMiswire: 2510-796 Node Miswire detected for switch port -
Expected =3 Actual = 1.
ProcessNodeMiswire: Connection line:  s 15 3  tb3 4 0
E01-S17-BH-J7 to E01-N5

ProcessNodeMiswire: 2510-920 Node Miswire detected, Please check node
cabling
```

Figure 158. Sample of the cable_miswire file

5.4.4 Diagnosing SP Switch adapter problems

To diagnose switch adapter problems, the first step is to check if the switch adapter is configured on the node. To do this, you can run the following command:

```
[sp5en0:/]# SDRGetObjects switch_responds
node_number  switch_responds autojoin isolated adapter_config_status
          1            1            1            0 css_ready
          5            1            1            0 css_ready
          9            1            1            0 css_ready
         13            1            1            0 css_ready
```

The information to look for on the output is the value of the *adapter_config_status* attribute. It should be *css_ready*. If any other value is found for the adapter_config_status, find the message in Table 8 and take the appropriate action.

Table 8. adapter_config_status messages

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. Rerun the adapter configuration command.
lname_error	The device logical name specified on the CSS adapter configuration command was invalid. Rerun the adapter configuration command.
undefine_system_fail define_system_fail xilnx_system_fail	The System Standard C library subroutine failed during CSS adapter configuration. Rerun the adapter configuration 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, and then rerun the adapter configuration.
chkslot_fail	Verify that the CSS adapter is properly seated. Then, rerun the adapter configuration command.
busreslove_fail	There are insufficient bus resources to configure the CSS adapter. You need to contact the IBM Support Center.
xiilnx_load_fail dd_load_fail fs_load_fail	See 5.2.1, “Verification of SP Switch software installation” on page 282. If the software verification is successful but the problem persists, contact the IBM Support Center.
make_special_fail	The CSS device special file could not be created during the adapter configuration. Rerun the adapter configuration command.
dd_config_fail fs_init_fail	An internal device driver error occurred during the CSS adapter configuration. Run css.snap on the node and contact the IBM Support Center.

adapter_config_status	Explanation and recovery action
diag_fail	See 5.4.4, “Diagnosing SP Switch adapter problems” on page 301.

In Table 8, we use the phrase *adapter configuration command.* This refers to the SP Switch adapter configuration method. The syntax to use when invoking these methods is as follows:

```
/usr/lpp/ssp/css/cfgtb3 -v -l css0 > <output_filename>
```

If the problem persists after running the adapter reconfiguration command, we recommend that you collect the switch-related data using the `css.snap` command and contact your IBM Support Center for further assistance.

5.4.4.1 Adapter diagnostics failure

If the SDR switch_responds class indicates the attribute *adapter_config_status* has the value *diag_fail,* it is an indication that adapter diagnostics have failed. The recovery actions to take for adapter diagnostics failure can, in most cases, be determined by examining the Service Request Number(SRN) posted in the error log. With adapter diagnostics failure, the suggested actions are as follows:

- Log in to the failing node as root and issue the `errpt -a` command to view detailed information on the failing entry. Now match the SRN number listed at the bottom of the error log entry to reference against Table 9 on page 303, where *xx* should be interpreted as any value:

Table 9. Switch Adapter Failure Types

SRN	Actions
1xx	Do a software installation verification. If the verification is successful, but the problem persists, contact IBM Support Center.
28xx	See 5.4.7, “Diagnosing clock synchronization problems” on page 314. Check the external clock and do cable verification. If these checks are unsuccessful, contact IBM Support Center.

SRN	Actions
Axx	See 5.4.7, “Diagnosing clock synchronization problems” on page 314. Check the external clock and do cable verification. If these checks are unsuccessful, contact IBM Support Center.
All other SRN values	Contact your local hardware support to arrange for the adapter to be replaced.

- The diagnostics can be run manually from the command line to double-check the output seen in the error log. The syntax of the diagnostics, which is similar to power-on self test, is as follows:

  ```
  diag -c -d css0
  ```

- When cable or adapter problems are suspected and the post diagnosis runs successfully, run the adapter and adapter cable wrap test as follows:

  ```
  diag -A -d css0
  ```

This test needs both the card and cable wrap plug to complete.

5.4.5 Diagnosing Eunfence command problems

The `Eunfence` command first distributes the topology file to the nodes before they can be unfenced. But, if the command fails to distribute the topology file, it puts an entry in the *dist_topology.log* file on the primary node in the /var/adm/SPlogs/css directory.

The `Eunfence` command fails to distribute the topology file if Kerberos authentication is not correct. Figure 159 on page 305 is a sample output of the dist_topology.log file. From this log, it is evident that the unfence operation failed because Kerberos had some problem. Refer to Chapter 3, “Security environment” on page 167 for more details on solving Kerberos-related problems.

```
unfence: Mon Oct 5 21:16:45 EDT 1998

sp5n01.msc.itso.ibm.com: sp5n01: krshd: Kerberos Authentication Failed:
User rcm
d.sp5n01@MSC.ITSO.IBM.COM is not authorized to login to account root.^M
sp5n01.msc.itso.ibm.com: sp5n01: /usr/lpp/ssp/rcmd/bin/rcp: 0041-004
Kerberos rcmd failed: rcmd protocol failure.
sp5n01.msc.itso.ibm.com: sp5n01: rshd: 0826-813 Permission is denied.
sp5n01.msc.itso.ibm.com: pexscr:  5025-509 sp5n01 rsh had exit code 1
sp5n01.msc.itso.ibm.com: sp5n09.msc.itso.ibm.com: krshd: Kerberos
Authentication
 Failed: User rcmd.sp5n01@MSC.ITSO.IBM.COM is not authorized to login
to accountroot.^M
sp5n01.msc.itso.ibm.com: sp5n09.msc.itso.ibm.com: spk4rsh: 0041-004
Kerberos rcmd failed: rcmd protocol failure.
sp5n01.msc.itso.ibm.com: sp5n09.msc.itso.ibm.com: rshd: 0826-813
Permission is denied.
sp5n01.msc.itso.ibm.com: dsh: 5025-509 sp5n09.msc.itso.ibm.com rsh had
exit code 1
```

Figure 159. Sample output of the dist_topology.log file

The `Eunfence` command will time out if the Worm daemon is not running on the node. So, before running the `Eunfence` command, make sure the Worm daemon is up and running on the node. To start the Worm daemon on the node, run the `rc.switch` script located in /usr/lpp/ssp/css.

If the problem persists after having correct Kerberos authentication, and the Worm daemon is running, the next step is to reboot the node. Then, try the `Eunfence` command again.

If neither of the previous steps resolve the problem, you can run diagnostics to isolate any hardware problem on the node. Refer to 5.4.4.1, “Adapter diagnostics failure” on page 303 for an explanation of how to run adapter diagnostics.

The last resort, if all else fails, would be to issue an `Eclock` command. This is completely disruptive to the entire switch environment; so, it should only be issued if no one is using the switch. An `Estart` must be run after `Eclock` completes. We discuss how to determine the clock status and how to run the `Eclock` command in 5.4.7, “Diagnosing clock synchronization problems” on page 314

5.4.6 Diagnosing Estart problems

Issuing the `Estart` command on the CWS can lead to three possible scenarios. The first possible case is the command is a complete success to bring the switch up. The second possible case is the command fails with messages or is timed-out without messages. The third possible event that can happen is the command is partly successful. In the following sections, we first describe what the actions are for a total failure of issuing `Estart`, and then we consider the case of partial success.

Note

Prior to PSSP3.1, the autojoin attribute in the switch_responds class is usually turned off for all nodes. The only exception is the node that is explicitly fenced with the `Efence -autojoin` command. The node has this attribute turned on.

With PSSP 3.1 or later, the autojoin attribute is now turned on for all nodes. The exceptions for this are the nodes fenced with the `Efence` command and the nodes that the fault service daemon considers as having a persistent problem.

The switch scan, conducted every two minutes by the primary node, has been modified in PSSP 3.1 to unfence any nodes that have the switch adapter microcode, and also the fault service daemon, running.

So, if a node is rebooted or powered on, it will automatically come up on the switch after the primary node found that it is ready.

5.4.6.1 Estart fails or times-out

If the `Estart` command fails or times-out, the list of actions to take is as follows:

1. Log in 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 10.

Table 10. Critical messages in the fs_daemon_print file of the Estart problem

Message	Analysis and actions
Error in buildDeviceDatabase()	Unable to build device database due to a missing or corrupt topology file or a malloc() system call failure. See 5.2.2, “Verification of the SP Switch topology file” on page 283.

Message	Analysis and actions
Error in TBSswitchinit()	Unable to initialize the switch network. See 5.4.6.2, “SP Switch worm error” on page 309.
Error in writeDeviceDatabase()	Unable to write /var/adm/SPlogs/css/out.top either because there is no space in the /var filesystem or due to a corrupt or missing topology. Check the /var filesystem size using the `df` command. See 5.2.2, “Verification of the SP Switch topology file” on page 283.
Can not access SDR - current backup not changed	An SDR failure has occurred. Run `SDR_test`.

Error in
fopen(act.top.PID)
fprint(act.top.PID)
fclose(act.top.PID)
rename(act.top, act.top.PID).
	An error occurred while accessing the /var/adm/SPlogs/css/act.top.PID file.
	Check the permission of the directory or check whether the file exists.

If nothing obvious is found in the fs_daemon_print.file, we recommend that you do the following before contacting the IBM Support Center:

1. Run `SDR_test` to double check that the SDR is working correctly.
2. Run `SDRGetObjects switch_responds` to read the SDR switch_responds class and look for the values of adapter_config_status attribute. See 5.4.4, “Diagnosing SP Switch adapter problems” on page 301 for details.
3. Run `Etopology -read <file_name>`. Compare the output of the topology file with the actual cabling and make sure all the entries are correct.
4. Make sure the Worm is up and running on all the nodes. Check the worm.trace file on the primary node for Worm initialization failure. See 5.4.6.2, “SP Switch worm error” on page 309 for details.
5. Make sure Kerberos authentication is correct for all the nodes.
6. Run `Eclock -d` and bring the Worm up on all nodes executing the `rc.switch` script.
7. Change the primary node to a different node using the `Eprimary` command. In changing the primary node, it is better to select a node attached to a different switch chip from the original primary or even a different switch board, if more than one is being used.

8. Check if some nodes are fenced. Use the `SDRChangeAttrValues` command as follows to unfence the primary and oncoming primary:

```
SDRChangeAttrValues switch_responds node_number==<primary node_num>
isolated=0
```

Note

This `SDRChangeAttrValues` command is a last option, and it should be used with caution.

Another point to be aware is the usage of = and ==. If = is used in place of ==, the value of that attribute will also be changed. The == is used to select the attribute of a SDR class. It is recommended that you make a copy of the object class before altering it.

- Now try `Estart`. If it fails, contact your IBM Support Center.

fs_daemon_print.file

This file on the primary node contains the status of the response from a node whenever a service packet or node command is sent from the primary node. Therefore, this file is useful to understand and to monitor the communication between the primary node and all other nodes.

For example, if at some instance of time we run the `Estart` command in our SP environment, the primary node sends broadcast messages to all other nodes, and all nodes that are “alive” respond back to the primary node. This sequence of operations is logged in the fs_daemon_print.file file.

In Figure 160, we see that primary node receives a response from switch nodes 12, 8, and 0 in reply to its broadcast message. Now, we deliberately kill the daemon on switch node 8. This change also gets reflected in the same file.

```
fs_daemon_bcast_cmd: Command to broadcast = 6
bcast_cmd_rcv_ACK: Received ACK from switch node number 12
bcast_cmd_rcv_ACK: Received ACK from switch node number 8
bcast_cmd_rcv_ACK: Received ACK from switch node number 0
bcast_cmd_ACKs_outstanding: no ACKs outstanding
bcast_cmd_ACKs_outstanding: no ACKs outstanding
fs_daemon_bcast_cmd: Success
fs_daemon_bcast_cmd: Returning 0
init_on_startup_msg: The Primary backup is switch node number = 12
```

Figure 160. fs_daemon_print.file on primary node

When we kill the Worm daemon on switch node 9, the primary node updates it database by removing switch node 8. Then, it broadcasts messages about this change and sets up communication with the responding nodes. In Figure 161, we see that the primary node runs an error recovery function and reports about the database change. Then, it sends the broadcast message. After that, the log confirms that it has received acknowledgments from switch node 12 and node 0.

```
primary_on_fault: SP_Switch_error_recovery() successful. Worm DB
changed.
fs_daemon_bcast_DBupdates: Entry
bcast_cmd_rcv_ACK: Received ACK from switch node number 12
bcast_cmd_rcv_ACK: Received ACK from switch node number 0
bcast_cmd_ACKs_outstanding: no ACKs outstanding
bcast_cmd_ACKs_outstanding: no ACKs outstanding
fs_daemon_bcast_DBupdates: Success
fs_daemon_bcast_DBupdates: Returning 0
fs_daemon_send_Routes: Entry
fs_daemon_send_Routes: Success
```

Figure 161. fs_daemon_print.file entries when a node dies

Monitoring this information is useful when some nodes get fenced automatically in the SDR. Especially on large SP systems, a distribution point is assigned on each switch board; then, during `Estart`, the primary node sends a packet to each distribution node. The distribution node, in turn, notifies all nodes on its switch board and waits for an acknowledgment from each node. Once the acknowledgments are all received, the distribution node, in turn, sends an acknowledgment back to the primary node.

In some failing scenarios, the primary node sends packets to all the distribution points, but one of the distribution nodes may fail to respond in time because either it is too busy, or one of its client nodes did not respond back in time. So, the nodes connected to that distribution nodes get fenced in the SDR.

5.4.6.2 SP Switch worm error

If the fault service daemon on the primary node detects some problem while initializing the switch, it creates an entry in the *worm.trace* file. The `Estart` command may fail if there is a Worm initialization error in the switch fabric. Therefore, it is worth checking the file while diagnosing `Estart` problems. The following steps are used to check the Worm error:

1. Log in to the primary node.

2. View the bottom of the /var/adm/SPlogs/css/worm.trace file. There should be messages similar to one of the following, where *xx* is any value:

TBSworm_bfs_phase1() failed with rc = x x

or

TBSworm_bfs_phase2() failed with rc = x x

Use the value of the xx retuned on either messages to index Table 11.

Table 11. SP Switch Worm return codes

Message codes	**Analysis and actions**
-3	The local adapter receiver port is not enabled. Check that oncoming primary is not fenced. The switch is not clocked. Issue `Eclock -d` command.
-4	Unable to generate routes for the network, probably due to a corrupt topology file. See 5.2.2 "verification of Switch Topology file"
-5	The send packet from the local node failed. Check the switch adapter by running switch adapter diagnostics on the primary node.
-6	A switch miswire 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.
-7	A node miswire detected because the switch network cabling does not match the topology file. There are two possible causes for this condition: The switch network is miswire, or the frame's supervisor TTY is not cabled properly. First, review /var/adm/SPlogs/css/cable_miswire file to determine if any cable are in question and check or reconnect if there are. Otherwise, issue the `Eclock -d` command and rerun `Estart`.
-8	The receive FIFO queue is full, probably a faulty switch adapter. Run switch adapter diagnostics to identify the problem.
-9	Unable to initialize FIFOs. Check the switch adapter by running switch adapter diagnostics to identify the problem.
-23	Unable to receive any response from the nodes. Issue the `Eclock -d` command and rerun `Estart`.

Message codes	Analysis and actions
-27	The TBIC was not initialized. This could simply be because the switch adapter did not initialize. Run `rc.switch` on the primary node followed by an `Estart`. If this has no effect, check the switch adapter by running the switch adapter diagnostics on the primary node.
-36	This node resigned as the primary node because it could no longer control and monitor the switch. This is an informational message.
-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.

Here, we mention some of the steps of switch initialization to understand the content of the *worm.trace* file. We know that `Estart` executes the `Estart_sw` script on the primary node. Once the script creates the *act.top.<pid>* file, the fault service daemon on the primary node executes the Worm daemon code. This code works in two phases called phase 1 and phase 2.

In phase 1, Worm uses the *Breadth First Search* (BFS) algorithm to discover the topology of the switch. In the worm.trace file, this phase appears as *TBSworm_bfs_phase1*. Worm uses the topology file to discover switch chips and node processors in phase 1.

Note

Breadth First Search is a graph search algorithm that tries all one-step extensions of current paths before trying larger extensions. This requires all current paths (or at least their end points) to be kept in memory simultaneously, at least their end points.

If the topology file is not correct, the Worm cannot complete phase 1. If in phase 1, Worm finds a defective link, it disables both ports on each side of the link. In subsequent switch reinitializations, the Worm will always find the link down, even if that link is then operational. Thus, to enable a link that was down and then fixed, we need to reset the switch chips running the `Eclock` command.

Phase 2 is marked as *TBSworm_bfs_phase2* in the worm.trace file. In this phase, the fault service daemon sends service packets called DataBase (DB) update packets to all the participating nodes in a hierarchical fashion.

The fault service daemon on the nodes uses the DB updates packet to create the out.top file.

Missing topology files on the nodes cause the daemon to fail to create the out.top file, and the nodes become fenced. Also in this phase, routing is generated by the route generation code (RTG) of the Worm daemon. The generated routes are loaded into the SRAM of the switch adapter. All these steps (as well any failure of these steps) get recorded in the worm.trace file. Analyzing the return code in the worm.trace file can help determine which steps failed to complete.

Figure 162 is a sample output of a worm.trace file. At any point in time, if Worm code is initiated by executing an `Estart` command, the trace logs the information shown in the figure. From this part of the log, the information we retrieve is as follows:

- Date and time of switch initialization.
- The switch node number of the primary backup node.
- The device ID visited first; in this case it is 100015 (which means switch chip 5 of switch board 1).
- Route information to and from the device visited.
- If Worm can send and receive information from the device, it puts the message `TBSworm_bfs_phase1: handleSwDeviceInitResponse() successful` in the log.

```
(i) print_the_time_worm: The date and time  = Wed Oct 21 20:35:09 1998
TBSswitchInit: Switch network Initialization Started!
(i) TBSswitchInit: The Primary backup is node  with switch node number 4
TBSworm_bfs_phase1: Switch Phase1 network Initialization Started!
syncFifoPh1: Cleaning up the Receive FIFO
(i) TBSworm_bfs_phase1: Device ID = 100015
route to device 100015   = 00000000 00000000
route from device 100015 = 38000000 00000000
TBSworm_bfs_phase1: Device ID = 100015 has been Visited.
----------------------------------------------------------
Current Device ID: 100015 type: 101 level: 1
----------------------------------------------------------
route to device 100015   = 00000000 00000000
route from device 100015 = 38000000 00000000
TBSworm_bfs_phase1: handleSwDeviceInitResponse() successful
```

Figure 162. worm.trace output—Part one

The reason Worm first visited switch chip 5 is because our primary node is switch node 0, which is connected to switch chip 5, port 0. After this, Worm uses the topology file to visit the next device connected to switch chip 5 and reports the status of communication. Figure 163 shows the next few lines of the trace file.

```
TBSworm_bfs_phase1: The current device port = 1
TBSworm_bfs_phase1: The attached device id = 4
route to device 4   = 89000000 00000001
route from device 4 = 83000000 00000001
ResponseWaitAndReceive: 2510-753 Time out on receive of original
packet.
----------------------------------------------------------
Phase-1 Node Initialization Packet for device 4
----------------------------------------------------------
route   = 89000000 00000001
Primary = 83000000 00000001
topology_fn = /etc/SP/expected.new.5
personality =  3  db_cmd =  1  error_enable = 0000
TBSworm_bfs_phase1: Device ID = 4 has been Visited.
TBSworm_bfs_phase1: The current device port = 2
TBSworm_bfs_phase1: The attached device id = 1
TBSworm_bfs_phase1: The current device port = 3
TBSworm_bfs_phase1: The attached device id = 0
TBSworm_bfs_phase1: The current device port = 4
```

Figure 163. worm.trace output—Part two

In Figure 163, Worm reports that the next device port is 1, and the attached device is switch node 4. It then tries to send and receive the route information from device 4, but some time-out occurs. So, Worm tries to contact device 4 again by sending a node initialization packet using the topology file named expected.new.5. It also reports the personality of device 4 is 3, which means the node is the primary backup node. Then, it continues with port 2. In phase 1, Worm visits all the switch chip and chip ports according to the topology file. If Worm encounters any problem, it puts entries in the trace file.

The rest of the log is the trace of sending packets to, and receiving responses from, switch ports and nodes using the topology file. If the logical flow of the portion of worm.trace file shown here is understood, then rest of the worm.trace file is easy to understand because it is the repetition of the same logic with different switch ports or nodes.

By reading the worm.trace file, we can check if Worm can reach all the nodes listed in the topology file. Any cable miswire can be detected using this trace file. Also, we can check the hardware status of switch ports. The worm.trace file is useful in understanding the topology file corruption problem.

Note

Successful Worm initialization is dependent on the correct expected topology file. If the switch is not initialized because of an incorrect or corrupted topology file, the content of the worm.trace file remains empty or contains a single line as follows:

```
Performing resetAdapter
```

5.4.6.3 Estart initializes some nodes

The first file to check is /var/adm/SPlogs/css/daemon.stderr. This file lists the nodes that did not initialize. The most recent entries are at the bottom of the file.

Therefore, you should log in to the primary node and check the /var/adm/SPlogs/css/out.top file. The return code in the out.top file tells you why nodes are not on the switch fabric and what action should be taken. See 5.4.2.3, “The out.top file” on page 299 for more information.

Also, cross-referencing the out.top file with the flt file on the primary node will help diagnose the problem.

5.4.7 Diagnosing clock synchronization problems

The SP Switch is a synchronous system. Communication between two switch chips or between a switch chip and a node adapter is synchronous.

In the simplest configuration of a single switch board, one of its internal oscillators is selected as the clock source. One of the switch chips on the switch board is selected as the master chip for the board. The master chip receives the signal from the selected oscillator. It then distributes the signal to all other switch chips, using the dedicated clock wires. All the data cables that connect a switch board to node adapters carry the clock signal.

There is a utility called `read_tbic`, which checks whether or not the external clock is operational on the local node. To check the clock, we issue the following command on the command line:

```
/usr/lpp/ssp/css/diags/read_tbic -s
```

The command `read_tbic -s` shows the status of the TBIC status register. If the external clock is present on the local node, the value of the TBIC status register must be 78000000. If the value is other than 7800000, it indicates that clock signal is absent (see Figure 164).

```
# /usr/lpp/ssp/css/diags/read_tbic -s
TBIC status register        : 78000000
```

Figure 164. TBIC status register value

To start the Worm daemon on the local node, we issue the following two commands:

`/usr/lpp/ssp/css/rc.switch` (on the node)

On the CWS, we run:

```
Eclock -d
```

Note

`Eclock -d` is a very disruptive command because it performs a power-on reset to all switch boards and resynchronizes all links in the whole system. In a production environment, the command must be used with extra caution. We recommend you use the command only when it is absolutely necessary. After issuing `Eclock`, we need to bring up Worm on all nodes by issuing the `rc.switch` command. Then `Estart` must be issued.

`Eclock -r` extracts the clock distribution data from the SDR and establishes the most recent clock distribution tree. Using `Eclock` with `-r` option is better than `-d` or `-f`, in an environment where the clock file is modified due to hardware problems.

In a multi-switch board environment, one of the switch boards is the master board. It uses its internal oscillator and then distributes the clock to the other slave boards. On each slave board, one of the switch chips acts as a master chip for receiving the clock from the master board. Then, it distributes the clock signal to the remaining seven switch chips. For this reason, a problem on a single switch cable or the power-off of a switch board may also affect the other switch boards.

Since clock signals are carried by cables. A faulty cable between the switch chip-to-node adapter or a faulty cable between switch board-to-switch board may cause you to lose the external clock signal. If the `Eclcok -d` command fails more than once, verify the cables and cable connectors. We recommend you call your local IBM hardware support center to do the cable verification. See *PSSP: Diagnosis Guide*, GA22-7350, for cable verification procedures.

5.5 Case studies

In this section, we show how to use the problem determination methodology to solve SP Switch related problems.

5.5.1 Single node failure

This is a typical example of a single node failure problem.

5.5.1.1 Problem symptom

In this scenario, we have an SP with four high nodes and an SP Switch. While the nodes were running, something happened, and we lost the switch responds of one of the nodes.

We issue `spmon -d` to examine the status of the switch responds. The output of the `spmon` is shown in Figure 165.

```
5.  Checking nodes
--------------------------------- Frame 1
Frame  Node    Node         Host/Switch  Key     Env   Front Panel
LCD/LED is
Slot   Number  Type  Power   Responds    Switch  Fail    LCD/LED
Flashing
-----------------------------------------------------------------------
  1       1    high    on    yes  yes     normal   no   LCDs are blank   no
  5       5    high    on    yes  yes     normal   no   LCDs are blank   no
  9       9    high    on    yes  autojn  normal   no   LCDs are blank   no
 13      13    high    on    yes  yes     normal   no   LCDs are blank   no
```

Figure 165. SP Switch problem symptom

The output of `spmon` shows that the switch respond is autojoin for node 9. We try to unfence the node as follows:

```
Eunfence 9
```

The command fails with the following messages:

```
Unable to unfence the following nodes

sp5n09.msc.itso.ibm.com
```

Since `Eunfece` is not successful, we run `Estart` to reinitialize the switch. Figure 166 shows that `Estart` is complaining about sp5n09, reporting that it is fenced. The problem node is the oncoming primary backup node, and since it is fenced, some other nodes will be selected as backup node. We need to find out why we cannot unfence node sp5n09.

```
Estart: Oncoming primary backup node sp5n09.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
Estart:  0028-061 Estart is being issued to the primary node:
sp5n05.msc.itso.ib
m.com.
/usr/lpp/ssp/css/Estart_sw[422]:
/u/dawn/trout/obj/power/ssp/css/msg/spmsg_css:
 not found.
/usr/lpp/ssp/css/Estart_sw[538]:
/u/dawn/trout/obj/power/ssp/css/msg/spmsg_css:
 not found.
/usr/lpp/ssp/css/Estart_sw[554]:
/u/dawn/trout/obj/power/ssp/css/msg/spmsg_css:
 not found.
```

Figure 166. Estart failure

5.5.1.2 Problem determination

We will use the flowchart, shown in Figure 149 on page 290, for a single node failure problem in order to solve this problem.

We start from the very beginning of the flowchart where the switch is up and running, and nothing has changed. We can skip the prerequisites verification and follow the yes branch of the verification test.

The next step in the flowchart is to check the AIX error log on the problem node. We notice an entry as shown in Figure 167 on page 318. The error log indicates a probable cable fault. We need to double check everything before replacing the cable or placing a hardware call.

```
LABEL:          TB3_TRANSIENT_RE
IDENTIFIER:     06C2F1C9

Date/Time:       Wed Oct 14 14:12:11
Sequence Number: 104
Machine Id:      00091036A400
Node Id:         sp5n09
Class:           H
Type:            TEMP
Resource Name:   css
Resource Class:  NONE
Resource Type:   NONE
Location:        NONE

Description
Switch adapter transient error

Probable Causes
Loose, disconnected or bad switch cable

User Causes
Switch cable loose or disconnected

        Recommended Actions
        Check / reconnect / replace cable if problem persists

Failure Causes
Switch cable faulty

        Recommended Actions
        Check / reconnect / replace cable if problem persists

Detail Data
Software ID String
LPP=PSSP,Fn=TB3recovery.c,SID=1.29,L#=566,
Interrupt source (ISR or MX CFG3)
TBIC intr
Bus Err (DMA CSR, MX MBA_ER, PCI C/S)
not applicable
Error Status Regs (INT_ERR and INT_ERR2)
00011000 00000000
```

Figure 167. AIX error log entry

We continue with the flowchart. After checking the error log, we need to check the Kerberos authentication. We then run the `dsh -a date` command over the Ethernet interface, both on the CWS and on the nodes, to make sure Kerberos is working.

The `spmon` output shows us that host_responds is yes for this node. We check the `SDRGetObjects switch_responds` command output to find out what is the value of the adapter_config_status, and it is css_ready; however, the isolation bit has value 1, indicating that the node is fenced. Figure 168 shows the output of the command. This gives an indication that adapter itself was configured when the problem occurred.

```
[sp5en0:/] SDRGetObjects switch_responds
node_number  switch_responds autojoin     isolated
adapter_config_status
          1           1           1           0 css_ready
          5           1           1           0 css_ready
          9           0           1           1 css_ready
         13           1           1           0 css_ready
```

Figure 168. SDRGetObjects switch_responds output

Now, the flowchart suggests we should check the status of the Worm daemon on the node. We run the command `ps -ef|grep Worm` on the node and find that the Worm daemon is up and running.

Now, the flowchart tells us to check the clock synchronization of the node, which refers us to 5.4.7, “Diagnosing clock synchronization problems” on page 314. To check the clock synchronization on the node, we run the following command:

```
/usr/lpp/ssp/css/diags/read_tbic -s
```

The output of the command is shown in Figure 169 on page 320, and the value of the register is not 780000. Now, we know that node has a clock problem. The next step is to run the `Eclock` command. But `Eclock` will disturb the operation of the switch for the whole system. The clock signal can also be missing if we have cable problem. So, we should do the cable verification first and then try the `Eclock` command.

```
# /usr/lpp/ssp/css/diags/read_tbic -s
TBIC status register        : 3c000000
```

Figure 169. Checking the clock synchronization on the node

Cable verification shows that the switch cable to the node is not properly connected to the switch adapter. So, we reset the cable. We kill the Worm daemon and issue the `rc.switch` command to reinitialize the Worm daemon. Now, we issue the command `read_tbic -s` again and find the desired value of 780000 for clock.

Now, we check the `spmon` output and find that switch responds change the attribute from autojoin to yes.

5.5.1.3 Conclusion

The node has a loose connection between the switch adapter and the switch cable. The AIX log already reports the problem as a switch transient error. We confirmed that it was a cable error by checking the clock signal on the node. The clock signal is carried by the cable, and we were missing the signal on the node because of the loose cable. We made a cable verification and then reset the cable. Finally, we restarted the Worm daemon on the node, and then the switch admin daemon automatically unfenced the node.

5.5.2 Multiple node failure

This is a typical example of a multiple node failure.

5.5.2.1 Problem symptom

Two new high nodes are added to the frame. These two nodes are installed from the mksysb of an existing node. The primary and primary backup nodes are selected, and then `Estart` is executed, but it fails as shown in Figure 170. The messages indicate that the oncoming primary node is fenced.

```
Estart: Oncoming primary != primary, Estart directed to oncoming
primary
Estart:  0028-035  Oncoming primary node is fenced, cannot Estart:
sp5n01.msc.itso.ibm.com
```

Figure 170. Estart failure message

An attempt is made to unfence the primary node, but it also failed, the `Eunfence` command timed-out. We check the `spmon` output and find that all nodes have host responds, and node 9 and node 13 also have the switch responds. We check the Kerberos using the `dsh` command from the CWS, and it works without any complaint. Then, we check the Worm daemon on all nodes, and it is up and running.

5.5.2.2 Problem determination

The problem looks like an `Estart` failure problem. We also do not have switch responds for two nodes on the SP Perspectives graphical interface; so, this is a multiple node failure problem. We follow the flowchart for the multiple node failure problem. The flow chart refers to 5.4.6, "Diagnosing Estart problems" on page 306 for solving `Estart` problems.

Before digging into the `Estart` problem, we want to check the summlog file to get an overall picture of the switch fabric. Figure 171 shows part of the entries in the summlog file when the problem is reported. The file reports an `SP_SW_SIGTERM_ER` and `HPS_FAULT6_ER error` on the primary node sp5n09.

```
101419071998 sp5n13 N sp5en0 187 SP_SW_SIGTERM_ER
101419071998 sp5n13 N sp5en0 188 HPS_FAULT6_ER
101419071998 sp5n01 N sp5en0 245 SP_SW_SIGTERM_ER
101419071998 sp5n09 N sp5en0 126 SP_SW_SIGTERM_ER
101419071998 sp5n01 N sp5en0 246 HPS_FAULT6_ER
101419071998 sp5n09 N sp5en0 127 HPS_FAULT6_ER
101419281998 sp5en0 N global 1381 SP_SW_ECLOCK_RE
101419281998 sp5n01 N sp5en0 247 SP_SW_SIGTERM_ER
101419281998 sp5n01 N sp5en0 248 HPS_FAULT6_ER
101419281998 sp5n09 N sp5en0 128 SP_SW_SIGTERM_ER
101419281998 sp5n09 N sp5en0 129 HPS_FAULT6_ER
101419281998 sp5n13 N sp5en0 189 SP_SW_SIGTERM_ER
101419281998 sp5n13 N sp5en0 190 HPS_FAULT6_ER
101419301998 sp5n09 N sp5en0 130 HPS_FAULT6_ER
101419371998 sp5en0 N global 1392 SP_SW_ECLOCK_RE
101419371998 sp5n13 N sp5en0 191 SP_SW_SIGTERM_ER
101419371998 sp5n13 N sp5en0 192 HPS_FAULT6_ER
101419371998 sp5n01 N sp5en0 249 SP_SW_SIGTERM_ER
101419371998 sp5n01 N sp5en0 250 HPS_FAULT6_ER
101419371998 sp5n09 N sp5en0 131 SP_SW_SIGTERM_ER
101419371998 sp5n09 N sp5en0 132 SP_SW_SIGTERM_ER
```

Figure 171. The summlog entries at the time the problem is reported

For the explanation of `P_SW_SIGTERM_ER` and `HPS_FAULT6_ER`, we check the *PSSP: Diagnosis Guide*, GA22-7350. Table 12 on page 322 is an excerpt from this guide and shows the explanation of the error label of the summlog file. Analyzing the summlog file, we can say that, for some reason, the Worm daemon died on the primary node.

Table 12. Explanation of error label

Error label	Explanation
SP_SW_SIGTERM_ER	Switch daemon received SIGTERM. Another process sent a SIGTERM to Worm. To recover, run the `rc.switch` script.
HPS_FAULT6_ER	The switch daemon process terminated. A bad switch adapter or missing external clock is the cause of the message.

We run the `Eprimary` command to verify which are the current primary and primary backup nodes. Figure 172 shows that node 9 is the current primary node, and node 13 is the primary backup node.

```
[sp5en0:/spdata/sys1/ha/css] Eprimary
9       - primary
1       - oncoming primary
13      - primary backup
5       - oncoming primary backup
```

Figure 172. Checking the primary and primary backup nodes

We run the `Efence` command to find the status of node 5 and node 1. These nodes are fenced as shown in Figure 173.

```
[sp5en0:/spdata/sys1/ha/css] Efence
sp5n01.msc.itso.ibm.com
sp5n05.msc.itso.ibm.com
```

Figure 173. Checking which nodes are fenced

5.4.6, “Diagnosing Estart problems” on page 306 tells us to check the fs_daemon_print.file on the primary node for probable causes of the `Estart` failure. The messages related to our problem found in the fs_daemon_print.file, as shown in Figure 174, which only indicates that, for some reason, node sp5n05 is fenced.

```
e) 10/14/98 20:16:50 : 2510-730 Node sp5n01.msc.itso.ibm.com NOT
UnFenced, rc =-32.
(e) 10/14/98 20:17:03 : 2510-730 Node sp5n05.msc.itso.ibm.com NOT
UnFenced, rc =-32.
_____________________________________________________________________
(e) 10/14/98 20:17:19 : 2510-730 Node sp5n01.msc.itso.ibm.com NOT
UnFenced, rc =-32.
_____________________________________________________________________
(e) 10/14/98 20:18:50 : 2510-730 Node sp5n01.msc.itso.ibm.com NOT
UnFenced, rc =-32.
(e) 10/14/98 20:19:03 : 2510-730 Node sp5n05.msc.itso.ibm.com NOT
UnFenced, rc = -32.
```

Figure 174. fs_daemon_print.file on the primary node

There are three more logs that should be checked for problem determination. One of the logs is the out.top file. We check the out.top file and notice entries for node 1 and node 5 as shown in Figure 175.

```
s 15 1  tb3 0 0            E01-S17-BH-J9 to E01-N1   -5 R: device has
been removed from network by the system administrator (link has been
removed from network- fenced)
s 15 3  tb3 4 0            E01-S17-BH-J7 to E01-N5   -5 R: device has
been removed from network by the system administrator (link has been
removed from network- fenced)
```

Figure 175. The out.top file

Return code -5 indicates the following:

```
the device is placed off-line by the System Administrator with the Efence
command. Run Eunfence command against the device.
```

We already tried the `Eunfence` command, but it did not change the scenario. One thing we notice from the out.top file is that both node 1 and node 5 are on the same switch chip (switch chip 5). There might be a problem with this chip, which caused those nodes to be fenced. To check switch faults, we need to check the AIX error log on the primary node.

We now check the AIX error log on the primary node (node 9) and find a related entry as shown in Figure 176 on page 324. The error log states a problem with the SDR daemon.

```
LABEL:          SP_SW_SDR_FAIL_RE
IDENTIFIER:     EC771C6B

Date/Time:       Wed Oct 14 20:13:49
Sequence Number: 140
Machine Id:      00091036A400
Node Id:         sp5n09
Class:           S
Type:            TEMP
Resource Name:   Worm

Description
Switch daemon SDR communications failed

Probable Causes
Ethernet overloaded
Excessive SDR traffic
SDR daemon or control workstation down
Software Error
Call software service if problem persists
```

Figure 176. AIX error log entry on the primary node

To verify the SDR, we run `SDR_test` on the CWS but did not find any problem. Therefore, we can say that there is no problem with the SDR daemon.

The next thing to check in the SDR is the topology file. Now, we check if we have the right topology file and make sure the file is not corrupted. To read the topology file, we run the following command:

```
Etopology -read /tmp/test
```

The command saves the topology file stored in the SDR to /tmp/test.

We now compare the content of the file (which is named test with the actual cabling). We find that the topology file stored in the SDR is different from the actual cabling on the switch board. The cabling was done according to the IBM-supplied topology file. So, we compare the topology stored in the SDR with the IBM-supplied expected.top.1nsb.0isb.0 file, and we notice that the file stored in the SDR has wrong port information for node 5 and node 1. Now, we know that the topology file in the SDR is incorrect.

The easy workaround for solving the corrupted topology file is to annotate the topology file again using expected.top.1nsb.0isb.0 file and save it in the SDR by using the following command:

```
Eannotator -F expected.top.1nsb.0isb.0 -f expected.top.new
```

Since we have annotated the topology file, we are required to synchronize the clock on all nodes. We initialize the clock using the following command:

```
Eclock -d
```

Before issuing `Estart`, we make sure that the Worm daemon is up and running on all the nodes. We run the `Estart` command to start the switch, but, again, the command prompt comes back with the message that the oncoming primary node is fenced.

To check if the oncoming primary node is fenced, we run the `Efence` command, and the message we receive is as follows:

```
Efence: 0028-036 No functional switch primary node, run Estart
```

We check the isolation bit in the SDR by using the command `SDRGetObjects switch_responds`. From Figure 177, we see that node 5 and node 1 are fenced in the SDR (the isolated bit is 1).

```
[sp5en0:/] SDRGetObjects switch_responds
node_number switch_responds autojoin isolated adapter_config_status
   1              0             0        1          css_ready
   5              0             0        1          css_ready
   9              0             1        0          css_ready
   13             0             1        0          css_ready
```

Figure 177. Isolation status of node 5 and node 1

One way to solve this problem is to change the primary node (oncoming primary node) and restart the switch. However, here we will change the isolated bit in the SDR for the current oncoming primary node as a way of demonstrating how this can be solved in case *all* your nodes are fenced; so, there is no “unfenced” node to assign the duties of being the primary. To change the isolated bit in the SDR, we use the following command:

```
SDRChangeAttrValues switch_responds node_number==1 isolated=0
```

Figure 178 shows the effect of the command.

```
[sp5en0:/] SDRGetObjects switch_responds
node_number switch_responds autojoin isolated adapter_config_status
     1             0            0        0          css_ready
     5             0            0        1          css_ready
     9             0            1        0          css_ready
     13            0            1        0          css_ready
```

Figure 178. Changing the node to non-isolated

Now, we run `Estart` again. This time `Estart` is successful in bringing all nodes up on the switch network.

5.5.2.3 Conclusion

We had a corrupted topology file in the SDR; therefore, the first time `Estart` was successful only in bringing up two nodes out of four nodes. However, through the analysis of different logs, we determined the topology corruption.

We annotated the topology file and saved it in SDR. Annotating alone, however, could not fix the problem because the clock was not synchronized on the nodes, and the oncoming primary node was fenced in the SDR. We were required to run the `Eclock` command to synchronize the clock and then manually unfenced the node in SDR.

We also checked the worm.trace file during our problem determination because the summlog file indicated problems with Worm. We revisited the worm.trace file and found some indication of a miswire fault during the time the problem was reported. Figure 179 on page 327 shows that Worm detected a miswire problem, which gave us an indication that the wrong topology file was being used.

```
Phase-1 Node Initialization Packet for device 0
----------------------------------------------------------
route   = 45890000 00000002
Primary = 4c830000 00000002
topology_fn = /etc/SP/expected.top.last.3
personality =  3  db_cmd =  1  error_enable = 0000

ResponseWaitAndReceive: 2510-924 Packet(s) found ,before packet match
was tried. Packetsmay or may not be discarded.

WormdisplayPacket Packet type = NODE_SVC_CMD_LEN_256:
        Command:  0xf9  Flag: 0x02
WormdisplayPacket Node Cmd = NODE_ERROR_STATUS:

WormdisplayPacket Packet type = NODE_SVC_CMD_LEN_256:
        Command:  0xf9  Flag: 0x02
WormdisplayPacket Node Cmd = NODE_ERROR_STATUS:
ResponseWaitAndReceive: 2510-923 First matched packet being returned.
handleNodeSvcInitVisitResponse: 2510-745 Node Miswire detected for
Device id - Expected= 0 Actual = 4.
handleNodeSvcInitVisitResponse: 2510-796 Node Miswire detected for
switch port - Expected = 1 Actual = 3.
ResponseWaitAndReceive: 2510-924 Packet(s) found,before packet match
was tried. Packetsmay or may not be discarded.
```

Figure 179. The content of the worm.trace file

5.5.2.4 ARP definition

This case study is unique in nature because it does not fall under our general classification of switch failures. This problem usually occurs when new nodes are installed, and wrong ARP information is put in the SDR.

5.5.2.5 Problem symptom

Two new nodes were installed in the existing frame, the switch was configured, and we had host responds and switch responds. The problem is the new two nodes can ping each other over the switch interface, but the other nodes cannot ping them.

5.5.2.6 Problem determination

From TCP/IP, we understand that this may occur if we do not have the correct IP addresses of the switch interface on the /etc/hosts file of the older nodes, or if we have wrong hostname resolution. Therefore, we check the IP addresses of the css0 interface on all the nodes as shown in Figure 180.

```
sp5en0{/} dsh -a grep sp5sw05 /etc/hosts
sp5n01: 192.168.15.5 sp5sw05.msc.itso.ibm.com sp5sw05
sp5n05: 192.168.15.5 sp5sw05.msc.itso.ibm.com  sp5sw05
sp5n09: 192.168.15.5 sp5sw05.msc.itso.ibm.com sp5sw05
sp5n13: 192.168.15.5 sp5sw05.msc.itso.ibm.com  sp5sw05

sp5en0{/} dsh -a grep sp5sw13 /etc/hosts
sp5n01: 192.168.15.13 sp5sw13.msc.itso.ibm.com sp5sw13 s13
sp5n05: 192.168.15.13 sp5sw13.msc.itso.ibm.com p5sw13
sp5n09: 192.168.15.13 sp5sw13.msc.itso.ibm.com sp5sw13 s13
sp5n13: 192.168.15.13 sp5sw13.msc.itso.ibm.com  sp5sw13
```

Figure 180. Checking the IP address for css0 interface on all nodes

Figure 180 shows the IP addresses and host names for the css0 interface on node 5 and node 13, the two new nodes. From this figure, we know that the information in the /etc/hosts file is consistent across the nodes.

Now let us examine the characteristics of the css0 interface on the new nodes as shown Figure 181:

```
sp5en0{/} dsh -w sp5n05,sp5n13 ifconfig css0

sp5n05: css0: flags=8008c3<UP,BROADCAST,RUNNING,NOARP,SIMPLEX>
sp5n05: inet 192.168.15.5 netmask 0xffffff00 broadcast 192.168.15.255
sp5n13: css0: flags=8008c3<UP,BROADCAST,RUNNING,NOARP,SIMPLEX>
sp5n13: inet 192.168.15.13 netmask 0xffffff00 broadcast 192.168.15.255
```

Figure 181. Comparing the css0 interface attribute on the new nodes

Figure 181 shows the ifconfig output on node 5 and node 13. In both cases, the attributes are the same except for their IP addresses. We expected that these attributes would be the same; so, we now examine the attributes of the css0 interface of older nodes as shown in Figure 182 on page 328.

```
sp5en0{/} dsh -w sp5n01,sp5n09 ifconfig css0
sp5n01: css0: flags=800843<UP,BROADCAST,RUNNING,SIMPLEX>
sp5n01: inet 192.168.15.1 netmask 0xffffff00 broadcast 192.168.15.255
sp5n09: css0: flags=800843<UP,BROADCAST,RUNNING,SIMPLEX>
sp5n09: inet 192.168.15.9 netmask 0xffffff00 broadcast 192.168.15.255
```

Figure 182. css0 interface attributes for node 1 and node 9

Now, we compare the information of Figure 181 on page 328 and that of Figure 182. First, we compare the netmask value for the interface of all nodes. We find that the netmask value is same for each node css0 interface. Other than the IP addresses of node 5 and node 9 css0, the interface has attribute NOARP, which means that ARP is not enabled for these nodes. We try to change this attribute on node 5 as follows:

```
/usr/lpp/ssp/css/ifconfig css0 arp
```

The command enables the ARP for the node 5 css0 interface. Now, we try to ping the css0 interface from node 1, and the test is successful. So, for some reason, the ARP was not enabled on node 5. We examine why the ARP was not enabled when we installed the node. We check the rc.switch.log file in the /var/adm/SPlogs/css directory because it holds the information when the switch adapter gets configured. The excerpt of the rc.switch.log file is shown in Figure 183 on page 329.

```
Thu Oct 22 13:35:18 EDT 1998
hostname is sp5n05
node_number is 5
adapter_config_status = css_ready
Enodes does not exist - using ODM and SDR.
switch_node_number is 4
switch_chip is 5
switch_board is 1
switch_chip_port is 1
IP_switch_netaddr is 192.168.15.5
IP_switch_netmask is 255.255.255.0
IP_switch_ARP_enabled is no
adapter is TB3
IP_switch_offset is 1
/usr/lpp/ssp/css/ifconfig css0 inet 192.168.15.5 netmask 255.255.255.0
down offset 1 tb2 -arp
```

Figure 183. rc.switch.log file on node 5

The log for the rc.switch shows explicitly that the adapter gets configured without ARP enabled. This initial information comes from the SDR. Therefore, we examine the Switch_partition object class in the SDR because ARP information is kept in this object class. We run the following command to check the ARP information as shown in Figure 184. We find that ARP is not enabled.

```
sp5en0{/usr/lpp/ssp/bin} SDRGetObjects Switch_partition arp_enabled
arp_enabled
no
```

Figure 184. Checking the ARP information in the SDR

To solve the problem, we need to enable the ARP in SDR, put node 5 and node 13 in customizing mode, and run setup_ server. Then, we run the pssp_script on the nodes to do the node customization. After customization, we need to start rc.switch on the nodes to bring up the switch adapter.

Appendix A. Wrappers

This appendix provides reference information regarding `setup_server` wrappers.

A.1 services_config

This script is invoked by /etc/rc.sp to set up designated services on the nodes, and by `setup_server` to set up services on the boot/install server.

It reads the SDR information stored in the SP object class and checks which services will be run on the node. Then, it calls the appropriate service configuration scripts. The possible services to invoke are the following:

- NTP
- Print management
- User management
- AMD
- File collection
- Accounting

Command Syntax:

```
services_config
```

This command has no input parameters.

The general flow and logic for `services_config` is:

1. Check the site_environment configuration. If one of the following options is configured, and ssp.sysman is not installed, get it from the CWS and install it:

   ntp_config != none

   amd_config = true

   print_config != false

   usermgmt_config = true

   filecoll_config = true

2. If the switch is installed, and ssp.css is not installed, get it from the CWS and install it.

3. If ssp.st is installed on the CWS, and is not installed on the node, get it from the CWS and install it.
4. Invoke config scripts for required subsystems:

   Invoke `ntp_config`

   Invoke `print_config`

   Invoke `admin_config`

   Invoke `filec_config`

   Invoke `amd_config`

   If accounting is enabled (spacct_enable=true), invoke acct_config.

A.2 setup_CWS

This script reads the information from the SDR, checks prerequisites, and updates Kerberos files to reflect CWS and node network interface names. It then makes a setup of CWS-specific items.

Command Syntax:

```
setup_CWS [-h]
```

This command has no input parameters.

The general flow and logic for `setup_CWS` is:

1. Parse command line parameters.
2. Retrieve information from the SDR.
3. Investigate prerequisites and conditions:
   - Verify that this is the CWS.
   - Verify that the required Kerberos files exist on the system:

     /etc/krb-srvtab

     /etc/krb.conf

     /etc/krb.realms

5. If there are new interfaces on the CWS or on the nodes that are not defined in the Kerberos database, perform the following actions:
   1. Update the Kerberos database (whether local or remote, AFS or k4) by adding new principals (new interfaces are defined).
   2. Update the /.klogin file.

3. Update the /etc/krb.realms.
4. If you are using AFS or remote Kerberos servers, create <reliable_hostname>-new-srvtab files on /tftpboot for the nodes. If the CWS is a Kerberos server, do not create these files since the `create_krb_files` wrapper does that function.
5. Update the local srvtab (/etc/krb-srvtab) file if a CWS host principal was added.

A.3 delnimmast

This wrapper deletes the NIM master definition, that is, nodes that were configured as NIM master are unconfigured, and the NIM filesets (master and SPOT filesets) are deleted from them.

Command Syntax:

```
delnimmast -l <node_number_list>
```

where: `<node_number_list>` is the list of nodes to undefine as NIM master. Node number 0 means the CWS.

The general flow and logic for `delnimmast` is:

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

For each node:

3. Set global variables.
4. Check prerequisites and conditions:

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

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

   If a node cannot be contacted via dsh, skip that node.
5. If a node is configured as a NIM master, unconfigure it.
   - Remove the SPOTs from the boot/install server. For each SPOT on this node, execute: `nim -o remove <spot_name>`
   - Deactivate the node as a NIM master. Execute:

     ```
     nim -o unconfigure master
     ```

   If it is not the CWS, remove everything under the /spdata/sys1/install directory.

6. If there are any of the NIM "master" filesets (master, SPOT), uninstall them.
7. If it is not the CWS, reinitialize NIM. Execute:

```
niminit -a name=<node_name> -a master=<bis_of_this_node>.
```

A.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 then 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> - The list of node numbers to remove as NIM clients. Node number 0 (the CWS) is not allowed.
-s	<server_node_list - > The list of server (NIM master) nodes on which to delete all clients that are no longer defined as boot/install clients in the SDR.

The general flow and logic for `delnimclient` is:

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

If the `-l` flag was specified, then for each specified client node in the <node_number_list>:

1. Determine the client NIM master.
2. Check prerequisites and conditions:
    1. If CWS, then skip to the next node.
    2. Specified node must be a valid SP node.
    3. Specified node boot server must be a valid SP node.
    4. Specified node boot server can be contacted via dsh.
    5. If the client boot/install server is not configured as a NIM master, then issue a message and skip the node.
    6. Specified node is configured as a client on its boot/install server.

3. Undefine the client on the master of that node. If an error occurs, issue a warning message.
    1. Reset the NIM client. Issue: `nim -o reset <node_name>`
    2. Deallocate all the NIM resources for the client node. Issue:

       ```
       nim -Fo deallocate -a subclass=all <node_name>
       ```
    3. Remove the client. Issue:

       ```
       nim -o remove <node_name>
       ```
    4. Remove the custom bosinst_data resource from this ex-client. The `mknimres` wrapper creates <node_number>_prompt, <node_number>_noprompt, and <node_number>_migrate resources for boot/install clients when hdisk0 is not the default installation disk. So, if we are no longer a server to the node, the resources are removed. Issue:

       ```
       nim -o remove <custom_bosinst_data>
       ```

       for each custom bosinst_data of this ex-client, and remove these files from the /spdata/sys1/install/pssp directory.

If the `-s` flag was specified, then for each specified server node:

1. Check prerequisites and conditions.
2. Delete clients from the NIM database that are no longer boot/install clients of this boot/install server (`setup_server` only calls these wrappers with the `-s` flag).
    1. Reset the NIM client. Issue:

       ```
       nim -o reset <node_name>
       ```
    2. Deallocate all the NIM resources for the client node. Issue:

       ```
       nim -Fo deallocate -a subclass=all <node_name>
       ```
    3. Remove the client. Issue:

       ```
       nim -o remove <node_name>
       ```
    4. Remove the custom bosinst_data resource from the ex-client. The mknimres wrapper creates <node_number>_prompt, <node_number>_noprompt, and <node_number>_migrate resources for the boot/install clients when hdisk0 is not the default installation disk. So, if we are no longer a server to the node, these resources are removed. Issue:

       ```
       nim -o remove <custom_bosinst_data>
       ```

for each custom bosinst_data of the ex-client, and remove the associated file from the /spdata/sys1/install/pssp directory.

A.5 mknimmast

This wrapper creates a NIM master. To do this, the NIM master filesets (bos.sysmgt.nim.master and bos.sysmgt.nim.spot) must be installed. The node is configured as a NIM master by using the `nimconfig` command.

Command Syntax:

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

`-h`	Display command syntax.
`-l`	<node_number_list> - List of node numbers to define as NIM masters. Node number 0 indicates the CWS.

General flow and logic for `mknimmast` is:

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

For each node:

1. Set global variables.
2. Check prerequisites and conditions. Skip the node if any of the following are true:
   - Node is not the CWS or boot/install server.
   - Node cannot be contacted via dsh.
   - Node is already configured as a NIM master.
   - The lppsource directory does not exist.
3. If the NIM master file sets are not installed, then:
   1. If the NIM master you are defining is not the CWS, export and mount the correct lppsource directory from the CWS.
   2. Install the NIM filesets (master and SPOT) from lppsource.
   3. If the NIM master you are defining is not the CWS, unmount the lppsource directory.
4. Activate the NIM master.

   Configure the NIM master on the node. Issue:

```
nimconfig -a netname=spnet_<if_name> -a pif_name=<if_name> -a
master_port=1058 -a cable_type=<cable_type>.
```

A.6 create_krb_files

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

Whether the operation is done on the server node or the CWS, 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

The general flow and logic for `create_krb_files` is:

1. Parse command line parameters.
2. Retrieve information from the SDR.
3. Check prerequisites.

   Verify that this is a boot/install server.
4. Create/update the /etc/tftpaccess.ctl file on the boot/install server.

For each boot/install client (that has bootp_response: install, migrate or customize):

1. Create the srvtab file on the CWS.
2. Copy the srvtab file from the CWS to the boot/install server (unless the CWS is the server).
3. If the /tftpboot/<hostname>-new-srvtab file exists, make it read-only for nobody (TFTP) by issuing the following commands:

   ```
   chmod 400 /tftpboot/<hostname>-new-srvtab
   chown nobody /tftpboot/<hostname>-new-srvtab
   ```

A.7 mknimint

This wrapper defines new Ethernet network and interface objects on the NIM master (boot/install server). On the CWS, any networks not previously defined are defined, and NIM interfaces are added. On a boot/install server that is not the CWS, all Ethernet network adapters and interfaces of the node are defined in the NIM master. Also, all CWS Ethernet and Token-Ring adapters and interfaces are defined. The CWS networks and interfaces are added to the boot/install server because there are resources, such as lppsource (which have the CWS as resource server), that the nodes have to access.

To serve a resource from the CWS to a client that is not on the same subnet as the CWS, routing is required. Routing is done by the `mknimclient` wrapper.

If `mknimint` is executed on a boot/install server that is not the CWS, it creates the CWS as a NIM client for the boot/install server, thus, making this server able to access the resources on the CWS.

Command Syntax:

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

`-h`	Display command syntax
`-l <node_number_list>`	List of node numbers to define as NIM masters. Node 0 is the CWS.

The general flow and logic for `mknimint` is:

1. Retrieve information from SDR.
2. Parse command line parameters.
3. Check prerequisites.

   Make sure this is a boot/install server.

For each node:

1. For each Ethernet network adapter not yet defined in this NIM master:
   1. Define the network to the NIM master. Execute:

      ```
      nim -o define -t ent -a net_addr=<ip-addr> -a snm=<netmask>
      spnet_<ent_if>.
      ```

   2. Define the interface to the NIM master. Execute:

      ```
      nim -o change -a if<seq>="spnet_<ent_if> <long_hostname>
      <hardware_addr>" -a cable_type<seq>=<cable_type> master.
      ```

2. If the node on which the operation is performed is not the CWS:

    1. Find all the Token-Ring and Ethernet interfaces on the CWS via the `netstat` command.

    2. Define all Token-Ring and Ethernet networks from the CWS on the boot/install server. Define only the CWS networks that are not on the same subnet as the Ethernet interface on the boot/install server. Execute:

    ```
    nim -o define -t <type> -a net_addr=<ip-addr> -a snm=<netmask>
    cw_<interface>
    ```

    where <type>=ent or tok.

    3. If the CWS is not yet defined as NIM client of this boot/install server, take any of the defined CWS networks and make the CWS a client of this boot/install server so that it can serve the lppsource resources to the clients of this server. Execute:

    ```
    nim -o define -t standalone -a platform=rs6k -a if1="<network>
    <long_hostname> 0" -a cable_type1=<cable_type> <CW_name>
    ```

    4. Define interfaces for all the networks on the CWS that were not used in the definition of the CWS client on this boot/install server. Execute:

    ```
    nim -o change -a if<seq>="<network> <long_hostname> 0 tok" -a
    ring_speed<seq>=<cable_type> <CWS_name>
    ```

    for Token-Ring interfaces and

    ```
    nim -o change -a if<seq>="<network> <long_hostname> 0 ent" -a
    cable_type<seq>=<cable_type> <CWS_name>
    ```

    for Ethernet interfaces.

A.8 mknimres

This wrapper creates NIM resources that are needed for operations on the nodes. Depending on the value of the bootp_response field from SDR, the resources that are needed in the processes with the given bootp_response, and do not yet exist, will be created. Table 13 shows this relationship.

Table 13. Resources table

bootp_response	**Resource needed**
install	bosinst_data, lpp_source, mksysb, spot, pssplpp, psspscript
migrate	bosinst_data, lpp_source, spot, pssplpp, psspscript

bootp_response	Resource needed
customize	pssplpp, psspscript
maintenance	bosinst_data, lpp_source, spot
diag	bosinst_data, spot
disk	None

`mknimres` checks the Rstate attribute of resource objects, and if it is not “ready for use,” they will be removed (exception on boot resource).

Command Syntax:

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

`-h` Display command syntax.

`-l` `<node_number_list>` List of node numbers to be defined as NIM masters. Node 0 is the CWS.

The general flow and logic for `mknimres` is:

1. Parse command line parameters.
2. Get information from the SDR.
3. Check prerequisites:
   - NIM master must be the CWS or server node (see SDR).
   - Make sure you have dsh access to the node.
4. Delete all resources with Rstate not "ready for use." If the resource name is not boot, execute:

   ```
   nim -o remove <resource_name>
   ```

   If it is boot, execute:

   ```
   m_chattr -a Rstate=available boot
   ```

5. Make a bosint_data object for the following NIM resources if they do not exist: prompt, noprompt, and migrate. Execute:

   ```
   nim -o define -a server=master -t bosinst_data -a
   location=/spdata/sys1/install/pssp/<bosinst_data>
   <bosinst_resource_name>
   ```

6. Make a custom bosinst_data object. If this NIM master has a boot client node that has a target disk other than hdisk0, create the <node_number>.migrate and <node_number>.noprompt files and the associated NIM resource.

7. Make the psspscript NIM resource if it does not already exist. Execute:

   ```
   nim -o define -a server=master -t script -a
   location=/spdata/sys1/install/pssp/pssp_script pssp_script
   ```

8. Make lpp_source NIM resources, if not already defined, and if some boot/install client needs it.

   - If the NIM master is the CWS, make lpp_source NIM resources. Execute:

     ```
     nim -o define -a server=master -t lppsource -a
     location=/spdata/sys1/install/<aixlvl>/lppsource lppsource_<aixlvl>
     ```

     and NFS export the directory.

   - If the NIM master is not the CWS, make lpp_source resource. This resource is served by the CWS. Execute:

     ```
     nim -o define -a server=<CW_name> -t lppsource -a
     location=/spdata/sys1/install/<aixlvl>/lppsource lppsource_<aixlvl>
     ```

9. Make mksysb NIM resources if some boot/install client needs it.

   - If the NIM master is the CWS, make mksysb NIM resources if they do not already exist. Execute:

     ```
     nim -o define -a server=master -t mksysb -a
     location=/spdata/sys1/install/images/<image_name> mksysb_<seq>
     ```

   - If the NIM master is not the CWS:

     1. If the mksysb NIM resource already exists and is of a different size than the one on the CWS, delete the resource and delete the file. Execute: `nim -o remove mksysb_<seq>`. Otherwise, do not create it.
     2. Create/increase the install_images logical volume and file system to hold the installation images that the boot/install clients need (/usr/lpp/ssp/install/makelv is executed).
     3. Copy the installation images from the boot/install server of this NIM master node and put them on the file system just created.
     4. Make the mksysb NIM resource. Execute:

        ```
        nim -o define -a server=master -t mksysb -a
        location=/spdata/sys1/install/images/<image_name> mksysb_<seq>
        ```

10. Make SPOT NIM resources, if they do not already exist, and there is a boot/install client that needs them.

    - Create/increase the spot_<aixlvl> logical volumes and file systems to hold the SPOTs that the boot/install clients need (/usr/lpp/ssp/install/makelv is executed).

- Make the SPOT NIM resource and redirect the output to the /tmp/spot.out.<pid> log file. Execute:

  ```
  nim -o define -a server=master -t spot -a
  location=/spdata/sys1/install/<aixlvl>/spot -a verbose=5 -a
  source=/spdata/sys1/install/<aixlvl>/lppsource spot_<aixlvl>
  ```

- Update the SPOT from the lppsource. Execute:

  ```
  nim -o cust -a lpp_source=lppsource_<aixlvl> -a fixes=update_all
  spot_<aixlvl>
  ```

11. Copy pssp.installp to the node server (boot/install server that is not the CWS).
    - Create or increase the install_pssplpp logical volume and file system to hold the PSSP level codes that the boot/install clients need (/usr/lpp/ssp/install/makelv is executed).
    - Copy the PSSP levels from the boot/install server of this NIM master node and put them on the file system just created.

A.9 mknimclient

This wrapper creates NIM client definitions on the boot/install server. It searches for the processor type of a client (UP/MP) and the platform type (rs6k/chrp). The information about processor type and platform is used in the definition of the client, so that when creating the boot images for the nodes, it can construct the correct one. If the client node is not on the same subnet as the CWS, the `mknimclient` command builds the client routes to the CWS so that the node can use the lppsource.

Command Syntax:

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

`-h`	Display command syntax.
`-l`	`<node_number_list>` List of node numbers to define as NIM clients. Node 0 (the CWS) is not allowed.

The general flow and logic for `mknimclient` is:

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

For each specified node:

1. Determine the client NIM master.

2. Check prerequisites and conditions:
   - Client server must have the NIM master fileset installed.
   - Client server must be configured as a NIM master.
   - Client and boot server must be on the same subnet.
   - If the client is already defined on the server, skip this node.
3. Define the client on the master. Execute:

```
nim -o define -t standalone -a platform=<platform> -a
netboot_kernel=<kernel_type> -a if1='<net_name>
<client_short_hostname> <hdwr_ent_addr> ent' -a
cable_type=<cable_type> <client_short_hostname>
```

4. If the client is not directly attached to any CWS interface, define routing to the CWS. Execute:

```
nim -Fo change -a routing<seq>="spnet_en0 <default_route>
<gateway_ip>" <CW_network>
```

A.10 mkconfig

This wrapper creates the /tftpboot/<reliable_hostname>.config_info files for every node that has bootp_response not set to disk. These files are used when running pssp_script. The information is retrieved from the SDR for every node with bootp_response not set to disk.

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

An example of the /tftpboot/<hostname>.config_info file is given in Figure 185:

```
1 sp5n01 192.168.5.1 192.168.5.150 0 1 5 3 4 yes rootvg false hdisk0 1
false
en0 192.168.5.1 255.255.255.0 NA 10 "" 192.168.5.0
css0 192.168.15.1 255.255.255.0 NA NA "" 192.168.15.0
en1 192.168.6.1 255.255.255.0 NA 10 "" 192.168.6.0
```

Figure 185. Sample config_info file

The fields of the config_info file contain the following:

- Node number
- Initial hostname

- Node IP address
- Default route
- Switch node number
- Switch number (switch board)
- Switch chip
- Switch chip port
- Slot that this node uses (physically)
- If ARP is enabled for the switch

This additional information is only for clients running PSSP 3.1 or later:

- Selected volume group
- Quorum
- Physical volume list
- Number of copies (mirroring)
- Volume group mapping

An entry for each adapter defined in the SDR lists the following:

- Adapter name
- Adapter IP address
- netmask
- ring_speed (for Token Ring)
- bnc_select (dix/bnc/tp for Ethernet)

This additional information is only for clients running PSSP 3.1 or later:

- Ethernet rate
- Duplex type

Command Syntax:

```
mkconfig
```

This command has no input parameters.

The general flow and logic for `mkconfig` is:

1. Investigate prerequisites and conditions:
    - NIM master must be a boot/install server (see SDR).

  - Make sure you have dsh access to the node.
  - For all nodes with bootp_response != disk:

2. Get SDR data for each node (see previous list).
3. Add the arp_enabled field to the definition of each client.
4. Create the /tftpboot/<reliable_hostname>.config_info file for each client.

A.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 by pssp_script. The `mkinstall` wrapper retrieves site environment data from the SP SDR class. It gets the node information, and if the node 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. The `mkinstall` wrapper uses reliable_hostname.

An example of the /tftpboot/<hostname>.install_info file is shown in Figure 186.

```
#!/bin/ksh
export control_workstation="9.12.1.150 192.168.5.150"
export cw_hostaddr="192.168.5.150"
export cw_hostname="sp5en0.msc.itso.ibm.com"
export server_addr="192.168.5.150"
export server_hostname="sp5en0.msc.itso.ibm.com"
export rel_addr="192.168.5.1"
export rel_hostname="sp5n01.msc.itso.ibm.com"
export initial_hostname="sp5n01"
export auth_ifs="sp5en0.msc.itso.ibm.com/192.168.5.150"
export authent_server="ssp"
export netinst_boot_disk="hdisk0"
export netinst_bosobj="bos.obj.ssp.432"
export remove_image="false"
export sysman="true"
export code_version="PSSP-3.1"
export proctype="MP"
export lppsource_name="aix432"
export cwsk4=""
export lppsource_hostname="sp5en0.msc.itso.ibm.com"
export lppsource_addr="192.168.5.150"
export platform="rs6k"
export ssp_jm="yes"
```

Figure 186. Sample install_info file

Command Syntax:

```
mkinstall
```

This command has no input parameters.

The general flow and logic for `mkinstall` is:

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.st 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.

A.12 export_clients

This wrapper ensures that the required directories (pssplpp, images, lppsource, SPOT, and others) are exported from a NIM master to its clients. This command must be run on the NIM master.

Command Syntax:

```
export_clients [-h]
```

This command has no input parameters.

The general flow and logic for `export_clients` is:

1. Get information from the SDR.
2. Parse command line parameters.
3. Check prerequisites:

   Node must be a NIM master.
4. Export the required file systems from the server to all its clients.

A.13 allnimres

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

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

For each client that has a bootp_response different from disk or customize, the following actions will be performed:

- The /etc/bootptab file is updated.
- The boot image file /tftpboot/<reliable_hostname> is created.
- The NIM info file /tftpboot/<reliable_hostname>.info is created.

Command Syntax:

```
allnimres -l <list_of_nodes>
```

The general flow and logic for `allnimres` is:

1. Get information from the SDR.
2. Parse command line parameters.
3. Check prerequisites:
    - Make sure client node is a valid SP node.
    - Make sure client server is a valid SP node.
    - Make sure you have dsh access to the client's server.
    - Server has NIM filesets installed and is configured.
    - If client is already defined on server, skip this node.
    - Client and server must be on the same subnet.
    - Client's server must have NIM master fileset installed.
    - Client's server must be configured as a NIM master.

For each specified node:

1. Reset the node. Execute: `nim -Fo reset <node_name>`
2. Deallocate all resources from the node. Execute:

    ```
    nim -Fo deallocate -a subclass=all <node_name>
    ```

3. Allocate the NIM resources necessary for the assigned bootp operation to be performed.
    - For install operation, allocate the following resources: lppsource, spot, mksysb, psspscript, noprompt.
    - For migrate operation, allocate the following resources: lppsource, spot, psspscript, migrate.
    - For maintenance operation, allocate the following resources: lppsource, spot, prompt.

- For diag operation, allocate the following resources: spot, prompt.
- For disk operation, allocate the following resources: None
- For customize operation, allocate the following resources: None

To allocate the lppsource resource object, execute:

```
nim -o allocate -a lpp_source=lppsource_<aixlvl> <node_name>
```

To allocate the SPOT resource object, execute:

```
nim -o allocate -a spot=spot_<aixlvl> <node_name>
```

To allocate the mksysb resource object, execute:

```
nim -o allocate -a mksysb=mksysb_<seq> <node_name>
```

To allocate the psspscript resource object, execute:

```
nim -o allocate -a script=psspscript <node_name>
```

To allocate the bosinst_data resource type object, execute:

```
nim -o allocate -a bosint_data=<bosinst_data> node_name>
```

4. Boot the node in the assigned bootp_response mode:
   - Install: Perform a bos_inst operation for the client by executing:

     ```
     nim -o bos_inst -a no_client=yes -a source=mksysb <node_name>
     ```
   - Migrate: Perform a bos_inst operation for the client by executing:

     ```
     nim -o bos_inst -a no_client=yes -a source=rte <node_name>
     ```
   - Maintenance: Perform a bos_inst operation for the client by executing:

     ```
     nim -o bos_inst -a no_client=yes <node_name>
     ```
   - Diag: Perform a diag operation for the client by executing:

     ```
     nim -o diag <node_name>
     ```

A.14 unallnimres

This wrapper deallocates all NIM resources to a list of boot/install clients.

Command Syntax:

```
unallnimres -l <list_of_nodes>
```

The general flow and logic for `unallnimres` is:

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

For each specified node:

- Reset NIM on the node. Execute: `nim -Fo reset <node_name>`
- Unallocate all NIM resources. Execute:

  ```
  nim -o deallocate -a subclass=all <node_name>
  ```

Appendix B. NIM output for the C45 case study

In this appendix, we present the output of the node console with NIM debug set used in the case study described in “LED C45 problem” on page 153.

```
instruction interrupt.
>0> st 001f3f6c 2
>0> go
LED{815}
LED{FFF}
LED{814}
AIX Version 4.3
Starting NODE#000 physical CPU#001 as logical CPU#001... done.
Starting NODE#000 physical CPU#002 as logical CPU#002... done.
Starting NODE#000 physical CPU#003 as logical CPU#003... done.
+ [ 1 -ne 1 ]
+ PHASE=1
+ + bootinfo -p
PLATFORM=chrp
+ [ ! -x /usr/lib/boot/bin/bootinfo_chrp ]
+ [ 1 -eq 1 ]
+ 1> /usr/lib/libc.a
+ init -c unlink /usr/lib/boot/bin/!(*_chrp)
+ + bootinfo -t
BOOTYPE=5
+ [ 0 -ne 0 ]
+ [ -z 5 ]
+ chramfs -t
+ init -c unlink /usr/sbin/chramfs
+ 1> /dev/null
+ unset NIM_DEBUG
+ export NIM_DEBUG=set -x
+ unset fd_invoker
+ set -x
+ /usr/lib/methods/showled 0x600
 showled LED{600}
+ bootinfo -b
+ [  = atm0 ]
+ cfgmgr -fv
 cfgmgr LED{538}
 cfgmgr LED{539}
 cfgmgr LED{538}
 cfgsys_chrp LED{811}
 cfgmgr  cfgmgr  cfgbus_pcic  cfgbus_isac  cfgbus_pcic  cfgmgr  cfgmgr
cfgmgr Method error (/usr/lib/methods/cfgkma_chrp -1 -l siokma0):
sh: /usr/lib/methods/cfgkma_chrp:  not found
 cfgmgr  cfgmgr Method error (/usr/lib/methods/cfgfda_chrp -1 -l fda0):
sh: /usr/lib/methods/cfgfda_chrp:  not found
 cfgmgr  cfgmgr Method error (/usr/lib/methods/cfgncr_scsi -1 -l scsi0):
sh: /usr/lib/methods/cfgncr_scsi:  not found
 cfgmgr  cfgkent  cfgmgr  cfgmgr  cfgphxent  cfgmgr  cfgmgr  cfgmgr
Method error
 (/usr/lib/methods/deflvm):
        0514-068 Cause not known.
sh: /usr/lib/methods/deflvm:  not found
 cfgmgr  cfgmgr Method error (/usr/lib/methods/defssar):
        0514-068 Cause not known.
sh: /usr/lib/methods/defssar:  not found
 cfgmgr  cfgmgr Method error (/usr/lib/methods/deftmssar):
        0514-068 Cause not known.
sh: /usr/lib/methods/deftmssar:  not found
```

```
 cfgmgr cfgmgr is running in phase 1
--------------------------------------------------------------------
invoking top level program -- "/etc/methods/defsys"
return code = 0
****************** stdout ***********
sys0
****************** no stderr ***********
-------------------------------------------------------------------
attempting to configure device 'sys0'
invoking /usr/lib/methods/cfgsys_chrp -1 -l sys0
return code = 0
****************** stdout ***********
pci0
:devices.chrp.IBM.TB3MX:devices.chrp.IBM.TB3MX
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'pci0'
invoking /usr/lib/methods/cfgbus_pcic -1 -l pci0
return code = 0
****************** stdout ***********
:devices.isa_sio.serial:devices.isa_sio.pnpPNP.501
:devices.isa_sio.serial:devic
es.isa_sio.pnpPNP.501
:devices.pci.ssa:devices.pci.14104500:devices.pci.0c000c02
siokma0,fda0,scsi0,ent0,ent1
****************** no stderr ***********
--------------------------------------------------------------------att
empting to configure device 'siokma0'
invoking /usr/lib/methods/cfgkma_chrp -1 -l siokma0
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgkma_chrp:  not found
--------------------------------------------------------------------att
empting to configure device 'fda0'
invoking /usr/lib/methods/cfgfda_chrp -1 -l fda0
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgfda_chrp:  not found
--------------------------------------------------------------------att
empting to configure device 'scsi0'
invoking /usr/lib/methods/cfgncr_scsi -1 -l scsi0
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgncr_scsi:  not found
--------------------------------------------------------------------
attempting to configure device 'ent0'
invoking /usr/lib/methods/cfgkent -1 -l ent0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'ent1'
invoking /usr/lib/methods/cfgphxent -1 -l ent1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deflvm"
return code = 127
****************** no stdout ***********
```

```
****************** stderr ***********
sh: /usr/lib/methods/deflvm:  not found
--------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defssar"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/defssar:  not found
--------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deftmssar"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/deftmssar:  not found
+ export NSORDER=local
+ bootinfo -b
+ [ ent0 = atm0 ]
+ native_netboot_cfg
+ bootinfo -c
+ set -- 9.114.117.66 9.114.117.125 9.114.117.125 FF FF 0
/tftpboot/fs219n02.ppd
.pok.ibm.com
99.130.83.99.1.4.255.255.255.192.255.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
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
+ CLIENT_IPADDR=9.114.117.66
+ BOOT_SERV_IP=9.114.117.125
+ BOOT_GATE_IP=9.114.117.125
+ E802=0
+ BOOTFILE=/tftpboot/fs219n02.ppd.pok.ibm.com
VEND=99.130.83.99.1.4.255.255.255.192.255.0.0.0.0.0.0.0.0.0.0.0.0.0
.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.
+ [ -n 255.255.255.192 ]
+ SUBMASK=netmask 255.255.255.192
+ [ 9.114.117.125 = 0 -o 9.114.117.125 = 0.0.0.0 -o 9.114.117.125 =
9.114.117.125 ]
+ unset BOOT_GATE_IP
+ + bootinfo -b
PHY_BOOT_DEV=ent0
+ pdev_to_ldev
+ /usr/lib/methods/showled 0x606
 showled + ifconfig lo0 inet 127.0.0.1 up
+ ifconfig en0 inet 9.114.117.66 up netmask 255.255.255.192
+ [ 0 -ne 0 ]
+ [  ]
+ [ -n  ]
+ CLIENT_INFO_FILE=/tftpboot/fs219n02.ppd.pok.ibm.com.info
+ [ 0 -ne 0 ]
+ /usr/lib/methods/showled 0x608
 showled + tftp -go /SPOT/niminfo 9.114.117.125
/tftpboot/fs219n02.ppd.pok.ibm.com.info image
Received 1326 Bytes in 0.1 Seconds
+ [ -s /SPOT/niminfo ]
+ . /SPOT/niminfo
+ export NIM_NAME=fs219n02
+ export NIM_HOSTNAME=fs219n02.ppd.pok.ibm.com
+ export NIM_CONFIGURATION=standalone
+ export NIM_MASTER_HOSTNAME=fs219cw.ppd.pok.ibm.com
+ export NIM_MASTER_PORT=1058
+ export NIM_REGISTRATION_PORT=1059
+ 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=fs219n02.ppd.pok.ibm.com
+ export
SPOT=fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/aix432/spot/spot_aix4
32/usr
+ export NIM_BOSINST_DATA=/NIM_BOSINST_DATA
+ export NIM_CUSTOM=/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script -a
location
=fs219cw.ppd.pok.ibm.com:/export/nim/scripts/fs219n02.script
+ export NIM_BOS_IMAGE=/NIM_BOS_IMAGE
+ export NIM_BOS_FORMAT=mksysb
+ export NIM_HOSTS= 9.114.117.66:fs219n02.ppd.pok.ibm.com
9.114.117.125:fs219cw.ppd.pok.ibm.com
+ export NIM_MOUNTS=
fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/aix432/lppsource:/SPOT/usr
/sys/inst.images:dir fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/p
ssp/bosinst_data:/NIM_BOSINST_DATA:file
fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.432:/NI
M_BOS_IMAGE:file
+ [ -n  9.114.117.66:fs219n02.ppd.pok.ibm.com
9.114.117.125:fs219cw.ppd.pok.ibm.com  ]
+ OIFS=
+ IFS=:
+ set -- 9.114.117.66 fs219n02.ppd.pok.ibm.com
+ IFS=
+ echo 9.114.117.66 fs219n02.ppd.pok.ibm.com
+ 1>> /etc/hosts
+ OIFS=
+ IFS=:
+ set -- 9.114.117.125 fs219cw.ppd.pok.ibm.com
+ IFS=
+ echo 9.114.117.125 fs219cw.ppd.pok.ibm.com
+ 1>> /etc/hosts
+ [ -n  ]
+ 1> /etc/filesystems
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=LED 610:
mount -r
fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/aix432/spot/spot_aix432/us
r /SPOT/usr
+ /usr/lib/methods/showled 0x610
 showled + mount -r
fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/aix432/spot/spot_aix432/us
r /SPOT/usr
+ [[ 0 -ne 0 ]]
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=
+ /usr/lib/methods/showled 0x612
 showled
+ cp /SPOT/usr/lib/boot/network/rc.bos_inst /etc
+ RC_CONFIG=/etc/rc.bos_inst
+ . /etc/rc.bos_inst
+ set -x
+ /SPOT/usr/sbin/nimclient -S booting
+ mount_from_list
+ [ -n
fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/aix432/lppsource:/SPOT/usr
/
sys/inst.images:dir
fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/pssp/bosinst_d
ata:/NIM_BOSINST_DATA:file
fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/images/
bos.obj.ssp.432:/NIM_BOS_IMAGE:file  ]
+ /usr/lib/methods/showled 0x610
 showled + OIFS=
+ IFS=:
+ set -- fs219cw.ppd.pok.ibm.com /spdata/sys1/install/aix432/lppsource
/SPOT/usr
/sys/inst.images dir
+ IFS=
+ [ ! -d /SPOT/usr/sys/inst.images ]
+ /SPOT/usr/sbin/nimclient -o change -a force=yes -a ignore_lock=yes -a
info=LED
 610: mount
fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/aix432/lppsource /SPOT/
usr/sys/inst.images
+ mount fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/aix432/lppsource
/SPOT/usr/sys/inst.images
+ [[ 0 -ne 0 ]]
+ IFS=:
+ set -- fs219cw.ppd.pok.ibm.com /spdata/sys1/install/pssp/bosinst_data
/NIM_BOS
INST_DATA file
+ IFS=
+ [ ! -d /NIM_BOSINST_DATA ]
+ /SPOT/usr/bin/mkdir -p /NIM_BOSINST_DATA
+ /SPOT/usr/sbin/nimclient -o change -a force=yes -a ignore_lock=yes -a
info=LED
 610: mount
fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/pssp/bosinst_data /NIM_
BOSINST_DATA
+ mount fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/pssp/bosinst_data
/NIM_BOSI
NST_DATA
+ [[ 0 -ne 0 ]]
+ IFS=:
+ set -- fs219cw.ppd.pok.ibm.com
/spdata/sys1/install/images/bos.obj.ssp.432 /NI
M_BOS_IMAGE file
+ IFS=
+ [ ! -d /NIM_BOS_IMAGE ]
+ /SPOT/usr/bin/mkdir -p /NIM_BOS_IMAGE
+ /SPOT/usr/sbin/nimclient -o change -a force=yes -a ignore_lock=yes -a
info=LED 610: mount
fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.432
/NIM_BOS_IMAGE
+ mount
fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.432
/NIM_BOS_IMAGE
+ [[ 0 -ne 0 ]]
+ /SPOT/usr/sbin/nimclient -o change -a force=yes -a ignore_lock=yes -a
info=
+ /usr/lib/methods/showled 0x622
 showled ln: /usr/lib is a directory.  (cannot unlink)
ln: /usr/lib/methods is a directory.  (cannot unlink)
+ /SPOT/usr/lib/boot/network/link_methods
+ strload -f /dev/null
+ cfgmgr -f -v
 cfgmgr  cfgmgr  cfgmgr  cfgsys_chrp  cfgmgr  cfgmgr  cfgbus_pcic
cfgbus_isac
cfgbus_pcic  cfgmgr  cfgmgr  cfgkent  cfgmgr  cfgmgr  cfgphxent  cfgmgr
cfgmgr
 cfgkma_chrp  cfgmgr  cfgmgr  cfgfda_chrp  cfgmgr  cfgmgr  cfgncr_scsi
cfgmgr
```

```
cfgmgr  cfgkm_chrp  cfgmgr  cfgmgr  cfgkm_chrp  cfgmgr  cfgmgr  cfgscdisk
cfgmg
r  cfgmgr  cfgscdisk  cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfglvdd  cfgmgr
cfgmgr  c
fgmgr cfgmgr is running in phase 1
--------------------------------------------------------------------
invoking top level program -- "/etc/methods/defsys"
return code = 0
****************** stdout ***********
sys0
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'sys0'
invoking /usr/lib/methods/cfgsys_chrp -1 -l sys0
return code = 0
****************** stdout ***********
pci0
:devices.chrp.IBM.TB3MX:devices.chrp.IBM.TB3MX
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'pci0'
invoking /usr/lib/methods/cfgbus_pcic -1 -l pci0
return code = 0
****************** stdout ***********
:devices.isa_sio.serial:devices.isa_sio.pnpPNP.501
:devices.isa_sio.serial:devic
es.isa_sio.pnpPNP.501
:devices.pci.ssa:devices.pci.14104500:devices.pci.0c000c02
ent0,ent1,siokma0,fda0,scsi0
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'ent0'
invoking /usr/lib/methods/cfgkent -1 -l ent0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'ent1'
invoking /usr/lib/methods/cfgphxent -1 -l ent1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'siokma0'
invoking /usr/lib/methods/cfgkma_chrp -1 -l siokma0
return code = 0
****************** stdout ***********
sioka0 sioma0
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'fda0'
invoking /usr/lib/methods/cfgfda_chrp -1 -l fda0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'scsi0'
invoking /usr/lib/methods/cfgncr_scsi -1 -l scsi0
return code = 0
****************** stdout ***********
hdisk0 hdisk1
****************** no stderr ***********
--------------------------------------------------------------------
```

```
attempting to configure device 'sioka0'
invoking /usr/lib/methods/cfgkm_chrp -1 -l sioka0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'sioma0'
invoking /usr/lib/methods/cfgkm_chrp -1 -l sioma0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'hdisk0'
invoking /etc/methods/cfgscdisk -1 -l hdisk0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'hdisk1'
invoking /etc/methods/cfgscdisk -1 -l hdisk1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deflvm"
return code = 0
****************** stdout ***********
lvdd
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'lvdd'
invoking /usr/lib/methods/cfglvdd -1 -l lvdd
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defssar"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
---------------------------------------- cfgmgr  cfgmgr Method error
(/usr/lib/methods/deftmssar):
        0514-068 Cause not known.
sh: /usr/lib/methods/deftmssar:  not found
 cfgmgr ------------------------------
invoking top level program -- "/usr/lib/methods/deftmssar"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/deftmssar:  not found
+ /usr/lib/methods/showled 0x622
 showled + /SPOT/usr/lib/boot/network/link_methods
+ export DEV_PKGNAME=ALL
+ cfgmgr -s -v
 cfgmgr  cfgmgr Method error (/etc/methods/cfgprobe -c
/etc/drivers/coreprobe.ex
t):
        0514-068 Cause not known.
sh: /etc/methods/cfgprobe:  not found
 cfgmgr  cfgmgr  cfgmgr  cfgsys_chrp  cfgmgr  cfgmgr  cfgmgr Method error
(/usr/lib/methods/cfgpmchrp -2 -l pmc0):
sh: /usr/lib/methods/cfgpmchrp:  not found
```

```
 cfgmgr  cfgbus_pcic  cfgbus_isac  cfgbus_pcic  cfgmgr  cfgmgr
cfgkma_chrp  cfg
mgr  cfgmgr  cfgfda_chrp  cfgmgr  cfgmgr  cfgncr_scsi  cfgmgr  cfgmgr
cfgkent
cfgmgr  cfgmgr  cfgphxent  cfgmgr  cfgmgr  cfgkm_chrp  cfgmgr  cfgmgr
cfgkm_chr
p  cfgmgr  cfgmgr  cfgscdisk  cfgmgr cfgmgr is running in phase 2
--------------------------------------------------------------------
invoking top level program -- "/etc/methods/cfgprobe -c
/etc/drivers/coreprobe.ext"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/cfgprobe:  not found
--------------------------------------------------------------------
invoking top level program -- "/etc/methods/defsys"
return code = 0
****************** stdout ***********
:devices.chrp.base sys0
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'sys0'
invoking /usr/lib/methods/cfgsys_chrp -2 -l sys0
return code = 0
****************** stdout ***********
:devices.common.IBM.pmmd_chrp
pmc0
:devices.chrp.pci
pci0
:devices.chrp.IBM.TB3MX:devices.chrp.IBM.TB3MX
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'pmc0'
invoking /usr/lib/methods/cfgpmchrp -2 -l pmc0
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgpmchrp:  not found
--------------------------------------------------------------------
attempting to configure device 'pci0'
invoking /usr/lib/methods/cfgbus_pcic -2 -l pci0
return code = 0
****************** stdout ***********
:devices.pci.isa
:devices.isa_sio.serial:devices.isa_sio.pnpPNP.501
:devices.isa_sio.serial:devic
es.isa_sio.pnpPNP.501 :devices.isa_sio.8042:devices.isa_sio.chrp.8042
:devices.i
sa_sio.fdc:devices.isa_sio.pnpPNP.700
:devices.pci.scsi:devices.pci.00100300:dev
ices.pci.NCR.53C825:devices.pci.SYM.53C825:devices.pci.NCR.8251S:device
s.pci.SYM
.8251S:devices.pci.AAPL.NCR8250S
:devices.pci.ethernet:devices.pci.22100020
:devices.pci.ssa:devices.pci.14104500:devices.pci.0c000c02
:devices.pci.ethernet:devices.pci.23100020
siokma0,fda0,scsi0,ent0,ent1
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'siokma0'
invoking /usr/lib/methods/cfgkma_chrp -2 -l siokma0
return code = 0
****************** stdout ***********
```

```
sioka0 sioma0
****************** no stderr ***********
-------------------------------------------------------------------
attempting to configure device 'fda0'
invoking /usr/lib/methods/cfgfda_chrp -2 -l fda0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
-------------------------------------------------------------------
attempting to configure device 'scsi0'
invoking /usr/lib/methods/cfgncr_scsi -2 -l scsi0
return code = 0
****************** stdout ***********
:devices.scsi.disk :devices.scsi.disk hdisk0 hdisk1 :devices.scsi.tape
****************** no stderr ***********
-------------------------------------------------------------------
attempting to configure device 'ent0'
invoking /usr/lib/methods/cfgkent -2 -l ent0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
-------------------------------------------------------------------
attempting to configure device 'ent1'
invoking /usr/lib/methods/cfgphxent -2 -l ent1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
------------------------------------------------------------------
attempting to configure device 'sioka0'
invoking /usr/lib/methods/cfgkm_chrp -2 -l sioka0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
-------------------------------------------------------------------
attempting to configure device 'sioma0'
invoking /usr/lib/methods/cfgkm_chrp -2 -l sioma0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
-------------------------------------------------------------------
attempting to configure device 'hdisk0'
invoking /etc/methods/cfgscdisk -2 -l hdisk0
return code = 0
************* cfgmgr  cfgscdisk  cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfgmgr
Method e
rror (/usr/lib/methods/defops):
        0514-068 Cause not known.
sh: /usr/lib/methods/defops:  not found
 cfgmgr  cfgmgr  cfgmgr  cfgmgr Method error (/usr/lib/methods/definet >
/dev/nu
ll 2>&1;opt=`/usr/sbin/lsattr -E -l inet0 -a bootup_option -F value`
        if [ $opt = "no" ];then nf=/etc/rc.net
        else nf=/etc/rc.bsdnet
       fi;$nf -2;x=$?;test $x -ne 0&&echo $nf failed. Check for invalid
command
s >&2;exit $x):
        0514-068 Cause not known.
lsattr: 0514-519 The following device was not found in the customized
        device configuration database:
        inet0
sh[2]: test: argument expected
sh[4]: /etc/rc.bsdnet:  not found
/etc/rc.bsdnet failed. Check for invalid commands
```

```
 cfgmgr  cfgmgr Method error (/etc/methods/ptynode):
        0514-068 Cause not known.
sh: /etc/methods/ptynode:  not found
 cfgmgr  cfgmgr  cfgmgr  cfgmgr Method error (/etc/methods/startrcm):
        0514-068 Cause not known.
sh: /etc/methods/startrcm:  not found
 cfgmgr  cfgmgr  cfgmgr  cfgmgr Method error (/etc/methods/startsmt):
        0514-068 Cause not known.
sh: /etc/methods/startsmt:  not found
 cfgmgr  cfgmgr Method error (/etc/methods/startsgio):
        0514-068 Cause not known.
sh: /etc/methods/startsgio:  not found
 cfgmgr  cfgmgr Method error (/usr/lib/methods/fdarcfgrule):
        0514-068 Cause not known.
sh: /usr/lib/methods/fdarcfgrule:  not found
 cfgmgr  cfgmgr Method error (/etc/methods/darcfgrule):
        0514-068 Cause not known.
sh: /etc/methods/darcfgrule:  not found
 cfgmgr  cfgmgr  cfgmgr  cfgmgr Method error
(/usr/lib/methods/deftmssar):
        0514-068 Cause not known.
sh: /usr/lib/methods/deftmssar:  not found
***** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
attempting to configure device 'hdisk1'
invoking /etc/methods/cfgscdisk -2 -l hdisk1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deflvm"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defops"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/defops:  not found
--------------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -Cc ipsec -r name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/definet > /dev/null
2>&1;opt=`/u
sr/sbin/lsattr -E -l inet0 -a bootup_option -F value`
        if [ $opt = "no" ];then nf=/etc/rc.net
        else nf=/etc/rc.bsdnet
       fi;$nf -2;x=$?;test $x -ne 0&&echo $nf failed. Check for invalid
command
s >&2;exit $x"
return code = 127
****************** no stdout ***********
****************** no stderr ***********
--------------------------------------------------------------------
invoking top level program -- "/etc/methods/ptynode"
return code = 127
****************** no stdout ***********
****************** stderr ***********
```

```
sh: /etc/methods/ptynode:  not found
------------------------------------------------------------------
invoking top level program -- "/etc/methods/startlft"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
-------------------------------------------------------------------
invoking top level program -- "/etc/methods/startrcm"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/startrcm:  not found
-------------------------------------------------------------------
invoking top level program -- "/etc/methods/starttty"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
-------------------------------------------------------------------
invoking top level program -- "/etc/methods/startsmt"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/startsmt:  not found
-------------------------------------------------------------------
invoking top level program -- "/etc/methods/startsgio"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/startsgio:  not found
-------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/fdarcfgrule"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/fdarcfgrule:  not found
-------------------------------------------------------------------
invoking top level program -- "/etc/methods/darcfgrule"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/darcfgrule:  not found
-------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defssar"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
-------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deftmssar"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/deftmssar:  not found
------------------------------------------------ cfgmgr  cfgmgr Method
error (/
usr/lib/methods/cfgfan):
        0514-068 Cause not known.
sh: /usr/lib/methods/cfgfan:  not found
 cfgmgr  cfgmgr  cfgmgr  cfgmgr Method error (/usr/lib/methods/defaio):
        0514-068 Cause not known.
sh: /usr/lib/methods/defaio:  not found
 cfgmgr  cfgmgr  cfgmgr  cfgmgr Method error
(/etc/methods/load_blockset_ext):
        0514-068 Cause not known.
```

```
sh: /etc/methods/load_blockset_ext:  not found
 cfgmgr ----------------------
invoking top level program -- "/usr/lib/methods/cfgfan"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgfan:  not found
-----------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -C -c port -s tsd -t tsp
-F name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defaio"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/defaio:  not found

-----------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -C -c driver -s sdlc -t
scie -F name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------
invoking top level program -- "/etc/methods/load_blockset_ext"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/load_blockset_ext:  not found
+ /usr/lib/methods/showled 0x622
 showled + exit 0
+ [ 1 -ne 1 ]
+ PHASE=2
+ + bootinfo -p
PLATFORM=chrp
+ [ ! -x /usr/lib/boot/bin/bootinfo_chrp ]
+ [ 2 -eq 1 ]
+ + bootinfo -t
BOOTYPE=5
+ [ 0 -ne 0 ]
+ [ -z 5 ]
+ chramfs -t
/sbin/rc.boot[321]: chramfs:  not found
+ init -c unlink /usr/sbin/chramfs
+ 1> /dev/null
Could not load program init
Symbol setpcred in init is undefined
Symbol _FloatingReleaseLicense in init is undefined
Error was: Exec format error
+ unset NIM_DEBUG
+ export NIM_DEBUG=set -x
+ unset fd_invoker
+ set -x
+ export NSORDER=local
+ + bootinfo -b
PHY_BOOT_DEV=ent0
+ . /SPOT/niminfo
+ export NIM_NAME=fs219n02
+ export NIM_HOSTNAME=fs219n02.ppd.pok.ibm.com
+ export NIM_CONFIGURATION=standalone
+ export NIM_MASTER_HOSTNAME=fs219cw.ppd.pok.ibm.com
+ export NIM_MASTER_PORT=1058
+ export NIM_REGISTRATION_PORT=1059
+ 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=fs219n02.ppd.pok.ibm.com
+ export
SPOT=fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/aix432/spot/spot_aix4
32/usr
+ export NIM_BOSINST_DATA=/NIM_BOSINST_DATA
+ export NIM_CUSTOM=/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script -a
location
=fs219cw.ppd.pok.ibm.com:/export/nim/scripts/fs219n02.script
+ export NIM_BOS_IMAGE=/NIM_BOS_IMAGE
+ export NIM_BOS_FORMAT=mksysb
+ export NIM_HOSTS= 9.114.117.66:fs219n02.ppd.pok.ibm.com
9.114.117.125:fs219cw
.ppd.pok.ibm.com
+ export NIM_MOUNTS=
fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/aix432/lppsour
ce:/SPOT/usr/sys/inst.images:dir
fs219cw.ppd.pok.ibm.com:/spdata/sys1/install/p
ssp/bosinst_data:/NIM_BOSINST_DATA:file
fs219cw.ppd.pok.ibm.com:/spdata/sys1/in
stall/images/bos.obj.ssp.432:/NIM_BOS_IMAGE:file
+ RC_CONFIG=/etc/rc.bos_inst
+ . /etc/rc.bos_inst
+ set -x
+ /SPOT/usr/sbin/no -o tcp_keepintvl=150
+ /SPOT/usr/sbin/no -o tcp_keepidle=1200
+ cp /usr/lpp/diagnostics/obj/CDiagDev /etc/objrepos/CDiagDev
+ cp /usr/lpp/diagnostics/obj/TMInput /etc/objrepos/TMInput
+ cp /usr/lpp/diagnostics/obj/MenuGoal /etc/objrepos/MenuGoal
+ cp /usr/lpp/diagnostics/obj/FRUB /etc/objrepos/FRUB
+ cp /usr/lpp/diagnostics/obj/FRUs /etc/objrepos/FRUs
+ cp /usr/lpp/diagnostics/obj/DAVars /etc/objrepos/DAVars
+ cp /usr/lpp/diagnostics/obj/CDiagAtt /etc/objrepos/CDiagAtt
+ cp /usr/lpp/diagnostics/obj/CDiagAtt.vc /etc/objrepos/CDiagAtt.vc
+ mkdir -p /etc/lpp/diagnostics/data
+ /usr/lib/methods/showled 0xfff
 showled + exec /usr/lpp/bosinst/bi_main
+ typeset +f
+ typeset -ft BI_Clock
+ typeset -ft BI_Copy_Files
+ typeset -ft BI_Error
+ typeset -ft Breakpt
+ typeset -ft Call_Merge_Methods
+ typeset -ft Change_Mounts
+ typeset -ft Change_Status
+ typeset -ft Check_DCE_Level
+ typeset -ft Check_Down_Level
+ typeset -ft Check_Other_Stanzas
+ typeset -ft Check_SNA_Level
+ typeset -ft Clean_Migrated_VPD
+ typeset -ft Copy_Customized
+ typeset -ft Create_Bosinst_Data
+ typeset -ft Display_Status_Screen
+ typeset -ft Extract_Diskette_Data
+ typeset -ft Fill_Target_Stanzas
+ typeset -ft Finish
+ typeset -ft Get_Data_Files
+ typeset -ft Get_Locale_Packages
+ typeset -ft Get_User_Input
+ typeset -ft Image_Data_Exists
+ typeset -ft Init_Target_Disks
+ typeset -ft Initialize
+ typeset -ft Initialize_Disk_Environment
+ typeset -ft Initialize_Log
+ typeset -ft Install_Updates
+ typeset -ft Log
+ typeset -ft Make_Sys_FS
+ typeset -ft Make_Sys_LV
+ typeset -ft Make_Sys_VG
+ typeset -ft Mount_Rootfs
+ typeset -ft Other_Initialization
+ typeset -ft Post_Install
+ typeset -ft Prepare_Target_Disks
+ typeset -ft Restore_Bosinst
+ typeset -ft Restore_System
+ typeset -ft Run_Customization
+ typeset -ft Save_Devs
+ typeset -ft Set_Console
+ typeset -ft Set_PP_Size
+ typeset -ft Set_Primary_Locale
+ typeset -ft Shrink_FS
+ typeset -ft Shrink_It
+ typeset -ft Swapon_hd6
+ typeset -ft Turbo_Restore
+ typeset -ft Update_Status
+ typeset -ft Verify_Image_Parameters
+ typeset -ft Zero_Screen
+ typeset -ft free_dump_lv
+ typeset -ft loopled
+ [ ! -s ./bi_main.debug ]
+ restbyname -xqf - ./bi_main.debug ./startup
+ 1> /dev/null 2>& 1 + [ -s ./bi_main.debug ]
+ a=none
+ [ none = bi_main.debug ]
+ bootinfo -p
+ export PLATFORM=chrp
+ PLAT_HOOK=
+ [ -x /SPOT/platform/chrp/plat_get_debug ]
+ [ -x /SPOT/usr/lpp/bos/inst_root/platform/chrp/plat_get_debug ]
+ [ -x /platform/chrp/plat_get_debug ]
+ [ -n  ]
+ trap BI_Error "BOS Install" 29 2 INT
+ cd /
+ Initialize
+ trap BI_Error "BOS Install" 29 2 INT
+ /usr/lib/methods/showled 0xA46
 showled + umask 002
+ BOSINSTDATA=/bosinst.data
+ BUNDLE_INCR=40
+ CDROM=3
+ CURLEVEL=4.3
+ DEV_INCR=35
+ IMAGEDATA=/image.data
+ INST_INCR=75
+ INSTTASK=2
+ MEG=1048576
+ MKSYSB_INCR=75
+ MROOT=/mnt
+ NETWORK=5
+ NOM_INCR=2
+ NOT=!
+ NOPVID=0000000000000000
+ NOVG=0
+ NOVGID=0000000000000000
+ ODMDIR=/etc/objrepos
+ OTHERVG=2
+ PRE_LIST=/etc/preserve.list
+ REST_INCR=5
+ ROOTVG=1
+ PERCENT=0
+ TASK_INCR=1
+ TIME=0
+ TURBO_INCR=22
+ FATAL=
+ TAPE=4
+ BAR=
+ TAPEBLKSZ=/tapeblksz
+ TAPEDEVICE=/tapedevice
+ TARGETVGS=/tmp/targetvgs
+ BOS_FORMAT=nonturbo
+ export BOSINSTDATA CDROM IMAGEDATA INSTTASK
+ export MEG MROOT NETWORK NOPVID NOT NOVG NOVGID
+ export OTHERVG ROOTVG TAPE TAPEBLKSZ TARGETVGS ODMDIR TAPEDEVICE
+ export TIME PERCENT BAR FATAL PRE_LIST CURLEVEL BOS_FORMAT
+ export DEBUG=
+ export SYSCFG_PHASE=BOSINST
+ + bootinfo -r
REALMEM=1048576
+ export REALMEM
+ MIN_PGSP=32
+ export MIN_PGSP
+ [ ! -s /tapedevice ]
+ set -x
+ unset BOOTDEV BOOTTYPE
+ [ -s /tapedevice ]
+ + /usr/sbin/bootinfo -t
BOOTTYPE=5
+ + /usr/sbin/bootinfo -b
BOOTDEV=ent0
+ export BOSMENU_LANG=
+ ORIG_BOOTTYPE=5
+ ORIG_BOOTDEV=ent0
+ export ORIG_BOOTTYPE ORIG_BOOTDEV
+ export BOOTTYPE BOOTDEV
+ + bootinfo -z
+ 2> /dev/null
BOOT_Z=1
+ export BOOT_Z
+ [ 5 -eq 5 ]
+ [  -ne 1 ]
+ nimclient -R success
+ /usr/lib/methods/showled 0xA40
 showled + [ 5 -eq 4 ]
+ Get_Data_Files
+ trap BI_Error "BOS Install" 29 2 INT
+ nimclient -o change -a force=yes -a ignore_lock=yes -a
info=extract_data_files
+ [  -eq 1 ]
+ [ /NIM_BOS_IMAGE -a mksysb ]
+ restbyname -xqdSf - ./bosinst.data ./image.data ./tmp/vgdata
+ dd if=/NIM_BOS_IMAGE bs=1k count=128
+ 1> /dev/null 2>& 1
```

```
128+0 records in
128+0 records out
+ [ -s /NIM_BOSINST_DATA ]
+ cp /NIM_BOSINST_DATA /bosinst.data
+ 1> /dev/null 2>& 1
+ [ -s  ]
+ [ -z  ]
+ cp /SPOT/usr/lpp/bos/inst_root/etc/preserve.list /etc/preserve.list
+ 1> /dev/null 2>& 1
+ return 0
+ [ 5 -eq 5 ]
+ nimclient -o change -a force=yes -a ignore_lock=yes -a
info=query_disks
+ rm -f /tmp/Get_RVG_Disks.pn
+ bootinfo -r
+ [ 1048576 -le 8192 ]
+ echo 3002
+ /usr/lpp/bosinst/Get_RVG_Disks
+ 1> /tmp/Get_RVG_Disks.pn
+ [ -s /bosinst.data -a 5 -ne 5 ]
+ unset IMAGE_ON_DISKETTE
+ unset PRESERVE_ON_DISKETTE
+ PLAT_HOOK=
+ [ -x /SPOT/platform/chrp/plat_get_data ]
+ [ -x /SPOT/usr/lpp/bos/inst_root/platform/chrp/plat_get_data ]
+ [ -x /platform/chrp/plat_get_data ]
+ [ -n  ]
+ [ -s /platform/chrp/bosinst.data ]
+ [ -s /platform/chrp/image.data ]
+ [ -s /platform/chrp/preserve.list ]
+ /usr/lib/methods/showled 0xA42
 showled + Extract_Diskette_Data
+ trap BI_Error "BOS Install" 29 2 INT
+ [ 5 -eq 5 ]
+ nimclient -o change -a force=yes -a ignore_lock=yes -a
info=extract_diskette_d
ata
+ cd /
+ rm -f ./signature
+ restbyname -xqSf /dev/rfd0 ./signature
+ 1> /dev/null 2>& 1
+ [ 1 -eq 0 -a -s ./signature ]
+ [ -f /usr/bin/dosread ]
+ return 0
+ [ ! -s /bosinst.data ]
+ [ -n  ]
+ Image_Data_Exists
+ trap BI_Error "BOS Install" 29 2 INT
+ [ ! -s /image.data ]
+ return 0
+ /usr/lpp/bosinst/datadaemon
+ sleep 1
+ [ -n  ]
+ Set_Console
+ trap BI_Error "BOS Install" 29 2 INT
+ [ 5 -eq 5 ]
+ nimclient -o change -a force=yes -a ignore_lock=yes -a
info=setting_console
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f CONSOLE
C=/dev/tty0
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f PROMPT
P=no
+ [ /dev/tty0 -a no = no ]
+ [[ /dev/tty0 = none ]]
+ [[ 5 -eq 5 ]]
+ [[ /dev/tty0 != /dev/?* ]]
+ chcons /dev/tty0
chcons: invalid request - Console login is enabled and
        the specified device is not an available lft or tty.
+ /usr/lib/methods/cfgcon
cfgcon: console is not defined and no candidate terminals were detected
+ loopled 0xA45
+ [ 5 -eq 5 ]
+ nimclient -R failure
+ /etc/methods/showled 0xA45
 showled + :
```

Appendix C. NIM debug console output

This appendix shows the NIM debug output from an MCA node and a CHRP node when there is no problem.

The MCA node output does not contain information after the `bi_main` start since this is exactly the same as the output from the CHRP node.

CHRP NIM Debug Output

```
st 001f3d1c 2
>0> go
LED{815}
LED{FFF}
LED{814}

AIX Version 4.3
Starting NODE#000 physical CPU#001 as logical CPU#001... done.
Starting NODE#000 physical CPU#002 as logical CPU#002... done.
Starting NODE#000 physical CPU#003 as logical CPU#003... done.
+ [ 1 -ne 1 ]
+ PHASE=1
+ + bootinfo -p
PLATFORM=chrp
+ [ ! -x /usr/lib/boot/bin/bootinfo_chrp ]
+ [ 1 -eq 1 ]
+ 1> /usr/lib/libc.a
+ init -c unlink /usr/lib/boot/bin/!(*_chrp)
+ + bootinfo -t
BOOTYPE=5
+ [ 0 -ne 0 ]
+ [ -z 5 ]
+ chramfs -t
+ init -c unlink /usr/sbin/chramfs
+ 1> /dev/null
+ unset NIM_DEBUG
+ export NIM_DEBUG=set -x
+ unset fd_invoker
+ set -x
+ /usr/lib/methods/showled 0x600
 showled LED{600}
+ bootinfo -b
+ [  = atm0 ]
+ cfgmgr -fv
 cfgmgr LED{538}
 cfgmgr LED{539}
 cfgmgr LED{538}
 cfgsys_chrp LED{811}
 cfgmgr  cfgmgr  cfgbus_pcic  cfgbus_isac  cfgbus_pcic  cfgmgr  cfgmgr  cfgmgr
Method error (/usr/lib/methods/cfgasync_rspc -1 -l sa0):
sh: /usr/lib/methods/cfgasync_rspc:  not found

 cfgmgr  cfgmgr Method error (/usr/lib/methods/cfgasync_rspc -1 -l sa1):
sh: /usr/lib/methods/cfgasync_rspc:  not found

 cfgmgr  cfgmgr Method error (/usr/lib/methods/cfgkma_chrp -1 -l siokma0):
sh: /usr/lib/methods/cfgkma_chrp:  not found

 cfgmgr  cfgmgr Method error (/usr/lib/methods/cfgfda_chrp -1 -l fda0):
sh: /usr/lib/methods/cfgfda_chrp:  not found
```

```
 cfgmgr  cfgmgr Method error (/usr/lib/methods/cfgncr_scsi -1 -l scsi0):
sh: /usr/lib/methods/cfgncr_scsi:  not found

 cfgmgr  cfgkent  cfgmgr  cfgmgr  cfgmgr Method error (/usr/lib/methods/deflvm ):
0514-068 Cause not known.
sh: /usr/lib/methods/deflvm:  not found

 cfgmgr  cfgmgr Method error (/usr/lib/methods/defssar ):
0514-068 Cause not known.
sh: /usr/lib/methods/defssar:  not found

 cfgmgr  cfgmgr Method error (/usr/lib/methods/deftmssar ):
0514-068 Cause not known.
sh: /usr/lib/methods/deftmssar:  not found

 cfgmgr cfgmgr is running in phase 1
----------------------------------------------------------------------
invoking top level program -- "/etc/methods/defsys"
return code = 0
****************** stdout ***********
sys0

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sys0'
invoking /usr/lib/methods/cfgsys_chrp -1 -l sys0
return code = 0
****************** stdout ***********
pci0
:devices.chrp.IBM.TB3MX:devices.chrp.IBM.TB3MX

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'pci0'
invoking /usr/lib/methods/cfgbus_pcic -1 -l pci0
return code = 0
****************** stdout ***********
sa0,sa1,siokma0,fda0,scsi0,ent0

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sa0'
invoking /usr/lib/methods/cfgasync_rspc -1 -l sa0
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgasync_rspc:  not found

----------------------------------------------------------------------
attempting to configure device 'sa1'
invoking /usr/lib/methods/cfgasync_rspc -1 -l sa1
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgasync_rspc:  not found

----------------------------------------------------------------------
attempting to configure device 'siokma0'
invoking /usr/lib/methods/cfgkma_chrp -1 -l siokma0
return code = 127
****************** no stdout ***********
****************** stderr ***********
```

```
sh: /usr/lib/methods/cfgkma_chrp:  not found

-----------------------------------------------------------------------
attempting to configure device 'fda0'
invoking /usr/lib/methods/cfgfda_chrp -1 -l fda0
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgfda_chrp:  not found

-----------------------------------------------------------------------
attempting to configure device 'scsi0'
invoking /usr/lib/methods/cfgncr_scsi -1 -l scsi0
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgncr_scsi:  not found

-----------------------------------------------------------------------
attempting to configure device 'ent0'
invoking /usr/lib/methods/cfgkent -1 -l ent0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
-----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deflvm"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/deflvm:  not found

-----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defssar"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/defssar:  not found

-----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deftmssar"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/deftmssar:  not found

+ export NSORDER=local
+ bootinfo -b
+ [ ent0 = atm0 ]
+ native_netboot_cfg
+ bootinfo -c
+ set -- 192.168.4.10 192.168.4.130 192.168.4.130 FF FF 0
/tftpboot/sp4n10.msc.itso.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.
0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0. 0
+ CLIENT_IPADDR=192.168.4.10
+ BOOT_SERV_IP=192.168.4.130
+ BOOT_GATE_IP=192.168.4.130
+ E802=0
+ BOOTFILE=/tftpboot/sp4n10.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
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
+ [ -n 255.255.255.0 ]
+ SUBMASK=netmask 255.255.255.0
+ [ 192.168.4.130 = 0 -o 192.168.4.130 = 0.0.0.0 -o 192.168.4.130 = 192.168.4.130
]
+ unset BOOT_GATE_IP
+ + bootinfo -b
PHY_BOOT_DEV=ent0
+ pdev_to_ldev
+ /usr/lib/methods/showled 0x606
 showled + ifconfig lo0 inet 127.0.0.1 up
+ ifconfig en0 inet 192.168.4.10 up netmask 255.255.255.0
+ [ 0 -ne 0 ]
+ [  ]
+ [ -n  ]
+ CLIENT_INFO_FILE=/tftpboot/sp4n10.msc.itso.ibm.com.info
+ [ 0 -ne 0 ]
+ /usr/lib/methods/showled 0x608
 showled + tftp -go /SPOT/niminfo 192.168.4.130
/tftpboot/sp4n10.msc.itso.ibm.com.info image
Received 1328 Bytes in 0.0 Seconds
+ [ -s /SPOT/niminfo ]
+ . /SPOT/niminfo
+ export NIM_NAME=sp4n10
+ export NIM_HOSTNAME=sp4n10.msc.itso.ibm.com
+ export NIM_CONFIGURATION=standalone
+ export NIM_MASTER_HOSTNAME=sp4en0.msc.itso.ibm.com
+ export NIM_MASTER_PORT=1058
+ export NIM_REGISTRATION_PORT=1059
+ 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=sp4n10.msc.itso.ibm.com
+ export
SPOT=sp4en0.msc.itso.ibm.com:/spdata/sys1/install/default/spot/spot_default/usr
+ export NIM_BOSINST_DATA=/NIM_BOSINST_DATA
+ export NIM_CUSTOM=/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script -a
location=sp4en0.msc.itso.ibm.com:/export/nim/scripts/sp4n10.script
+ export NIM_BOS_IMAGE=/NIM_BOS_IMAGE
+ export NIM_BOS_FORMAT=mksysb
+ export NIM_HOSTS= 192.168.4.10:sp4n10.msc.itso.ibm.com
192.168.4.130:sp4en0.msc.itso.ibm.com
+ export NIM_MOUNTS=
sp4en0.msc.itso.ibm.com:/spdata/sys1/install/default/lppsource:/SPOT/usr/sys/inst
.images:dir
sp4en0.msc.itso.ibm.com:/spdata/sys1/install/pssp/bosinst_data:/NIM_BOSINST_DATA:
file
sp4en0.msc.itso.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.43K_9833B:/NIM_BO
S_IMAGE:file
+ [ -n  192.168.4.10:sp4n10.msc.itso.ibm.com
192.168.4.130:sp4en0.msc.itso.ibm.com  ]
+ OIFS=

+ IFS=:
+ set -- 192.168.4.10 sp4n10.msc.itso.ibm.com
+ IFS=

+ echo 192.168.4.10 sp4n10.msc.itso.ibm.com
+ 1>> /etc/hosts
+ OIFS=

+ IFS=:
+ set -- 192.168.4.130 sp4en0.msc.itso.ibm.com
+ IFS=
+ echo 192.168.4.130 sp4en0.msc.itso.ibm.com
+ 1>> /etc/hosts
+ [ -n  ]
+ 1> /etc/filesystems
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=LED 610: mount -r
sp4en0.msc.itso.ibm.com:/spdata/sys1/install/default/spot/spot_default/usr
/SPOT/usr
+ /usr/lib/methods/showled 0x610
 showled + mount -r
sp4en0.msc.itso.ibm.com:/spdata/sys1/install/default/spot/spot_default/usr
/SPOT/usr
+ [[ 0 -ne 0 ]]
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=
+ /usr/lib/methods/showled 0x612
 showled + cp /SPOT/usr/lib/boot/network/rc.bos_inst /etc
+ RC_CONFIG=/etc/rc.bos_inst
+ . /etc/rc.bos_inst
+ set -x
+ /SPOT/usr/sbin/nimclient -S booting
+ mount_from_list
+ [ -n
sp4en0.msc.itso.ibm.com:/spdata/sys1/install/default/lppsource:/SPOT/usr/sys/inst
.images:dir
sp4en0.msc.itso.ibm.com:/spdata/sys1/install/pssp/bosinst_data:/NIM_BOSINST_DATA:
file
sp4en0.msc.itso.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.43K_9833B:/NIM_BO
S_IMAGE:file  ]
+ /usr/lib/methods/showled 0x610
 showled + OIFS=

+ IFS=:
+ set -- sp4en0.msc.itso.ibm.com /spdata/sys1/install/default/lppsource
/SPOT/usr/sys/inst.images dir
+ IFS=

+ [ ! -d /SPOT/usr/sys/inst.images ]
+ /SPOT/usr/sbin/nimclient -o change -a force=yes -a ignore_lock=yes -a info=LED
610: mount sp4en0.msc.itso.ibm.com:/spdata/sys1/install/default/lppsource
/SPOT/usr/sys/inst.images
+ mount sp4en0.msc.itso.ibm.com:/spdata/sys1/install/default/lppsource
/SPOT/usr/sys/inst.images
+ [[ 0 -ne 0 ]]
+ IFS=:
+ set -- sp4en0.msc.itso.ibm.com /spdata/sys1/install/pssp/bosinst_data
/NIM_BOSINST_DATA file
+ IFS=

+ [ ! -d /NIM_BOSINST_DATA ]
+ /SPOT/usr/bin/mkdir -p /NIM_BOSINST_DATA
+ /SPOT/usr/sbin/nimclient -o change -a force=yes -a ignore_lock=yes -a info=LED
610: mount sp4en0.msc.itso.ibm.com:/spdata/sys1/install/pssp/bosinst_data
/NIM_BOSINST_DATA
+ mount sp4en0.msc.itso.ibm.com:/spdata/sys1/install/pssp/bosinst_data
/NIM_BOSINST_DATA
+ [[ 0 -ne 0 ]]
+ IFS=:
+ set -- sp4en0.msc.itso.ibm.com
/spdata/sys1/install/images/bos.obj.ssp.43K_9833B /NIM_BOS_IMAGE file
+ IFS=

+ [ ! -d /NIM_BOS_IMAGE ]
+ /SPOT/usr/bin/mkdir -p /NIM_BOS_IMAGE
+ /SPOT/usr/sbin/nimclient -o change -a force=yes -a ignore_lock=yes -a info=LED
610: mount
sp4en0.msc.itso.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.43K_9833B
/NIM_BOS_IMAGE
+ mount sp4en0.msc.itso.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.43K_9833B
/NIM_BOS_IMAGE
+ [[ 0 -ne 0 ]]
+ /SPOT/usr/sbin/nimclient -o change -a force=yes -a ignore_lock=yes -a info=
+ /usr/lib/methods/showled 0x622
 showled ln: /usr/lib is a directory.  (cannot unlink)
ln: /usr/lib/methods is a directory.  (cannot unlink)
+ /SPOT/usr/lib/boot/network/link_methods
+ strload -f /dev/null
+ cfgmgr -f -v
 cfgmgr  cfgmgr  cfgmgr  cfgsys_chrp  cfgmgr  cfgmgr  cfgbus_pcic  cfgbus_isac
cfgbus_pcic  cfgmgr  cfgmgr  cfgkent  cfgmgr  cfgmgr  cfgasync_rspc  cfgmgr
cfgmgr  cfgasync_rspc  cfgmgr  cfgmgr  cfgkma_chrp  cfgmgr  cfgmgr  cfgfda_chrp
cfgmgr  cfgmgr  cfgncr_scsi  cfgmgr  cfgmgr  cfgkm_chrp  cfgmgr  cfgmgr
cfgkm_chrp  cfgmgr  cfgmgr  cfgscdisk  cfgmgr  cfgmgr  cfgscdisk  cfgmgr  cfgmgr
cfgmgr  cfgmgr  cfglvdd  cfgmgr  cfgmgr  cfgmgr cfgmgr is running in phase 1
----------------------------------------------------------------------
invoking top level program -- "/etc/methods/defsys"
return code = 0
****************** stdout ***********
sys0

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sys0'
invoking /usr/lib/methods/cfgsys_chrp -1 -l sys0
return code = 0
****************** stdout ***********
pci0
:devices.chrp.IBM.TB3MX:devices.chrp.IBM.TB3MX

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'pci0'
invoking /usr/lib/methods/cfgbus_pcic -1 -l pci0
return code = 0
****************** stdout ***********
ent0,sa0,sa1,siokma0,fda0,scsi0

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'ent0'
invoking /usr/lib/methods/cfgkent -1 -l ent0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sa0'
invoking /usr/lib/methods/cfgasync_rspc -1 -l sa0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sa1'
invoking /usr/lib/methods/cfgasync_rspc -1 -l sa1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
```

```
attempting to configure device 'siokma0'
invoking /usr/lib/methods/cfgkma_chrp -1 -l siokma0
return code = 0
****************** stdout ***********
sioka0 sioma0
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'fda0'
invoking /usr/lib/methods/cfgfda_chrp -1 -l fda0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'scsi0'
invoking /usr/lib/methods/cfgncr_scsi -1 -l scsi0
return code = 0
****************** stdout ***********
hdisk0 hdisk1
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sioka0'
invoking /usr/lib/methods/cfgkm_chrp -1 -l sioka0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sioma0'
invoking /usr/lib/methods/cfgkm_chrp -1 -l sioma0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'hdisk0'
invoking /etc/methods/cfgscdisk -1 -l hdisk0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'hdisk1'
invoking /etc/methods/cfgscdisk -1 -l hdisk1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deflvm"
return code = 0
****************** stdout ***********
lvdd

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'lvdd'
invoking /usr/lib/methods/cfglvdd -1 -l lvdd
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defssar"
return code = 0
****************** no cfgmgr  cfgmgr Method error (/usr/lib/methods/deftmssar ):
0514-068 Cause not known.
sh: /usr/lib/methods/deftmssar:  not found
```

```
 cfgmgr  stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deftmssar"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/deftmssar:  not found

+ /usr/lib/methods/showled 0x622
 showled + /SPOT/usr/lib/boot/network/link_methods
+ export DEV_PKGNAME=ALL
+ cfgmgr -s -v
 cfgmgr  cfgmgr  cfgmgr  cfgsys_chrp  cfgmgr  cfgmgr  cfgbus_pcic  cfgbus_isac
cfgbus_pcic  cfgmgr  cfgmgr  cfgasync_rspc  cfgmgr  cfgmgr  cfgasync_rspc  cfgmgr
cfgmgr  cfgkma_chrp  cfgmgr  cfgmgr  cfgfda_chrp  cfgmgr  cfgmgr  cfgncr_scsi
cfgmgr  cfgmgr  cfgkent  cfgmgr  cfgmgr  cfgkm_chrp  cfgmgr  cfgmgr  cfgkm_chrp
cfgmgr  cfgmgr  cfgscdisk  cfgmgr  cfgmgr  cfgscdisk  cfgmgr cfgmgr is running in
phase 2
----------------------------------------------------------------------
invoking top level program -- "/etc/methods/defsys"
return code = 0
****************** stdout ***********
:devices.chrp.base sys0

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sys0'
invoking /usr/lib/methods/cfgsys_chrp -2 -l sys0
return code = 0
****************** stdout ***********
:devices.common.IBM.pmmd_chrp
:devices.chrp.pci
pci0
:devices.chrp.IBM.TB3MX:devices.chrp.IBM.TB3MX

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'pci0'
invoking /usr/lib/methods/cfgbus_pcic -2 -l pci0
return code = 0
****************** stdout ***********
:devices.pci.isa
:devices.isa_sio.serial:devices.isa_sio.pnpPNP.501
:devices.isa_sio.serial:devices.isa_sio.pnpPNP.501
:devices.isa_sio.8042:devices.isa_sio.chrp.8042
:devices.isa_sio.fdc:devices.isa_sio.pnpPNP.700
:devices.pci.scsi:devices.pci.00100300:devices.pci.NCR.53C825:devices.pci.SYM.53C
825:devices.pci.NCR.8251S:devices.pci.SYM.8251S:devices.pci.AAPL.NCR8250S
:devices.pci.ethernet:devices.pci.22100020
sa0,sa1,siokma0,fda0,scsi0,ent0

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sa0'
invoking /usr/lib/methods/cfgasync_rspc -2 -l sa0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sa1'
invoking /usr/lib/methods/cfgasync_rspc -2 -l sa1
return code = 0
```

```
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'siokma0'
invoking /usr/lib/methods/cfgkma_chrp -2 -l siokma0
return code = 0
****************** stdout ***********
sioka0 sioma0
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'fda0'
invoking /usr/lib/methods/cfgfda_chrp -2 -l fda0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'scsi0'
invoking /usr/lib/methods/cfgncr_scsi -2 -l scsi0
return code = 0
****************** stdout ***********
:devices.scsi.disk :devices.scsi.disk hdisk0 hdisk1 :devices.scsi.tape
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'ent0'
invoking /usr/lib/methods/cfgkent -2 -l ent0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sioka0'
invoking /usr/lib/methods/cfgkm_chrp -2 -l sioka0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sioma0'
invoking /usr/lib/methods/cfgkm_chrp -2 -l sioma0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'hdisk0'
invoking /etc/methods/cfgscdisk -2 -l hdisk0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'hdisk1'
invoking /etc/methods/cfgscdisk -2 -l hdisk1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib cfgmgr  cfgmgr  cfgmgr  cfgmgr Method
error (/usr/lib/methods/defops ):
0514-068 Cause not known.
sh: /usr/lib/methods/defops:  not found

 cfgmgr  cfgmgr Method error (/usr/lib/methods/definet > /dev/null
2>&1;opt=`/usr/sbin/lsattr -E -l inet0 -a bootup_option -F value`
if [ $opt = "no" ];then nf=/etc/rc.net
else nf=/etc/rc.bsdnet
fi;$nf -2;x=$?;test $x -ne 0&&echo $nf failed. Check for invalid commands >&2;exit
$x ):
0514-068 Cause not known.
lsattr: 0514-519 The following device was not found in the customized
device configuration database:
inet0
sh[2]: test: argument expected
sh[4]: /etc/rc.bsdnet:  not found
/etc/rc.bsdnet failed. Check for invalid commands

 cfgmgr  cfgmgr Method error (/etc/methods/ptynode ):
0514-068 Cause not known.
sh: /etc/methods/ptynode:  not found

 cfgmgr  cfgmgr  cfgmgr  cfgmgr Method error (/etc/methods/startrcm ):
0514-068 Cause not known.
sh: /etc/methods/startrcm:  not found

 cfgmgr  cfgmgr  cfgmgr  cfgtty  cfgmgr /usr/sbin/mkitab: could not open
/etc/inittab

 cfgmgr  cfgmgr Method error (/etc/methods/startsgio ):
0514-068 Cause not known.
sh: /etc/methods/startsgio:  not found

 cfgmgr  cfgmgr Method error (/usr/lib/methods/fdarcfgrule ):
0514-068 Cause not known.
sh: /usr/lib/methods/fdarcfgrule:  not found

 cfgmgr  cfgmgr Method error (/etc/methods/darcfgrule ):
0514-068 Cause not known.
sh: /etc/methods/darcfgrule:  not found

 cfgmgr  cfgmgr  cfgmgr  cfgmgr Method error (/usr/lib/methods/deftmssar ):
0514-068 Cause not known.
sh: /usr/lib/methods/deftmssar:  not found

 cfgmgr  cfgmgr Method error (/usr/lib/methods/cfgfan ):
0514-068 Cause not known.
sh: /usr/lib/methods/cfgfan:  not found

 cfgmgr  cfgmgr Method error (/usr/lib/methods/cfgvcons -l ttyvcons0 ):
0514-068 Cause not known.
sh: /usr/lib/methods/cfgvcons:  not found

/methods/deflvm"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defops"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/defops:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/definet > /dev/null
2>&1;opt=`/usr/sbin/lsattr -E -l inet0 -a bootup_option -F value`
if [ $opt = "no" ];then nf=/etc/rc.net
else nf=/etc/rc.bsdnet
fi;$nf -2;x=$?;test $x -ne 0&&echo $nf failed. Check for invalid commands >&2;exit
$x"
```

```
return code = 127
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/etc/methods/ptynode"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/ptynode:  not found

----------------------------------------------------------------------
invoking top level program -- "/etc/methods/startlft"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/etc/methods/startrcm"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/startrcm:  not found

----------------------------------------------------------------------
invoking top level program -- "/etc/methods/starttty"
return code = 0
****************** stdout ***********
tty0
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'tty0'
invoking /etc/methods/cfgtty -2 -l tty0
return code = 0
****************** no stdout ***********
****************** stderr ***********
/usr/sbin/mkitab: could not open /etc/inittab

----------------------------------------------------------------------
invoking top level program -- "/etc/methods/startsgio"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/startsgio:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/fdarcfgrule"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/fdarcfgrule:  not found

----------------------------------------------------------------------
invoking top level program -- "/etc/methods/darcfgrule"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/darcfgrule:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defssar"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
```

```
invoking top level program -- "/usr/lib/methods/deftmssar"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/deftmssar:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/cfgfan"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgfan:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/cfgvcons -l ttyvcons0"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgvcons:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -C -c driver -s sdl cfgmgr  cfgmgr
cfgmgr  cfgmgr Method error (/usr/lib/methods/defaio ):
0514-068 Cause not known.
sh: /usr/lib/methods/defaio:  not found

 cfgmgr  cfgmgr  cfgmgr c -t scie -F name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defaio"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/defaio:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -C -c port -s tsd -t tsp -F name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
+ /usr/lib/methods/showled 0x622
 showled + exit 0
+ [ 1 -ne 1 ]
+ PHASE=2
+ + bootinfo -p
PLATFORM=chrp
+ [ ! -x /usr/lib/boot/bin/bootinfo_chrp ]
+ [ 2 -eq 1 ]
+ + bootinfo -t
BOOTYPE=5
+ [ 0 -ne 0 ]
+ [ -z 5 ]
+ chramfs -t
/sbin/rc.boot[321]: chramfs:  not found
+ init -c unlink /usr/sbin/chramfs
+ 1> /dev/null
Could not load program init
Symbol setpcred in init is undefined
Symbol _FloatingReleaseLicense in init is undefined
Error was: Exec format error
+ unset NIM_DEBUG
+ export NIM_DEBUG=set -x
+ unset fd_invoker
+ set -x
+ export NSORDER=local
+ + bootinfo -b
PHY_BOOT_DEV=ent0
+ . /SPOT/niminfo
+ export NIM_NAME=sp4n10
+ export NIM_HOSTNAME=sp4n10.msc.itso.ibm.com
+ export NIM_CONFIGURATION=standalone
+ export NIM_MASTER_HOSTNAME=sp4en0.msc.itso.ibm.com
+ export NIM_MASTER_PORT=1058
+ export NIM_REGISTRATION_PORT=1059
+ 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=sp4n10.msc.itso.ibm.com
+ export
SPOT=sp4en0.msc.itso.ibm.com:/spdata/sys1/install/default/spot/spot_default/usr
+ export NIM_BOSINST_DATA=/NIM_BOSINST_DATA
+ export NIM_CUSTOM=/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script -a
location=sp4en0.msc.itso.ibm.com:/export/nim/scripts/sp4n10.script
+ export NIM_BOS_IMAGE=/NIM_BOS_IMAGE
+ export NIM_BOS_FORMAT=mksysb
+ export NIM_HOSTS= 192.168.4.10:sp4n10.msc.itso.ibm.com
192.168.4.130:sp4en0.msc.itso.ibm.com
+ export NIM_MOUNTS=
sp4en0.msc.itso.ibm.com:/spdata/sys1/install/default/lppsource:/SPOT/usr/sys/inst
.images:dir
sp4en0.msc.itso.ibm.com:/spdata/sys1/install/pssp/bosinst_data:/NIM_BOSINST_DATA:
file
sp4en0.msc.itso.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.43K_9833B:/NIM_BO
S_IMAGE:file
+ RC_CONFIG=/etc/rc.bos_inst
+ . /etc/rc.bos_inst
+ set -x
+ /SPOT/usr/sbin/no -o tcp_keepintvl=150
+ /SPOT/usr/sbin/no -o tcp_keepidle=1200
+ cp /usr/lpp/diagnostics/obj/CDiagDev /etc/objrepos/CDiagDev
+ cp /usr/lpp/diagnostics/obj/TMInput /etc/objrepos/TMInput
+ cp /usr/lpp/diagnostics/obj/MenuGoal /etc/objrepos/MenuGoal
+ cp /usr/lpp/diagnostics/obj/FRUB /etc/objrepos/FRUB
+ cp /usr/lpp/diagnostics/obj/FRUs /etc/objrepos/FRUs
+ cp /usr/lpp/diagnostics/obj/DAVars /etc/objrepos/DAVars
+ cp /usr/lpp/diagnostics/obj/CDiagAtt /etc/objrepos/CDiagAtt
+ cp /usr/lpp/diagnostics/obj/CDiagAtt.vc /etc/objrepos/CDiagAtt.vc
+ mkdir -p /etc/lpp/diagnostics/data
+ /usr/lib/methods/showled 0xfff
 showled + exec /usr/lpp/bosinst/bi_main
+ typeset +f
+ typeset -ft BI_Clock
+ typeset -ft BI_Copy_Files
+ typeset -ft BI_Error
+ typeset -ft Breakpt
+ typeset -ft Call_Merge_Methods
+ typeset -ft Change_Mounts
+ typeset -ft Change_Status
+ typeset -ft Check_DCE_Level
+ typeset -ft Check_Down_Level
+ typeset -ft Check_Other_Stanzas
+ typeset -ft Check_SNA_Level
+ typeset -ft Clean_Migrated_VPD
+ typeset -ft Copy_Customized
+ typeset -ft Create_Bosinst_Data
+ typeset -ft Display_Status_Screen
+ typeset -ft Extract_Diskette_Data
+ typeset -ft Fill_Target_Stanzas
+ typeset -ft Finish
+ typeset -ft Get_Data_Files
+ typeset -ft Get_Locale_Packages
+ typeset -ft Get_User_Input
+ typeset -ft Image_Data_Exists
+ typeset -ft Init_Target_Disks
+ typeset -ft Initialize
+ typeset -ft Initialize_Disk_Environment
+ typeset -ft Initialize_Log
+ typeset -ft Install_Updates
+ typeset -ft Log
+ typeset -ft Make_Sys_FS
+ typeset -ft Make_Sys_LV
+ typeset -ft Make_Sys_VG
+ typeset -ft Mount_Rootfs
+ typeset -ft Other_Initialization
+ typeset -ft Post_Install
+ typeset -ft Prepare_Target_Disks
+ typeset -ft Restore_Bosinst
+ typeset -ft Restore_System
+ typeset -ft Run_Customization
+ typeset -ft Save_Devs
+ typeset -ft Set_Console
+ typeset -ft Set_PP_Size
+ typeset -ft Set_Primary_Locale
+ typeset -ft Shrink_FS
+ typeset -ft Shrink_It
+ typeset -ft Swapon_hd6
+ typeset -ft Turbo_Restore
+ typeset -ft Update_Status
+ typeset -ft Verify_Image_Parameters
+ typeset -ft Zero_Screen
+ typeset -ft free_dump_lv
+ typeset -ft loopled
+ [ ! -s ./bi_main.debug ]
+ restbyname -xqf - ./bi_main.debug ./startup
+ 1> /dev/null 2>& 1 + [ -s ./bi_main.debug ]
+ a=none
+ [ none = bi_main.debug ]
+ bootinfo -p
+ export PLATFORM=chrp
+ PLAT_HOOK=
+ [ -x /SPOT/platform/chrp/plat_get_debug ]
+ [ -x /SPOT/usr/lpp/bos/inst_root/platform/chrp/plat_get_debug ]
+ [ -x /platform/chrp/plat_get_debug ]
+ [ -n  ]
+ trap BI_Error "BOS Install" 29 2 INT
+ cd /
+ Initialize
+ trap BI_Error "BOS Install" 29 2 INT
+ /usr/lib/methods/showled 0xA46
 showled + umask 002
+ BOSINSTDATA=/bosinst.data
+ BUNDLE_INCR=40
+ CDROM=3
+ CURLEVEL=4.3
+ DEV_INCR=35
+ IMAGEDATA=/image.data
+ INST_INCR=75
+ INSTTASK=2
+ MEG=1048576
+ MKSYSB_INCR=75
+ MROOT=/mnt
+ NETWORK=5
+ NOM_INCR=2
+ NOT=!
+ NOPVID=0000000000000000
+ NOVG=0
+ NOVGID=0000000000000000
+ ODMDIR=/etc/objrepos
+ OTHERVG=2
+ PRE_LIST=/etc/preserve.list
+ REST_INCR=5
+ ROOTVG=1
+ PERCENT=0
+ TASK_INCR=1
+ TIME=0
+ TURBO_INCR=22
+ FATAL=
+ TAPE=4
+ BAR=
+ TAPEBLKSZ=/tapeblksz
+ TAPEDEVICE=/tapedevice
+ TARGETVGS=/tmp/targetvgs
+ BOS_FORMAT=nonturbo
+ export BOSINSTDATA CDROM IMAGEDATA INSTTASK
+ export MEG MROOT NETWORK NOPVID NOT NOVG NOVGID
+ export OTHERVG ROOTVG TAPE TAPEBLKSZ TARGETVGS ODMDIR TAPEDEVICE
+ export TIME PERCENT BAR FATAL PRE_LIST CURLEVEL BOS_FORMAT
+ export DEBUG=
+ export SYSCFG_PHASE=BOSINST
+ + bootinfo -r
REALMEM=524288
+ export REALMEM
+ MIN_PGSP=32
+ export MIN_PGSP
+ [ ! -s /tapedevice ]
+ set -x
+ unset BOOTDEV BOOTTYPE
+ [ -s /tapedevice ]
+ + /usr/sbin/bootinfo -t
BOOTTYPE=5
+ + /usr/sbin/bootinfo -b
BOOTDEV=ent0
+ export BOSMENU_LANG=
+ ORIG_BOOTTYPE=5
+ ORIG_BOOTDEV=ent0
+ export ORIG_BOOTTYPE ORIG_BOOTDEV
+ export BOOTTYPE BOOTDEV
+ + bootinfo -z
+ 2> /dev/null
BOOT_Z=1
+ export BOOT_Z
+ [ 5 -eq 5 ]
+ [  -ne 1 ]
+ nimclient -R success
+ /usr/lib/methods/showled 0xA40
 showled + [ 5 -eq 4 ]
+ Get_Data_Files
+ trap BI_Error "BOS Install" 29 2 INT
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=extract_data_files
+ [  -eq 1 ]
+ [ /NIM_BOS_IMAGE -a mksysb ]
+ restbyname -xqdSf - ./bosinst.data ./image.data ./tmp/vgdata
+ dd if=/NIM_BOS_IMAGE bs=1k count=128
+ 1> /dev/null 2>& 1
128+0 records in
128+0 records out
+ [ -s /NIM_BOSINST_DATA ]
+ cp /NIM_BOSINST_DATA /bosinst.data
+ 1> /dev/null 2>& 1
+ [ -s  ]
+ [ -z  ]
+ cp /SPOT/usr/lpp/bos/inst_root/etc/preserve.list /etc/preserve.list
+ 1> /dev/null 2>& 1
+ return 0
+ [ 5 -eq 5 ]
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=query_disks
+ rm -f /tmp/Get_RVG_Disks.pn
+ bootinfo -r
+ [ 524288 -le 8192 ]
+ echo 3002
+ /usr/lpp/bosinst/Get_RVG_Disks
+ 1> /tmp/Get_RVG_Disks.pn
+ [ -s /bosinst.data -a 5 -ne 5 ]
+ unset IMAGE_ON_DISKETTE
+ unset PRESERVE_ON_DISKETTE
+ PLAT_HOOK=
+ [ -x /SPOT/platform/chrp/plat_get_data ]
+ [ -x /SPOT/usr/lpp/bos/inst_root/platform/chrp/plat_get_data ]
+ [ -x /platform/chrp/plat_get_data ]
+ [ -n  ]
+ [ -s /platform/chrp/bosinst.data ]
+ [ -s /platform/chrp/image.data ]
+ [ -s /platform/chrp/preserve.list ]
+ /usr/lib/methods/showled 0xA42
 showled + Extract_Diskette_Data
+ trap BI_Error "BOS Install" 29 2 INT
+ [ 5 -eq 5 ]
+ nimclient -o change -a force=yes -a ignore_lock=yes -a
info=extract_diskette_data
+ cd /
+ rm -f ./signature
+ restbyname -xqSf /dev/rfd0 ./signature
+ 1> /dev/null 2>& 1
+ [ 1 -eq 0 -a -s ./signature ]
+ [ -f /usr/bin/dosread ]
+ return 0
+ [ ! -s /bosinst.data ]
+ [ -n  ]
+ Image_Data_Exists
+ trap BI_Error "BOS Install" 29 2 INT
+ [ ! -s /image.data ]
+ return 0
+ /usr/lpp/bosinst/datadaemon
+ sleep 1
+ [ -n  ]
+ Set_Console
+ trap BI_Error "BOS Install" 29 2 INT
+ [ 5 -eq 5 ]
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=setting_console
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f CONSOLE
C=/dev/tty0
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f PROMPT
```

```
P=no
+ [ /dev/tty0 -a no = no ]
+ [[ /dev/tty0 = none ]]
+ [[ 5 -eq 5 ]]
+ [[ /dev/tty0 != /dev/?* ]]
+ chcons /dev/tty0
chcons: console assigned to: /dev/tty0, effective on next system boot
+ /usr/lib/methods/cfgcon
 cfgcon + unset C P
+ ln -f /dev/console /dev/tty
+ 1> /dev/null 2>& 1
+ + /usr/lpp/bosinst/bi_io -c
+ 0< /dev/console
CLEAR=
+ [ -z  ]
+ CLEAR=

+ export CLEAR
+ return 0
+ [  ]
+ Zero_Screen
+ trap BI_Error "BOS Install" 29 2 INT
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f PROMPT
PT=no
+ [ no = no ]
+ echo

        \c

         + echo 000\c
000+ echo 3894
+ 1> /tmp/Animate.pn
+ return 0
+ [ 5 -eq 5 ]
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=initialization
+ Animate \|/-
\|+ Init_Target_Disks
+ trap BI_E showled rror "BOS Install" 29 2 INT
+ /usr/lib/methods/showled 0xA44
+ [ -f /tmp/Get_RVG_Disks.pn ]
+ cat /tmp/Get_RVG_Disks.pn
+ wait 3002
/-\|/-\|+ [[ ! -s /tmp/targetvgs ]]
+ / showled usr/lib/methods/showled 0xA46
+ ANY_DEFINED=
+ + lsdev -Ccdisk -S Defined -Fname
ANY_DEFINED=
+ [  ]
+ unset RVG_EXISTS
+ exec
+ 3< /tmp/targetvgs
+ read -u3 VGID INVG LEVEL NAME LOC SIZE BOOTABLE PVID CONNECTION
+ [ 1 -eq 1 ]
+ RVG_EXISTS=TRUE
+ break
+ exec
+ + + /../SPOT/usr/lpp/bosinst/bidata -i -g image_data -f PRODUCT_TAPE
PT=no
+ [ no = no ]
+ /../SPOT/usr/lpp/bosinst/bidata -b -m control_flow -f INSTALL_METHOD -e
overwrite
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f INSTALL_METHOD
INSTALL_METHOD=overwrite
```

```
+ [ mksysb = spot -a overwrite = migrate ]
+ [ TRUE ]
+ [ -n  ]
+ /+ /../SPOT/usr/lpp/bosinst/bidata -b -g target_disk_data -f PVID
PV=
+ + echo
PV=
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g target_disk_data -f CONNECTION
PCONN=
+ + echo
PCONN=
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g target_disk_data -f LOCATION
LOC=
+ + echo
LOC=
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g target_disk_data -f SIZE_MB
SZ=
+ + echo
SZ=
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g target_disk_data -f HDISKNAME
HD=hdisk0
+ + echo hdisk0
HD=hdisk0
+ [ -z  -a -z  -a -z hdisk0 -a -z  -a -z  ]
+ unset MIGRATABLE
+ exec
+ 3< /tmp/targetvgs
+ read -u3 VGID INVG LEVEL NAME LOC SIZE BOOTABLE PVID CONNECTION
+ [[ 4.3 > 3.2 ]]
+ [[ 4.3 < 4.3 ]]
+ read -u3 VGID INVG LEVEL NAME LOC SIZE BOOTABLE PVID CONNECTION
+ [[ 4.3 > 3.2 ]]
+ [[ 4.3 < 4.3 ]]
+ read -u3 VGID INVG LEVEL NAME LOC SIZE BOOTABLE PVID CONNECTION
+ exec
+ + + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f INSTALL_METHOD
IM=overwrite
+ [  -a mksysb != spot ]
+ [ ! overwrite ]
+ [ overwrite = migrate ]
+ return 0
+ Other_Initialization
+ trap BI_Error "BOS Install" 29 2 INT
+ ln -sf /usr/lib/objrepos/CC /usr/lib/objrepos/CC.vc /usr/lib/objrepos/FONT
/usr/lib/objrepos/FONT.vc /usr/lib/objrepos/KEYBOARD
/usr/lib/objrepos/KEYBOARD.vc /usr/lib/objrepos/MESSAGES
/usr/lib/objrepos/MESSAGES.vc /etc/objrepos/
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f PROMPT
P=no
+ [ ! no ]
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f INSTALL_X_IF_ADAPTER
X=no
+ [ ! no ]
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f RUN_STARTUP
X=no
+ [ ! no ]
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f RM_INST_ROOTS
X=no
+ [ ! no ]
+ [ ! -f /usr/sbin/umount ]
+ return 0
+ [ -z  ]
+ [ 5 -eq 5 ]
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=verifying_data_files
-+ Verify_Image_Parameters
+ trap BI_Error "BOS Install" 29 2 INT
+ [ -d /SPOT/usr/sys/inst.images ]
+ \+ /../SPOT/usr/lpp/bosinst/bidata -i -g image_data -f OSLEVEL
OSLEVEL=4.3.2.0
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f PROMPT
PROMPT=no
+ [ -n 4.3.2.0 -a no = yes ]
+ [ -f /tapedevice ]
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g ils_data -f LANG
ML=en_US
+ [ -n en_US ]
+ + odmget -qlocale=en_US AND bosinst_translated=y MESSAGES
+ 2> /dev/null
IL_OK=
MESSAGES:
        locale = "en_US"
        text_string = "English (United States)"
        text_string_id = 106
        codeset = "ISO8859-1"
        package = "bos.loc.iso.en_US bos.msg.en_US.rte
bos.msg.en_US.net.tcp.client"
        variables = "LC_MESSAGES=en_US"
        bosinst_translated = "y"
        bosinst_menu = "y"
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g locale -f BOSINST_LANG
BL=en_US
+ [ -n
MESSAGES:
        locale = "en_US"
        text_string = "English (United States)"
        text_string_id = 106
        codeset = "ISO8859-1"
        package = "bos.loc.iso.en_US bos.msg.en_US.rte
bos.msg.en_US.net.tcp.client"
        variables = "LC_MESSAGES=en_US"
        bosinst_translated = "y"
        bosinst_menu = "y" -a -z en_US ]
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g vg_data -f VGNAME
RVG_STANZA_EXISTS=rootvg
+ [ rootvg != rootvg ]
+ ANY_PAGELVS=
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LOGICAL_VOLUME -c TYPE -v
paging
ANY_PAGELVS=hd6
+ ALL_LVS=TRUE
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LOGICAL_VOLUME -c
LOGICAL_VOLUME -v hd2
THERE=hd2
+ [ ! hd2 ]
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LOGICAL_VOLUME -c
LOGICAL_VOLUME -v hd4
THERE=hd4
+ [ ! hd4 ]
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LOGICAL_VOLUME -c
LOGICAL_VOLUME -v hd5
THERE=hd5
+ [ ! hd5 ]
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LOGICAL_VOLUME -c
LOGICAL_VOLUME -v hd8
THERE=hd8
+ [ ! hd8 ]
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LOGICAL_VOLUME -c
LOGICAL_VOLUME -v hd9var
THERE=hd9var
+ [ ! hd9var ]
+ [ ! hd6 -o ! TRUE ]
+ ALL_SYS_FSS=TRUE
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g fs_data -f FS_NAME -c FS_LV -v /dev/hd2
THERE=/usr
+ + echo /usr
THERE=/usr
+ [ /usr != /usr ]
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g fs_data -f FS_NAME -c FS_LV -v /dev/hd4
THERE=/
+ + echo /
THERE=/
+ [ / != / ]
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g fs_data -f FS_NAME -c FS_LV -v
/dev/hd9var
THERE=/var
+ + echo /var
THERE=/var
+ [ /var != /var ]
+ ALL_FSS_HAVE_LVS=TRUE
+ /../SPOT/usr/lpp/bosinst/bidata -i -g fs_data -f FS_LV
+ + echo /dev/hd4
+ cut -d/ -f3
lv=hd4
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LOGICAL_VOLUME -c
LOGICAL_VOLUME -v hd4
THERE=hd4
+ [ ! hd4 ]
+ + echo /dev/hd1
+ cut -d/ -f3
lv=hd1
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LOGICAL_VOLUME -c
LOGICAL_VOLUME -v hd1
THERE=hd1
+ [ ! hd1 ]
+ + echo /dev/hd3
+ cut -d/ -f3
lv=hd3
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LOGICAL_VOLUME -c
LOGICAL_VOLUME -v hd3
THERE=hd3
+ [ ! hd3 ]
+ + echo /dev/hd2
+ cut -d/ -f3
lv=hd2
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LOGICAL_VOLUME -c
LOGICAL_VOLUME -v hd2
THERE=hd2
+ [ ! hd2 ]
+ + echo /dev/hd9var
+ cut -d/ -f3
lv=hd9var
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LOGICAL_VOLUME -c
LOGICAL_VOLUME -v hd9var
THERE=hd9var
+ [ ! hd9var ]
+ [ ! TRUE -o ! TRUE ]
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g image_data -f PRODUCT_TAPE
PT=no
+ [ no = no ]
+ /../SPOT/usr/lpp/bosinst/bidata -b -m control_flow -f INSTALL_METHOD -e
overwrite
+ [ -n mksysb -a mksysb = spot ]
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g logical_volume_policy -f SHRINK
SHR=no
+ [ no = yes ]
+ TOTAL_LV_SIZE_MB=0
+ /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LOGICAL_VOLUME -c VGNAME -v
rootvg
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LPs -c LOGICAL_VOLUME -v hd5
num_pps=1
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f PP_SIZE -c LOGICAL_VOLUME -v
hd5
pp_size=8
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f COPIES -c LOGICAL_VOLUME -v
hd5
copies=1
+ ((  TOTAL_LV_SIZE_MB = TOTAL_LV_SIZE_MB + (num_pps * pp_size * copies)  ))
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LPs -c LOGICAL_VOLUME -v hd6
num_pps=64
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f PP_SIZE -c LOGICAL_VOLUME -v
hd6
pp_size=8
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f COPIES -c LOGICAL_VOLUME -v
hd6
|copies=1
+ ((  TOTAL_LV_SIZE_MB = TOTAL_LV_SIZE_MB + (num_pps * pp_size * copies)  ))
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LPs -c LOGICAL_VOLUME -v hd8
num_pps=1
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f PP_SIZE -c LOGICAL_VOLUME -v
hd8
pp_size=8
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f COPIES -c LOGICAL_VOLUME -v
hd8
copies=1
+ ((  TOTAL_LV_SIZE_MB = TOTAL_LV_SIZE_MB + (num_pps * pp_size * copies)  ))
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LPs -c LOGICAL_VOLUME -v hd4
num_pps=4
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f PP_SIZE -c LOGICAL_VOLUME -v
hd4
pp_size=8
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f COPIES -c LOGICAL_VOLUME -v
hd4
copies=1
+ ((  TOTAL_LV_SIZE_MB = TOTAL_LV_SIZE_MB + (num_pps * pp_size * copies)  ))
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LPs -c LOGICAL_VOLUME -v hd2
num_pps=41
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f PP_SIZE -c LOGICAL_VOLUME -v
hd2
pp_size=8
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f COPIES -c LOGICAL_VOLUME -v
hd2
copies=1
+ ((  TOTAL_LV_SIZE_MB = TOTAL_LV_SIZE_MB + (num_pps * pp_size * copies)  ))
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LPs -c LOGICAL_VOLUME -v
hd9var
num_pps=6
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f PP_SIZE -c LOGICAL_VOLUME -v
hd9var
pp_size=8
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f COPIES -c LOGICAL_VOLUME -v
hd9var
copies=1
```

```
+ ((  TOTAL_LV_SIZE_MB = TOTAL_LV_SIZE_MB + (num_pps * pp_size * copies)  ))
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LPs -c LOGICAL_VOLUME -v hd3
num_pps=6
+ /+ /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f PP_SIZE -c LOGICAL_VOLUME -v
hd3
pp_size=8
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f COPIES -c LOGICAL_VOLUME -v
hd3
copies=1
+ ((  TOTAL_LV_SIZE_MB = TOTAL_LV_SIZE_MB + (num_pps * pp_size * copies)  ))
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f LPs -c LOGICAL_VOLUME -v hd1
num_pps=1
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f PP_SIZE -c LOGICAL_VOLUME -v
hd1
pp_size=8
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g lv_data -f COPIES -c LOGICAL_VOLUME -v
hd1
copies=1
+ ((  TOTAL_LV_SIZE_MB = TOTAL_LV_SIZE_MB + (num_pps * pp_size * copies)  ))
+ TOTAL_DISK_SIZE=0
+ ONE_BOOTABLE=
+ lsdev -Ccdisk -S Avaliable -F name
+ bootinfo -B hdisk0
+ [ 1 -eq 1 ]
+ ONE_BOOTABLE=TRUE
+ + bootinfo -s hdisk0
SZ=4303
+ [ -n 4303 ]
+ ((  TOTAL_DISK_SIZE = TOTAL_DISK_SIZE + SZ  ))
+ bootinfo -B hdisk1
+ [ 1 -eq 1 ]
+ ONE_BOOTABLE=TRUE
+ + bootinfo -s hdisk1
SZ=4303
+ [ -n 4303 ]
+ ((  TOTAL_DISK_SIZE = TOTAL_DISK_SIZE + SZ  ))
+ [ ! TRUE ]
+ [ 992 -le 8606 ]
+ [  ]
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g source_disk_data -f LOCATION
ALL_MATCH=00-00-00-0,0
+ [ 00-00-00-0,0 ]
+ /../SPOT/usr/lpp/bosinst/bidata -i -g source_disk_data -f LOCATION
+ + bootinfo -o 00-00-00-0,0
NAME_LIST=
+ [  ]
+ ALL_MATCH=
+ [ -z  ]
+ break
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f
EXISTING_SYSTEM_OVERWRITE
ESO=yes
+ [ yes = any ]
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g source_disk_data -f LOCATION
ALL_CLEAR=00-00-00-0,0
+ [ 00-00-00-0,0 ]
+ /../SPOT/usr/lpp/bosinst/bidata -i -g source_disk_data -f LOCATION
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g source_disk_data -f HDISKNAME -c
LOCATION -v 00-00-00-0,0
DISK_LIST=hdisk0
+ read a VGSTATUS c d e f g h i
+ grep  hdisk0  /tmp/targetvgs
+ [ 1 -ne 0 ]
+ ALL_CLEAR=
+ break
+ [ -z  ]
+ break
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g logical_volume_policy -f EXACT_FIT
EXFT=no
+ [  -a  -a ( no = yes -o -n  ) ]
+ [ no = yes ]
+ return 0
+ Fill_Target_Stanzas
+ trap /usr/bin/ksh INT
+ 1> /tmp/disks
+ 0< /tmp/targetvgs
+ read VGID rvg lvl DISK rest
+ echo :hdisk0:
+ 1>> /tmp/disks
+ read VGID rvg lvl DISK rest
+ echo :hdisk1:
+ 1>> /tmp/disks
+ read VGID rvg lvl DISK rest
+ rm -f /tmp/tdd.add
+ read TDD_STANZA
+ /../SPOT/usr/lpp/bosinst/bidata -G
+ + echo ::hdisk0::
+ cut -d: -f1
LOC=
+ + echo ::hdisk0::
+ cut -d: -f2
-SIZE=
+ + echo ::hdisk0::
+ cut -d: -f3
NAME=hdisk0
+ + echo ::hdisk0::
+ cut -d: -f4
PVID=
+ + echo ::hdisk0::
+ cut -d: -f5
CONNECTION=
+ [[ -n  ]]
+ [[ -z  ]]
+ [[ -n  ]]
+ [[ -z  ]]
+ [[ -z  ]]
+ [[ -n  ]]
+ [[ -n  ]]
+ [ -n  ]
+ [ -n  ]
+ [ -n hdisk0 ]
+ + grep :hdisk0:
+ lsdev -Ccdisk -S Available -F :name:
IT_IS_THERE=:hdisk0:
+ [ :hdisk0: ]
+ + getlvodm -p hdisk0
+ 2> /dev/null
pvid=0000916000019871
+ pvid=0000916000019871
+ + lsdev -Cc disk -F parent//connwhere -l hdisk0
conn=scsi0//0,0
+ conn=scsi0//0,0
+ bootinfo -o hdisk0
+ bootinfo -s hdisk0
+ echo 10-60-00-0,0 4303 hdisk0 0000916000019871 scsi0//0,0
+ 1>> /tmp/tdd.add
+ grep -v :hdisk0: /tmp/disks
+ 1> /tmp/d
+ mv /tmp/d /tmp/disks
+ read TDD_STANZA
+ rm /tmp/disks
+ /../SPOT/usr/lpp/bosinst/bidata -D
+ [ -s /tmp/tdd.add ]
+ read LOC SIZE NAME PVID CONNECTION
+ cat /tmp/tdd.add
+ /../SPOT/usr/lpp/bosinst/bidata -a -l 10-60-00-0,0 -s 4303 -n hdisk0
+ /../SPOT/usr/lpp/bosinst/bidata -b -m target_disk_data -f PVID -e
0000916000019871 -c HDISKNAME -v hdisk0
+ /../SPOT/usr/lpp/bosinst/bidata -b -m target_disk_data -f CONNECTION -e
scsi0//0,0 -c HDISKNAME -v hdisk0
+ read LOC SIZE NAME PVID CONNECTION
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f INSTALL_METHOD
IM=overwrite
+ [ overwrite = preserve -o overwrite = migrate ]
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g target_disk_data -f HDISKNAME
DISKLIST=hdisk0
+ [ -n hdisk0 ]
+ BOOTDISK=
+ + bootinfo -B hdisk0
IS_BOOTABLE=1
+ [ 1 -eq 1 ]
+ BOOTDISK=hdisk0
+ break
+ [ -n hdisk0 ]
+ OIFS=

+ IFS=:
+ /../SPOT/usr/lpp/bosinst/bidata -G
+ grep :hdisk0
+ set -- 10-60-00-0,0 4303 hdisk0 0000916000019871 scsi0//0,0
+ IFS=

+ echo 10-60-00-0,0 4303 hdisk0 0000916000019871 scsi0//0,0
+ 1> /tmp/tdd.add
+ /../SPOT/usr/lpp/bosinst/bidata -d HDISKNAME -v hdisk0
+ OIFS=

+ IFS=:
+ 1>> /tmp/tdd.add
+ read LOC SIZE NAME PVID CONNECTION
+ /../SPOT/usr/lpp/bosinst/bidata -G
+ IFS=

+ /../SPOT/usr/lpp/bosinst/bidata -D
+ read LOC SIZE NAME PVID CONNECTION
+ cat /tmp/tdd.add
+ /../SPOT/usr/lpp/bosinst/bidata -a -l 10-60-00-0,0 -s 4303 -n hdisk0
\+ /../SPOT/usr/lpp/bosinst/bidata -b -m target_disk_data -f PVID -e
0000916000019871 -c HDISKNAME -v hdisk0
+ /../SPOT/usr/lpp/bosinst/bidata -b -m target_disk_data -f CONNECTION -e
scsi0//0,0 -c HDISKNAME -v hdisk0
+ read LOC SIZE NAME PVID CONNECTION
+ [ -z  ]
+ Check_Other_Stanzas
+ trap BI_Error "BOS Install" 29 2 INT
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f
EXISTING_SYSTEM_OVERWRITE
ESO=yes
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f PROMPT
```

```
P=no
+ [ no = yes ]
+ [ yes = no ]
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f INSTALL_METHOD
IM=overwrite
+ [ overwrite != migrate -a overwrite != preserve -a overwrite != overwrite ]
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f RUN_STARTUP
RS=no
+ [ no != yes -a no != no ]
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f INSTALL_X_IF_ADAPTER
IX=no
+ [ no != yes -a no != no -a no != all ]
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g image_data -f PRODUCT_TAPE
PT=no
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g locale -f BOSINST_LANG
BL=en_US
+ + odmget -qlocale=en_US AND bosinst_translated=y MESSAGES
+ 2> /dev/null
BL_OK=
MESSAGES:
        locale = "en_US"
        text_string = "English (United States)"
        text_string_id = 106
        codeset = "ISO8859-1"
        package = "bos.loc.iso.en_US bos.msg.en_US.rte
bos.msg.en_US.net.tcp.client"
        variables = "LC_MESSAGES=en_US"
        bosinst_translated = "y"
        bosinst_menu = "y"
+ [ !
MESSAGES:
        locale = "en_US"
        text_string = "English (United States)"
        text_string_id = 106
        codeset = "ISO8859-1"
        package = "bos.loc.iso.en_US bos.msg.en_US.rte
bos.msg.en_US.net.tcp.client"
        variables = "LC_MESSAGES=en_US"
        bosinst_translated = "y"
        bosinst_menu = "y" -a no = yes ]
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g locale -f CULTURAL_CONVENTION
CC=en_US
+ + odmget -qlocale=en_US AND bosinst_menu=y CC
+ 2> /dev/null
CC_OK=
CC:
        locale = "en_US"
        text_string = "English (United States)"
        text_string_id = 106
        codeset = "ISO8859-1"
        messages = "en_US"
        keyboards = "en_US en_US@alt"
        package = "bos.loc.iso.en_US"
        variables = "LANG=en_US"
        sbcs_variables = ""
        bosinst_menu = "y"
        icon_path = ""
        menu = "101"
        messageLink = "en_US"
        keyboardLink = "en_US"
+ [ !
CC:
        locale = "en_US"
        text_string = "English (United States)"
        text_string_id = 106
        codeset = "ISO8859-1"
        messages = "en_US"
        keyboards = "en_US en_US@alt"
        package = "bos.loc.iso.en_US"
        variables = "LANG=en_US"
        sbcs_variables = ""
        bosinst_menu = "y"
        icon_path = ""
        menu = "101"
        messageLink = "en_US"
        keyboardLink = "en_US" -a no = yes ]
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g locale -f MESSAGES
|MSG=en_US
+ + odmget -qlocale=en_US AND bosinst_menu=y MESSAGES
+ 2> /dev/null
MSG_OK=
MESSAGES:
        locale = "en_US"
        text_string = "English (United States)"
        text_string_id = 106
        codeset = "ISO8859-1"
        package = "bos.loc.iso.en_US bos.msg.en_US.rte
bos.msg.en_US.net.tcp.client"
        variables = "LC_MESSAGES=en_US"
        bosinst_translated = "y"
        bosinst_menu = "y"
+ [ !
MESSAGES:
        locale = "en_US"
        text_string = "English (United States)"
        text_string_id = 106
        codeset = "ISO8859-1"
        package = "bos.loc.iso.en_US bos.msg.en_US.rte
bos.msg.en_US.net.tcp.client"
        variables = "LC_MESSAGES=en_US"
        bosinst_translated = "y"
        bosinst_menu = "y" -a no = yes ]
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g locale -f KEYBOARD
KBD=en_US
+ + odmget -qkeyboard_map=en_US AND bosinst_menu=y KEYBOARD
+ 2> /dev/null
KBD_OK=
KEYBOARD:
        locale = "en_US"
        keyboard_map = "en_US"
        text_string = "English (United States)"
        text_string_id = 106
        codeset = "ISO8859-1"
        package = "bos.loc.iso.en_US"
        variables = ""
        keyboard_cmd = "/usr/bin/chkbd /usr/lib/nls/loc/en_US.lftkeymap"
        key_text = "English (United States) keyboard."
        key_text_id = 206
        bosinst_menu = "y"
+ [ !
KEYBOARD:
        locale = "en_US"
        keyboard_map = "en_US"
        text_string = "English (United States)"
        text_string_id = 106
        codeset = "ISO8859-1"
        package = "bos.loc.iso.en_US"
        variables = ""
        keyboard_cmd = "/usr/bin/chkbd /usr/lib/nls/loc/en_US.lftkeymap"
        key_text = "English (United States) keyboard."
        key_text_id = 206
        bosinst_menu = "y" -a no = yes ]
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f TCB
TCB=no
+ [ no = C2 -a  != 1 ]
+ return 0
+ [ 5 -eq 5 -a  -eq 1 ]
+ Get_User_Input
+ trap BI_Error "BOS Install" 29 2 INT
+ + /../SPOT/usr/lpp/bosinst/bidata -S
status=0
+ [ 0 -ne 0 ]
+ cat /tmp/Animate.pn
+ kill -9 3894
+ rm /tmp/Animate.pn
+ /usr/lpp/bosinst/bi_io -t

   + rc=2
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f PROMPT
P=no
+ NOTDONE=TRUE
+ [ TRUE ]
+ [ no = yes ]
+ /../SPOT/usr/lpp/bosinst/bidata -w
+ [  ]
+ /../SPOT/usr/lpp/bosinst/bidata -b -g locale -f BOSINST_LANG
+ export LANG=en_US
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f BOSINST_DEBUG
BOSINST_DEBUG=
+ export BOSINST_DEBUG
+ Display_Status_Screen
+ trap BI_Error "BOS Install" 29 2 INT
+ [  = -c ]
+ START_TIME=50
+ echo




+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f INSTALL_METHOD
IM=overwrite
+ [[ overwrite = migrate ]]
+ /usr/lpp/bosinst/berror -f   -e 30 -a 1

                        Installing Base Operating System

If you used the system key to select SERVICE mode,
turn the system key to the NORMAL position any time before
the installation ends.


        Please wait...


        Approximate     Elapsed time
     % tasks complete   (in minutes)


+ echo 0
+ 1> /../percent
+ echo
+ 1> /../wip
+ [ -z set -x -a 524288 -gt 8192 ]
+ return 0
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g vg_data -f PPSIZE
PPSZ=8
+ Set_PP_Size
+ trap BI_Error "BOS Install" 29 2 INT
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f INSTALL_METHOD
IM=overwrite
+ [ overwrite = overwrite ]
+ PPSZ_VG=0
+ /../SPOT/usr/lpp/bosinst/bidata -b -g target_disk_data -f HDISKNAME
+ + bootinfo -P 0 -s hdisk0
PPSZ_PV=8
+ [ PPSZ_PV -gt 0 ]
+ PPSZ_VG=8
+ [ -z 8 -o 8 -gt 8 ]
+ /../SPOT/usr/lpp/bosinst/bidata -i -m vg_data -f PPSIZE -e 8 -c VGNAME -v rootvg
+ return 0
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f INSTALL_METHOD
IM=overwrite
+ [ overwrite = overwrite ]
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g image_data -f PRODUCT_TAPE
PT=no
+ + /../SPOT/usr/lpp/bosinst/bidata -i -g logical_volume_policy -f SHRINK
SHR=no
+ [ no = no -a no = yes ]
+ CheckSize
+ rc=0
+ [ 0 -eq 0 ]
+ NOTDONE=
+ [ nonturbo = overflow ]
+ [ -x /usr/lpp/diagnostics/bin/diagsrv ]
+ FAILED_DIAG=
+ /../SPOT/usr/lpp/bosinst/bidata -b -g target_disk_data -f HDISKNAME
+ /usr/lpp/diagnostics/bin/diagsrv hdisk0
+ 1> /dev/null 2>& 1
 dctrl  dctrl  dctrl + rc=0
+ [ 0 -ne 0 ]
+ [  ]
+ showled  [  ]
+ /usr/lib/methods/showled 0xA46
+ /../SPOT/usr/lpp/bosinst/bidata -b -g locale -f BOSINST_LANG
+ export LANG=en_US
+ rm -f /tmp/Get_RVG_Disks.pn
+ [ 5 -eq 5 ]
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=
+ return 0
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f TCB
tcb=no
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f INSTALL_X_IF_ADAPTER
IXIA=no
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f INSTALL_METHOD
IM=overwrite
+ [ nonturbo = turbo ]
+ Initialize_Log
+ trap BI_Error "BOS Install" 29 2 INT
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f BOSINST_DEBUG
BOSINST_DEBUG=
+ [[  = yes ]]
+ [[ -f /tmp/.BI_dbg ]]
+ export VERBOSE=
+ bootinfo -p
+ [ chrp = rs6k ]
+ return 0
+ [ 0 -eq 1 ]
+ [ -z  ]
+ Log Shrink_It
+ rm -f /tmp/badret
+ [ 0 -ne 1 ]
+ Shrink_It
+ 1>> /var/adm/ras/bi.log 2>> /var/adm/ras/bi.log
+ Log  showled Prepare_Target_Disks
+ rm -f /tmp/badret
+ [ 0 -ne 1 ]
+ Prepare_Target_Disks
+ 1>> /var/adm/ras/bi.log 2>> /var/adm/ras/bi.log
 showled + [ -z  ]
+ /usr/lib/methods/sh showled owled 0xA54
+ Log Restore_System
+ rm -f /tmp/badret
+ [ 0 -ne 1 ]
+ Restore_System
+ 1>> /var/adm/ras/bi.log 2>> /var/adm/ras/bi.log
+ /usr/lib/metho showled ds/showled 0xA52
+ Log Initialize_Disk_Environment
+ rm -f /tmp/badret
+ [ 0 -ne 1 ]
+ Initialize_Disk_Environment
+ 1>> /var/adm/ras/bi.log 2>> /var/adm/ras/bi.log
+ Log Change_Mounts
+ rm -f /tmp/badret
+ [ 0 -ne 1 ]
+ Change_Mounts
+ 1>> /var/adm/ras/bi.log 2>> /var/adm/ras/bi.log
+ Log Copy_Customized
+ rm -f /tmp/badret
+ [ 0 -ne 1 ]
+ Copy_Customized
+ 1>> /var/adm/ras/bi.log 2>> /var/adm/ras/bi.log
+ /usr/lib/methods/showled 0xA46
 showled + [ -z  ]
+ Log Post_Install
+ rm -f /tmp/badret
+ [ 0 -ne 1 ]
+ Post_Install
+ 1>> /var/adm/ras/bi.log 2>> /var/adm/ras/bi.log



            perfmgr.network 2.2.1.0
    perfmgr.local 2.2.1.0
    perfmgr.common 2.2.1.0

    perfmgr.network 2.2.1.0
    perfmgr.local 2.2.1.0
    perfmgr.common 2.2.1.0


installp:  APPLYING software for:
        devices.common.rspcbase.rte 4.3.2.0


. . . . . << Copyright notice for devices.common.rspcbase >> . . . . . . .
```

```
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1991, 1998.

 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.common.rspcbase >>. . . .


Filesets processed:  1 of 4
System Installation Time: 11 minutes       Tasks Complete: 86%

installp:  APPLYING software for:
        devices.common.IBM.modemcfg.data 4.3.1.0


. . . . . << Copyright notice for devices.common.IBM.modemcfg >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1996, 1998.

 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.common.IBM.modemcfg >>. . . .


Filesets processed:  2 of 4
System Installation Time: 11 minutes       Tasks Complete: 86%

installp:  APPLYING software for:
        devices.chrp.base.rte 4.3.2.0
        devices.chrp.base.diag 4.3.2.0


. . . . . << Copyright notice for devices.chrp.base >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1996, 1998.

 All rights reserved.
 US Government Users Restricted Rights - Use, duplication or disclosure
 restricted by GSA ADP Schedule Contract with IBM Corp.


 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1995, 1998.

 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.chrp.base >>. . . .

Finished processing all filesets.
```

```
System Installation Time: 11 minutes       Tasks Complete: 87%
    perfmgr.network 2.2.1.0
    perfmgr.local 2.2.1.0
    perfmgr.common 2.2.1.0

FAILURES
--------
  Filesets listed in this section failed pre-installation verification
  and will not be installed.

  Missing Filesets
  ----------------
  The following filesets could not be found on the installation media.
  If you feel these filesets really are on the media, check for typographical
  errors in the name specified or, if installing from directory, check for
  discrepancies between the Table of Contents file (.toc) and the images that
  reside in the directory.

    devices.chrp.IBM.TB3MX
    devices.isa_sio.8042
    devices.isa_sio.fdc
    devices.isa_sio.serial
    devices.pci.AAPL.NCR8250S
    devices.pci.NCR.53C825
    devices.pci.NCR.8251S
    devices.pci.SYM.53C825
    devices.pci.SYM.8251S
    devices.pci.ethernet
    devices.pci.scsi

  << End of Failure Section >>

    perfmgr.network 2.2.1.0
    perfmgr.local 2.2.1.0
    perfmgr.common 2.2.1.0

    perfmgr.network 2.2.1.0
    perfmgr.local 2.2.1.0
    perfmgr.common 2.2.1.0


installp:  APPLYING software for:
        devices.sys.pci.rte 4.3.2.0


. . . . . << Copyright notice for devices.sys.pci >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1991, 1998.

 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.sys.pci >>. . . .


Filesets processed:  1 of 26
System Installation Time: 11 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.pci.86808404.rte 4.3.2.0
        devices.pci.86808404.com 4.3.0.0

. . . . . << Copyright notice for devices.pci.86808404 >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1991, 1998.

 All rights reserved.
 US Government Users Restricted Rights - Use, duplication or disclosure
 restricted by GSA ADP Schedule Contract with IBM Corp.


 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1993, 1998.

 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.pci.86808404 >>. . . .


Filesets processed:  3 of 26
System Installation Time: 11 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.pci.22100020.rte 4.3.2.0
        devices.pci.22100020.diag 4.3.2.0


. . . . . << Copyright notice for devices.pci.22100020 >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1994, 1998.

 All rights reserved.
 US Government Users Restricted Rights - Use, duplication or disclosure
 restricted by GSA ADP Schedule Contract with IBM Corp.


 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1995, 1998.

 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.pci.22100020 >>. . . .


Filesets processed:  5 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.pci.00100f00.rte 4.3.2.0
```

```
. . . . . << Copyright notice for devices.pci.00100f00 >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1991, 1998.

 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.pci.00100f00 >>. . . .


Filesets processed:  6 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.pci.00100300.rte 4.3.2.0
        devices.pci.00100300.diag 4.3.2.0


. . . . . << Copyright notice for devices.pci.00100300 >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1991, 1998.

 All rights reserved.
 US Government Users Restricted Rights - Use, duplication or disclosure
 restricted by GSA ADP Schedule Contract with IBM Corp.


 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1995, 1998.

 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.pci.00100300 >>. . . .


Filesets processed:  8 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.pci.00100100.com 4.3.2.0
        devices.pci.00100100.rte 4.3.2.0


. . . . . << Copyright notice for devices.pci.00100100 >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1991, 1998.

 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.pci.00100100 >>. . . .


Filesets processed:  10 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.pci.isa.rte 4.3.2.0


. . . . . << Copyright notice for devices.pci.isa >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1996, 1998.

 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.pci.isa >>. . . .


Filesets processed:  11 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.isa_sio.pnpPNP.700.rte 4.3.2.0
        devices.isa_sio.pnpPNP.700.diag 4.3.0.0


. . . . . << Copyright notice for devices.isa_sio.pnpPNP.700 >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1995, 1998.

 All rights reserved.
 US Government Users Restricted Rights - Use, duplication or disclosure
 restricted by GSA ADP Schedule Contract with IBM Corp.


 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1996, 1998.

 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.isa_sio.pnpPNP.700 >>. . . .


Filesets processed:  13 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.isa_sio.pnpPNP.501.rte 4.3.2.0
        devices.isa_sio.pnpPNP.501.diag 4.3.0.0


. . . . . << Copyright notice for devices.isa_sio.pnpPNP.501 >> . . . . . . .
```

```
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1991, 1998.

 All rights reserved.
 US Government Users Restricted Rights - Use, duplication or disclosure
 restricted by GSA ADP Schedule Contract with IBM Corp.


 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1996, 1998.

 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.isa_sio.pnpPNP.501 >>. . . .


Filesets processed:  15 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.isa_sio.km.diag 4.3.2.0


. . . . . << Copyright notice for devices.isa_sio.km >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1995, 1998.

 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.isa_sio.km >>. . . .


Filesets processed:  16 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.isa_sio.chrp.8042.rte 4.3.2.0
        devices.isa_sio.chrp.8042.diag 4.3.0.0


. . . . . << Copyright notice for devices.isa_sio.chrp.8042 >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1993, 1998.

 All rights reserved.
 US Government Users Restricted Rights - Use, duplication or disclosure
 restricted by GSA ADP Schedule Contract with IBM Corp.


 Licensed Materials - Property of IBM
 5765C3403
   (C) Copyright International Business Machines Corp. 1996, 1998.

 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.isa_sio.chrp.8042 >>. . . .


Filesets processed:  18 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.isa_sio.PNP0F03.diag 4.3.2.0


. . . . . << Copyright notice for devices.isa_sio.PNP0F03 >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1995, 1998.

 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.isa_sio.PNP0F03 >>. . . .


Filesets processed:  19 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.isa_sio.PNP0700.rte 4.3.2.0


. . . . . << Copyright notice for devices.isa_sio.PNP0700 >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1995, 1998.

 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.isa_sio.PNP0700 >>. . . .


Filesets processed:  20 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.isa_sio.PNP0501.rte 4.3.2.0


. . . . . << Copyright notice for devices.isa_sio.PNP0501 >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1991, 1998.
 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.isa_sio.PNP0501 >>. . . .


Filesets processed:  21 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.isa_sio.PNP0303.diag 4.3.2.0


. . . . . << Copyright notice for devices.isa_sio.PNP0303 >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1995, 1998.

 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.isa_sio.PNP0303 >>. . . .


Filesets processed:  22 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.common.IBM.pmmd_chrp.rte 4.3.2.0


. . . . . << Copyright notice for devices.common.IBM.pmmd_chrp >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1991, 1998.

 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.common.IBM.pmmd_chrp >>. . . .


Filesets processed:  23 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        devices.chrp.pci.rte 4.3.2.0


. . . . . << Copyright notice for devices.chrp.pci >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1996, 1998.

 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.chrp.pci >>. . . .


Filesets processed:  24 of 26
System Installation Time: 12 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        bos.powermgt.rte 4.3.2.0


. . . . . << Copyright notice for bos.powermgt >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1985, 1998.

 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 bos.powermgt >>. . . .


Filesets processed:  25 of 26
System Installation Time: 13 minutes       Tasks Complete: 88%

installp:  APPLYING software for:
        X11.apps.pm 4.3.2.0


. . . . . << Copyright notice for X11.apps >> . . . . . . .
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1985, 1998.

 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 X11.apps >>. . . .

Finished processing all filesets.
System Installation Time: 13 minutes       Tasks Complete: 89%
 Dctrl  Dctrl  Dctrl  Dctrl  cfgif  cfgmgr




                        Installing Base Operating System

If you used the system key to select SERVICE mode,
turn the system key to the NORMAL position any time before the
installation ends.

        Please wait...




        Approximate     Elapsed time
     % tasks complete   (in minutes)
 showled  showled  showled  showled  showled  showled  showled  showled  showled
showled  showled  showled  showled  showled  showled  showled  showled  showled
showled  showled  showled  showled  showled  showled  showled  showled  showled
showled  showled  showled  showled  showled  showled  showled  showled  showled
cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfgsys_chrp  cfgmgr  cfgmgr  cfgpmchrp
cfgmgr  cfgmgr  cfgbus_pcic  cfgbus_isac  cfgbus_pcic  cfgmgr  cfgmgr  cfgtb3
cfgmgr  cfgmgr  cfgasync_rspc  cfgmgr  cfgmgr  cfgasync_rspc  cfgmgr  cfgmgr
cfgkma_chrp  cfgmgr  cfgmgr  cfgfda_chrp  cfgmgr  cfgmgr  cfgncr_scsi  cfgmgr
cfgmgr  cfgkent  cfgmgr  cfgmgr  cfgtty  cfgmgr  cfgmgr  cfgkm_chrp  cfgmgr  cfgmgr
cfgkm_chrp  cfgmgr  cfgmgr  cfgscdisk  cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfgmgr
cfgmgr  cfgmgr  cfgmgr  cfgif  cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfgpty  cfgmgr
cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfgmgr
cfgmgr  showled  showled  showled  showled  showled  showled  showled  showled
showled  showled  showled + /usr/lib/metho showled ds/showled 0xA56
+ Log Run_Customization
+ rm -f /tmp/badret
+ [ 0 -ne 1 ]
+ showled  Run_Customization
+ 1>> /var/adm/ras/bi.log 2>> /var/adm/ras/bi.log



+ /usr/lib/methods/showled 0xA46
+ Finish
+ trap BI_Error "BOS Install" 29 2 INT
+ [ 0 -eq 1 ]
+ Change_Status 100
+ trap BI_Error "BOS Install" 29 2 INT
+ [ 100 -ne 0 ]
+ [ 100 -gt 100 ]
+ PERCENT=100
+ BAR=****************
+ [ -n  ]
+ [ -n  ]
+ echo 100
+ 1> /../percent
+ [ 5 -eq 5 ]
+ [ -z  ]
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=BOS install 100%
complete
+ [ 524288 -le 8192 ]
+ return 0
+ [ -z  ]
+ alog -t bosinst -q -s 16384
+ cat /../var/adm/ras/bi.log /var/adm/ras/bi.log
+ rm -f /var/adm/ras/bi.log
+ [ -z  ]
+ cp /../var/adm/ras/BosMenus.log /var/adm/ras
+ 2> /dev/null
+ cp /../bosinst.data /var/adm/ras
+ 2> /dev/null
+ cp /../image.data /var/adm/ras
+ 2> /dev/null
+ cp /../Ch_Stat.log /var/adm/ras
+ 2> /dev/null
+ + /../SPOT/usr/lpp/bosinst/bidata -b -g control_flow -f CONSOLE
C=/dev/tty0
+ [ chrp != rs6k ]
+ KEYPOS=3
+ [ 3 -ne 3 -o  = true ]
+ + pwd
```

```
CWD=/
+ cd /tmp
+ ar x /usr/lpp/bos/liblpp.a
+ echo

+ cat bos.rte.copyright
 Licensed Materials - Property of IBM

 5765C3403
   (C) Copyright International Business Machines Corp. 1985, 1998.
   (C) Copyright AT&T 1984, 1985, 1986, 1987, 1988, 1989.
   (C) Copyright Regents of the University of California 1980, 1982,
                                1983, 1985, 1986, 1987, 1988, 1989.
   (C) Copyright BULL 1993, 1998.
   (C) Copyright Digi International Inc. 1988-1993.
   (C) Copyright Interactive Systems Corporation 1985, 1991.
   (C) Copyright (c) ISQUARE, Inc. 1990.
   (C) Copyright Mentat Inc. 1990, 1991.
   (C) Copyright Open Software Foundation, Inc. 1989, 1994.
   (C) Copyright Sun Microsystems, Inc. 1984, 1985, 1986, 1987, 1988, 1991.

 All rights reserved.
 US Government Users Restricted Rights - Use, duplication or disclosure
 restricted by GSA ADP Schedule Contract with IBM Corp.

+ ar t /usr/lpp/bos/liblpp.a
+ rm -f bos.rte.copyright bos.rte.cfgfiles bos.rte.post_i bos.rte.pre_i
bos.rte.usr.rmlist bos.rte.root.rmlist incompat.pkgs productid bos.rte.inventory
bos.rte.al bos.rte.size bos.rte.tcb
+ cd /
+ rm -f /liblpp.a /lpp_name /SPOT /../Update_Status.pn
+ [ 5 -eq 5 ]
+ nimclient -R success
+ nimclient -S shutdown
+ PLAT_HOOK=
+ [ -x /../SPOT/platform/chrp/plat_last_chance ]
+ [ -x /platform/chrp/plat_last_chance ]
+ [ -x /../platform/chrp/plat_last_chance ]
+ [ -n  ]
+ SHUTDOWN_F=
+ bootlist
+ rc=0
+ [ 0 -ne 0 -a /dev/tty0 = none ]
+ [ 0 -ne 0 -a 5 -ne 5 ]
+ [[ 0 -eq 0 ]]
+ [[ 5 -eq 3 ]]
+ [[ 5 -eq 3 ]]
+ mv /SPOT.save.2324 /SPOT
+ 1> /dev/null 2>& 1
+ + cut -c 9-10
+ uname -m
+ 2> /dev/null
+ 2> /dev/null
MAC_MODEL=4C
+ [ overwrite = migrate -a  != 3.2 ]
+ sync
+ sync
+ sync
+ [[ -n  ]]
+ [[ 5 -eq 5 ]]
+ [[ 4C = 4C ]]
+ [[ chrp = rspc ]]
+ reboot -q
```

```
Rebooting . . .



RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000



     memory      keyboard     network      scsi      speaker
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000                                 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000       STARTING SOFTWARE         RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000         PLEASE WAIT...          RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000                                 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000
RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000 RS/6000

-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-
\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\
|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|/-\|
-------------------------------------------------------------------------------
                                Welcome to AIX.
                       boot image timestamp: 18:34 10/23
                 The current time and date: 18:36:23 10/23/1998
                number of processors: 4    size of memory: 512Mb
boot device: /pci@fef00000/scsi@c/sd@0:2
closing stdin and stdout...
```

```
-------------------------------------------------------------------------------

Saving Base Customize Data to boot disk
Starting the sync daemon
Starting the error daemon
System initialization completed.
Fri Oct 23 14:37:52 1998
Starting Multi-user Initialization
 Performing auto-varyon of Volume Groups
 Activating all paging spaces
swapon: Paging device /dev/hd6 activated.
/dev/rhd1 (/home): ** Unmounted cleanly - Check suppressed
 Performing all automatic mounts
Multi-user initialization completed
System reconfiguration in progress.  Please wait.




AIX Version 4
(C) Copyrights by IBM and by others 1982, 1996.
Console login: TB3 device already configured.

en0
sp4n10.msc.itso.ibm.com
inet0 changed
en0 changed
TB3 device already configured.

Checking for srcmstr active...complete
Starting tcpip daemons:
0513-059 The syslogd Subsystem has been started. Subsystem PID is 6488.
0513-059 The sendmail Subsystem has been started. Subsystem PID is 5312.
0513-059 The portmap Subsystem has been started. Subsystem PID is 5748.
0513-059 The inetd Subsystem has been started. Subsystem PID is 7146.
0513-059 The snmpd Subsystem has been started. Subsystem PID is 4976.
0513-059 The dpid2 Subsystem has been started. Subsystem PID is 4486.
Starting NFS services:
0513-059 The biod Subsystem has been started. Subsystem PID is 6788.
0513-059 The rpc.statd Subsystem has been started. Subsystem PID is 7276.
0513-059 The rpc.lockd Subsystem has been started. Subsystem PID is 4178.
Redirecting console output to /var/adm/SPlogs/sysman/sp4n10.console.log
```

MCA Node NIM Debug Output

```
>0> 001f3d1c              st 001f3d1c 2
>0> go

AIX Version 4.3
Starting NODE#000 physical CPU#001 as logical CPU#001... done.
Starting NODE#000 physical CPU#002 as logical CPU#002... done.
Starting NODE#000 physical CPU#003 as logical CPU#003... done.
Starting NODE#000 physical CPU#004 as logical CPU#004... done.
Starting NODE#000 physical CPU#005 as logical CPU#005... done.
Starting NODE#000 physical CPU#006 as logical CPU#006... done.
Starting NODE#000 physical CPU#007 as logical CPU#007... done.
+ [ 1 -ne 1 ]
+ PHASE=1
+ + bootinfo -p
PLATFORM=rs6k
+ [ ! -x /usr/lib/boot/bin/bootinfo_rs6k ]
+ [ 1 -eq 1 ]
+ 1> /usr/lib/libc.a
+ init -c unlink /usr/lib/boot/bin/!(*_rs6k)
+ + bootinfo -t
BOOTYPE=5
+ [ 0 -ne 0 ]
+ [ -z 5 ]
+ chramfs -t
+ init -c unlink /usr/sbin/chramfs
+ 1> /dev/null
+ unset NIM_DEBUG
+ export NIM_DEBUG=set -x
+ unset fd_invoker
+ set -x
+ /usr/lib/methods/showled 0x600
 showled + botinfo -b
+ [  = atm0 ]
+ cfgmgr -fv
 cfgmgr  cfgmgr  cfgmgr  cfgsys_p  cfgmgr  fgmgr  cfgbus  cfgbus  cfgbus  cfgbus
cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbus
cfgbus  cfgbus  cfgbus  cfgbus  cfgsio  cfgbus  cfgmgr  cfgmgr  cfgbus  cfgbus
cfgbus  cfgbus  cfbus  cfgbus  gbus  cfgbus  cfgbus  cfgbs  cfgbus  cfgbus  cfgbus
cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgmgr  cfgmgr  cfgmgr Method error
(/etc/methods/cfgasync -1 -l sa0):
sh: /etc/methods/cfgasync:  not found

 cfgmgr  cfgmgr Method error (/etc/methods/cfgasync -1 -l sa1):
sh: /etc/methods/cfgasync:  not found

 cfgmgr  cfgmgr Method error (/etc/methds/cfgasync -1 -l sa2):
sh: /etc/methods/cfgasync:  not found

 cfgmgr  cfgmr Method error (/usr/lib/methods/cfgfda -1 -l fda0):
sh: /usr/lib/methods/cfgfda:  not found

 cfgmgr  cfgmgr Method error (/usr/lib/methods/cfgppa -1 -l ppa0):
sh: /usr/lib/methods/cfgppa:  not found

 cfgmgr  cfgmgr Method error (/usr/lib/mthods/cfgssa -1 -l ssa0):
sh: /usr/lib/methods/cfgssa:  not found

 cfgmgr  cfgmgr Method error (/etc/methods/cfgascsi -1 -l ascsi0):
sh: /etc/methods/cfgascsi:  not found

 cfgmgr  cfgent  cfgmgr  cfgmgr  cfgmgr Method error (/usr/lib/methods/cfgssa -1
-l ssa1):
sh: /usr/lib/methods/cfgssa:  not found

 cfgmgr  cfgmgr Method error (/usr/lib/methods/deflvm ):
0514-068 Cause not known.
sh: /usr/lib/methods/deflvm:  not found

 cfgmgr  cfgmgr cfgmgr is running in phase 1
----------------------------------------------------------------------
invoking top level program -- "/etc/methods/defsys"
return code = 0
****************** stdout ***********
sys0

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sys0'
invoking /usr/lib/methods/cfgsys_p -1 -l sys0
return code = 0
****************** stdout ***********
bus0 bus1
```

```
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'bus0'
invoking /usr/lib/methods/cfgbus -1 -l bus0
return code = 0
****************** stdout ***********
:devices.mca.8f69 sa0,sa1,sa2,fda0,ppa0,ssa0,ascsi0

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'bus1'
invoking /usr/lib/methods/cfgbus -1 -l bus1
return code = 0
****************** stdout ***********
ent0,ssa1

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sa0'
invoking /etc/methods/cfgasync -1 -l sa0
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/cfgasync:  not found

----------------------------------------------------------------------
attempting to configure device 'sa1'
invoking /etc/methods/cfgasync -1 -l sa1
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/cfgasync:  not found

----------------------------------------------------------------------
attempting to configure device 'sa2'
invoking /etc/methods/cfgasync -1 -l sa2
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/cfgasync:  not found

----------------------------------------------------------------------
attempting to configure device 'fda0'
invoking /usr/lib/methods/cfgfda -1 -l fda0
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgfda:  not found

----------------------------------------------------------------------
attempting to configure device 'ppa0'
invoking /usr/lib/methods/cfgppa -1 -l ppa0
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgppa:  not found

----------------------------------------------------------------------
attempting to configure device 'ssa0'
invoking /usr/lib/methods/cfgssa -1 -l ssa0
return code = 127
****************** no stdout ***********
****************** stderr ***********
```

```
sh: /usr/lib/methods/cfgssa:  not found

----------------------------------------------------------------------
attempting to configure device 'ascsi0'
invoking /etc/methods/cfgascsi -1 -l ascsi0
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/cfgascsi:  not found

----------------------------------------------------------------------
attempting to configure device 'ent0'
invoking /usr/lib/methods/cfgent -1 -l ent0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'ssa1'
invoking /usr/lib/methods/cfgssa -1 -l ssa1
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgssa:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deflvm"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/deflvm:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defssar"
return code = 127
****************** no stdout *****Method error (/usr/lib/methods/defssar ):
0514-068 Cause not known.
sh: /usr/lib/methods/defssar:  not found

 cfgmgr  fgmgr Method error (/usr/lib/methods/deftmssar ):
0514-068 Cause not known.
sh: /usr/lib/methods/deftmssar:  not found

 cfgmgr ******
***************** stder ***********
sh: /usr/lib/methods/defssar:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deftmssar"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/deftmssar:  not found

+ export NSORDER=local
+ bootinfo -b
+ [ ent0 = atm0 ]
+ native_netboot_cfg
+ bootinfo -c
+ set -- 192.168.6.13 192.168.6.1 192.168.6.1 68 16 0
/tftpboot/sp5n13.msc.itso.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.
0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0. 7
+ CLIENT_IPADDR=192.168.6.13
+ BOOT_SERV_IP=192.168.6.1
+ BOOT_GATE_IP=192.168.6.1
+ E802=0
+ BOOTFILE=/tftpboot/sp5n13.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
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
+ [ -n 255.255.255.0 ]
+ SUBMASK=netmask 255.255.255.0
+ [ 192.168.6.1 = 0 -o 192.168.6.1 = 0.0.0.0 -o 192.168.6.1 = 192.168.6.1 ]
+ unset BOOT_GATE_IP
+ + bootinfo -b
PHY_BOOT_DEV=ent0
+ pdev_to_ldev
+ /usr/lib/methods/showled 0x606
 showled + ifconfig lo0 inet 127.0.0.1 up
+ ifconfig en0 inet 192.168.6.13 up netmask 255.255.255.0
+ [ 0 -ne 0 ]
+ [  ]
+ [ -n  ]
+ CLIENT_INFO_FILE=/tftpboot/sp5n13.msc.itso.ibm.com.info
+ [ 0 -ne 0 ]
+ /usr/lib/methods/showled 0x608
 showled + tftp -go /SPOT/niminfo 192.168.6.1
/tftpboot/sp5n13.msc.itso.ibm.com.info imae
Received 1417 Bytes in 0.0 Seconds
+ [ -s /SPOT/niminfo ]
+ . /SPOT/niminfo
+ export NIM_NAME=sp5n13
+ export NIM_HOSTNAME=sp5n13.msc.itso.ibm.com
+ export NIM_CONFIGURATION=standalone
+ export NIM_MASTER_HOSTNAME=sp5nb01.msc.itso.ibm.com
+ export NIM_MASTER_PORT=1058
+ export NIM_REGISTRATION_PORT=1059
+ 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=sp5n13.msc.itso.ibm.com
+ export
SPOT=sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/aix432/spot/spot_aix432/usr
+ export NIM_BOSINST_DATA=/NIM_BOSINST_DATA
+ export NIM_CUSTOM=/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script -a
location=sp5nb01.msc.itso.ibm.com:/export/nim/scripts/sp5n13.script
+ export NIM_BOS_IMAGE=/NIM_BOS_IMAGE
+ export NIM_BOS_FORMAT=mksysb
+ export NIM_HOSTS= 192.168.6.13:sp5n13.msc.itso.ibm.com
192.168.6.1:sp5nb01.msc.itso.ibm.com  192.168.5.150:sp5en0.msc.itso.ibm.com
+ export NIM_MOUNTS=
sp5en0.msc.itso.ibm.com:/spdata/sys1/install/aix432/lppsource:/SPOT/usr/sys/inst.
images:dir
sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/pssp/13.noprompt:/NIM_BOSINST_DATA:
file
sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.432:/NIM_BOS_IMA
GE:file
+ export ROUTES= 192.168.5.0:255.255.255.0:192.168.6.1
+ [ -n  192.168.6.13:sp5n13.msc.itso.ibm.com
192.168.6.1:sp5nb01.msc.itso.ibm.com  192.168.5.150:sp5en0.msc.itso.ibm.com  ]
+ OIFS=

+ IFS=:
+ set -- 192.168.6.13 sp5n13.msc.itso.ibm.com
+ IFS=
+ echo 192.168.6.13 sp5n13.msc.itso.ibm.com
+ 1>> /etc/hosts
+ OIFS=

+ IFS=:
+ set -- 192.168.6.1 sp5nb01.msc.itso.ibm.com
+ IFS=

+ echo 192.168.6.1 sp5nb01.msc.itso.ibm.com
+ 1>> /etc/hosts
+ OIFS=

+ IFS=:
+ set -- 192.168.5.150 sp5en0.msc.itso.ibm.com
+ IFS=

+ echo 192.168.5.150 sp5en0.msc.itso.ibm.com
+ 1>> /etc/hosts
+ [ -n  192.168.5.0:255.255.255.0:192.168.6.1  ]
+ OIFS=

+ IFS=:
+ set -- 192.168.5.0 255.255.255.0 192.168.6.1
+ IFS=

+ [ 3 -ne 3 ]
+ route -v add -net 192.168.5.0 -netmask 255.255.255.0 192.168.6.1
so_dst: inet 192.168.5.0; so_gate: inet 192.168.6.1; RTM_ADD: Add Route
pid: 0, len 380, seq 1, errno 0, flags:<UP,GATEWAY>
number of gids: 0

locks:  inits:
sockaddrs: <DST,GATEWAY,NETMASK>
 192.168.5.0 sp5nb01.msc.itso.ibm.com (0) ffff ff00 0 0 0 0 0
192.168.6.1 net 192.168.5.0: gateway 192.168.6.1
+ 1> /etc/filesystems
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=LED 610: mount -r
sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/aix432/spot/spot_aix432/usr
/SPOT/usr
+ /usr/lib/methods/showled 0x610
 showled + mount -r
sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/aix432/spot/spot_aix432/usr
/SPOT/usr
+ [[ 0 -ne 0 ]]
+ nimclient -o change -a force=yes -a ignore_lock=yes -a info=
+ /usr/lib/methods/showled 0x612
 showled + cp /SPOT/usr/lib/boot/network/rc.bos_inst /etc
+ RC_CONFIG=/etc/rc.bos_inst
+ . /etc/rc.bos_inst
+ set -x
+ /SPOT/usr/sbin/nimclient -S booting
+ mount_from_list
+ [ -n
sp5en0.msc.itso.ibm.com:/spdata/sys1/install/aix432/lppsource:/SPOT/usr/sys/inst.
images:dir
sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/pssp/13.noprompt:/NIM_BOSINST_DATA:
file
sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.432:/NIM_BOS_IMA
GE:file  ]
+ /usr/lib/methods/showled 0x610
 showled + IFS=
+ IFS=:
```

```
+ set -- sp5e0.msc.itso.ibm.com /sdata/sys1/install/aix432/lppsource
/SPOT/usr/sys/inst.images dir
+ IFS=

+ [ ! -d /SPOT/usr/sys/inst.images ]
+ /SPOT/usr/sbin/nimclient -o change -a force=yes -a ignore_lock=yes -a info=LED
610: mount sp5en0.msc.itso.ibm.com:/spdata/sys1/install/aix432/lppsource
/SPOT/usr/sys/inst.images
+ mount sp5en0.msc.itso.ibm.com:/spdata/sys1/install/aix432/lppsource
/SPOT/usr/sys/inst.images
+ [[ 0 -ne 0 ]]
+ IFS=:
+ set -- sp5nb01.msc.itso.ibm.com /spdata/sys1/install/pssp/13.noprompt
/NIM_BOSINST_DATA file
+ IFS=

+ [ ! -d /NIM_BOSINST_DATA ]
+ /SPOT/usr/bin/mkdir -p /NIM_BOSINST_DATA
+ /SPOT/usr/sbin/nimclient -o change -a force=yes -a ignore_lock=yes -a info=LED
610: mount sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/pssp/13.noprompt
/NIM_BOSINST_DATA
+ mount sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/pssp/13.noprompt
/NIM_BOSINST_DATA
+ [[ 0 -ne 0 ]]
+ IFS=:
+ set -- sp5nb01.msc.itso.ibm.com /spdata/sys1/install/images/bos.obj.ssp.432
/NIM_BOS_IMAGE file
+ IFS=

+ [ ! -d /NIM_BOS_IMAGE ]
+ /SPOT/usr/bin/mkdir -p /NIM_BOS_IMAGE
+ /SPOT/usr/sbin/nimclient -o change -a force=yes -a ignore_lock=yes -a info=LED
610: mount sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.432
/NIM_BOS_IMAGE
+ mount sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.432
/NIM_BOS_IMAGE
+ [[ 0 -ne 0 ]]
+ /SPOT/usr/sbin/nimclient -o change -a force=yes -a ignore_lock=yes -a info=
+ /usr/lib/methods/showled 0x622
 showled ln: /usr/lib is a directory.  (cannot unlink)
ln: /usr/lib/methods is a directory.  (cannot unlink)
+ /SPOT/usr/lib/boot/network/link_methods
ln: /usr/lib/methods/defssar exists.  Specify -f to remove.
+ strload -f /dev/null
+ cfgmgr -f -v
 cfgmgr  cfgmgr  cfgmgr  cfgsys_p  cfgmgr  cfgmgr  cfgbus  cfgbus  cfgbus  cfgbus
cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cgbus  cfgbus
cfgbus  cfgbus  cfgbus  cfgbus  cfsio  cfgbus  cfgmgr  cfgmgr  cfgbus  cfgbus
cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cgbus  cfgbus  cfgbus  cfgbus
cfgbus  cfbus  cfgbus  cfgus  cfgbus  cfgbus  cfgmgr  cfgmgr  cfgasync  cfgmgr
cggr  cfgasync  cfgmgr  cfgmgr  cfgasync  cfgmgr  cfgmgr  cfgfda  cfgmgr  cfgmgr
cfgmgr Method error (/usr/lib/methods/cfgppa -1 -l ppa0):
sh: /usr/lib/methods/cfgppa:  not found

 cfgmgr  cfgssa  cfgmgr  fgmgr  cfgascsi  cfgmgr  cfgmgr  cfgent  cfgmgr Method
error (/usr/lib/methods/cfgent -1 -l ent0):
0514-061 Cannot find a child device.
 cfgmgr  cfgssa  cfgmgr  cfmgr  cfgvcsi  cfgmgr  cfgmgr  cfgvssi  cfgmgr  cfgmgr
cfgscdisk  cfgmgr cfgmgr is running in phase 1
--------------------------------------------------------------------
invoking top level program -- "/etc/methods/defsys"
return code = 0
****************** stdout ***********
```

```
sys0

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sys0'
invoking /usr/lib/methods/cfgsys_p -1 -l sys0
return code = 0
****************** stdout ***********
bus0 bus1
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'bus0'
invoking /usr/lib/methods/cfgbus -1 -l bus0
return code = 0
****************** stdout ***********
:devices.mca.8f69 sa0,sa1,sa2,fda0,ppa0,ssa0,ascsi0

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'bus1'
invoking /usr/lib/methods/cfgbus -1 -l bus1
return code = 0
****************** stdout ***********
ent0,ssa1

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sa0'
invoking /etc/methods/cfgasync -1 -l sa0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sa1'
invoking /etc/methods/cfgasync -1 -l sa1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sa2'
invoking /etc/methods/cfgasync -1 -l sa2
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'fda0'
invoking /usr/lib/methods/cfgfda -1 -l fda0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'ppa0'
invoking /usr/lib/methods/cfgppa -1 -l ppa0
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgppa:  not found

----------------------------------------------------------------------
attempting to configure device 'ssa0'
invoking /usr/lib/methods/cfgssa -1 -l ssa0
return code = 0
****************** no stdout ***********
```

```
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'ascsi0'
invoking /etc/methods/cfgascsi -1 -l ascsi0
return code = 0
****************** stdout ***********
vscsi0 vscsi1
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'ent0'
invoking /usr/lib/methods/cfgent -1 -l ent0
return code = 61
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'ssa1'
invoking /usr/lib/methods/cfgssa -1 -l ssa1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'vscsi0'
invoking /etc/methods/cfgvscsi -1 -l vscsi0
return code = 0
****************** stdout ***********
hdisk0 hdisk1
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'vscsi1'
invoking /etc/methods/cfgvscsi -1 -l vscsi1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'hdisk0'
invoking /etc/methods/cfgscdisk -1 -l hdisk0
return code = 0
************** cfgmgr  cfgscdisk  cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfglvdd  cfgmgr
cfgmg cfgmgr  cfgmgr  cfgssar  cfgmgr  cfgmgr  cfgmgr Method error
(/usr/lib/methods/deftmssar ):
0514-068 Cause not known.
sh: /usr/lib/methods/deftmssar:  not found

 cfgmgr **** no stdout **********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'hdisk1'
invoking /etc/methods/cfgscdisk -1 -l hdisk1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deflvm"
return code = 0
****************** stdout ***********
lvdd

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'lvdd'
invoking /usr/lib/methods/cfglvdd -1 -l lvdd
return code = 0
****************** no stdout ***********
```

```
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defssar"
return code = 0
****************** stdout ***********
ssar
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'ssar'
invoking /usr/lib/methods/cfgssar -1 -l ssar
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deftmssar"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/deftmssar:  not found

+ /usr/lib/methods/showled 0x622
 showled + /SPOT/usr/lib/boot/network/link_methods
ln: /usr/lib/methods/cfg_mid exists.  Specify -f to remove.
ln: /usr/lib/methods/defssar exists.  Specify -f to remove.
ln: /usr/lib/methods/cfgssa exists.  Specify -f to remove.
+ export DEV_PKGNAME=ALL
+ cfgmgr -s -v
 cfgmgr  cfggr Method error (/etc/methods/fgprobe -c /etc/dvers/coreprobe.ext ):
0514-068 Cause not known.
sh: /etc/methods/cfgprobe:  not found

 cfgmgr  cfgmgr  cfgmgr  cfgsys_p  cfgmgr  cfgmgr  cfgbus  cfgbus  cfgbus  cfgbus
cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbs  cfgbus  cfgbus  cfgbus  cfgbus
cfgbus  cfgbus  cfgbus  cfgbus  cfgsio cfgbus  cfgmgr  cgmgr  cfgus  cfgbus  cfgbus
cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbus  cfgbus
cfbus  cfgbus  cfgbus  cfgbs  cfgbus  cfgmgr  cfgmgr  cfgasync  cfgmgr  cfgmgr
cfgasync  cfmgr  cfgmgr  cfgasync  cfgmgr  cfgmgr  cfgfda  cfgmgr  cfgmgr  cfgssa
cfgmgr  cfgmgr  cfgascsi  cfmgr  cfgmgr  cfgmgr Method error
(/us/lib/methods/cfppa -2 -l ppa0):
sh: /usr/lib/methods/cfgppa:  not found

 cfgmgr  cfgent  cfgmgr Method error (/usr/lb/methods/cfgent -2 -l ent0):
0514-061 Cannot find a child device.
 cfgmgr  cfgssa  cfgmgr  cfgmgr  cfgvscsi  cfgmgr cfgmgr is running in phase 2
----------------------------------------------------------------------
invoking top level program -- "/etc/methods/cfgprobe -c
/etc/drivers/coreprobe.ext"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/cfgprobe:  not found

----------------------------------------------------------------------
invoking top level program -- "/etc/methods/defsys"
return code = 0
****************** stdout ***********
:devices.base sys0

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sys0'
invoking /usr/lib/methods/cfgsys_p -2 -l sys0
return code = 0
```

```
****************** stdout ***********
:devices.sys.mca bus0 bus1
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'bus0'
invoking /usr/lib/methods/cfgbus -2 -l bus0
return code = 0
****************** stdout ***********
:devices.mca.fed9 :devices.sio.sa :devices.sio.sa :devices.sio.fda
:devices.sio.ppa :devices.mca.8f97 :devices.mca.8f69 :devices.mca.8efc
sa0,sa1,sa2,fda0,ssa0,ascsi0,ppa0

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'bus1'
invoking /usr/lib/methods/cfgbus -2 -l bus1
return code = 0
****************** stdout ***********
:devices.mca.8ef5 :devices.mca.8f97 ent0,ssa1

****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sa0'
invoking /etc/methods/cfgasync -2 -l sa0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sa1'
invoking /etc/methods/cfgasync -2 -l sa1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'sa2'
invoking /etc/methods/cfgasync -2 -l sa2
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'fda0'
invoking /usr/lib/methods/cfgfda -2 -l fda0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'ssa0'
invoking /usr/lib/methods/cfgssa -2 -l ssa0
return code = 0
****************** stdout ***********
:devices.ssa.IBM_raid
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'ascsi0'
invoking /etc/methods/cfgascsi -2 -l ascsi0
return code = 0
****************** stdout ***********
vscsi0 vscsi1
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'ppa0'
invoking /usr/lib/methods/cfgppa -2 -l ppa0
return code = 127
```

```
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgppa:  not found

-----------------------------------------------------------------------
attempting to configure device 'ent0'
invoking /usr/lib/methods/cfgent -2 -l ent0
return code = 61
****************** no stdout ***********
****************** no stderr ***********
-----------------------------------------------------------------------
attempting to configure device 'ssa1'
invoking /usr/lib/methods/cfgssa -2 -l ssa1
return code = 0
****************** stdout ***********
:devices.ssa.IBM_raid
****************** no stderr ***********
-----------------------------------------------------------------------
attempting to configure device 'vscsi0'
invoking /etc/methods/cfgvscsi -2 -l vscsi0
return code = 0
**************** cfgmgr  cfgvscsi  cfgmgr  cfgmgr  cfgscdisk  cfgr  cfgmgr
cfscdisk  cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfgmgr Method error
(/usr/lib/methods/defops ):
0514-068 Cause not known.
sh: /usr/lib/methods/defops:  not found

 cfgmgr  cfgmgr  cfgmr  cfmgr  cfgmgr  cfgmgr  cfgmgr  cfgmgr Method error
(/usr/lib/methods/definet > /dev/null 2>&1;opt=`/usr/sbin/lsattr -E -l inet0 -a
bootup_option -F value`
if [ $opt = "no" ];then nf=/etc/rc.net
else nf=/etc/rc.bsdnet
fi;$nf -2;x=$?;test $x -ne 0&&echo $nf failed. Check for invalid commands >&2;exit
$x ):
0514-068 Cause not known.
lsattr: 0514-519 The following device was not found in the customized
device configuration database:
inet0
sh[2]: test: argument expected
sh[4]: /etc/rc.bsdnet:  not found
/etc/rc.bsdnet failed. Check for invalid commands

 cfgmgr  cfgmgr Method error (/etc/methods/ptynode ):
0514-068 Cause not known.
sh: /etc/methods/ptynode:  not found

 cfgmgr  cfgmgr  cgmgr  cfgmgr Method error (/etc/methods/startrcm ):
0514-068 Cause not known.
sh: /etc/methods/startrcm:  not found

 cfgmgr  cfgmgr Method error (/usr/lib/methods/load_rcm_stub_ext ):
0514-068 Cause not known.
sh: /usr/lib/methods/load_rcm_stub_ext:  not found

 cfgmgr  cfgmgr  cfgmgr  cfgtty  cfgmgr ** stdout ***********
:devices.scsi.disk :devices.scsi.disk hdisk0 hdisk1 :devices.scsi.tape
****************** no stderr ***********
-----------------------------------------------------------------------
attempting to configure device 'vscsi1'
invoking /etc/methods/cfgvscsi -2 -l vscsi1
return code = 0
****************** stdout ***********
:devices.scsi.tape
```

```
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'hdisk0'
invoking /etc/methods/cfgscdisk -2 -l hdisk0
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'hdisk1'
invoking /etc/methods/cfgscdisk -2 -l hdisk1
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deflvm"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defops"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/defops:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -Cc ipsec -r name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -C -c adapter -s atm -t lec_ent -F
name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -C -c adapter -s atm -t lec_tok -F
name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/definet > /dev/null
2>&1;opt=`/usr/sbin/lsattr -E -l inet0 -a bootup_option -F value`
if [ $opt = "no" ];then nf=/etc/rc.net
else nf=/etc/rc.bsdnet
fi;$nf -2;x=$?;test $x -ne 0&&echo $nf failed. Check for invalid commands >&2;exit
$x"
return code = 127
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/etc/methods/ptynode"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/ptynode:  not found

----------------------------------------------------------------------
invoking top level program -- "/etc/methods/startlft"
return code = 0
****************** no stdout ***********
```

```
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/etc/methods/startrcm"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/startrcm:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/load_rcm_stub_ext"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/load_rcm_stub_ext:  not found

----------------------------------------------------------------------
invoking top level program -- "/etc/methods/starttty"
return code = 0
****************** stdout ***********
tty0
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'tty0'
invoking /etc/methods/cfgtty -2 -l tty0
return code = 0
****************** no stdout ***********
**********/usr/sbin/mkitab: could not open /etc/inittab

 cfgmgr  cfgmgr Method error (/etc/methods/startsmt ):
0514-068 Cause not known.
sh: /etc/methods/startsmt:  not found

 cfgmgr  cfgr Method error (/et/methodsstartsgio ):
0514-068 Cause not known.
sh: /etc/methods/startsgio:  not found

 cfgmgr  cfgmgr Method error (/sr/li/methods/fdarcfgrule ):
0514-068 Cause not known.
sh: /usr/lib/methods/fdarcfgrule:  not found

 cfgmgr  cfgmgr Method error (/etc/methods/darcfgrule ):
0514-068 Cause not known.
sh: /etc/methods/darcfgrule:  not found

 cfgmgr  cfgmgr  cfgmgr  cfgssar  cfgmgr  cfgmgr  cfggr Method error
(/usr/lib/methods/deftmssar ):
0514-068 Cause not known.
sh: /usr/lib/methods/deftmssar:  not found

 cfgmgr  cfgmgr Method error (/us/lib/methods/cfgfan ):
0514-068 Cause not known.
sh: /usr/lib/methods/cfgfan:  not found

 cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfgmr  cfgmgr  cfgmgr  cfgmgr  cfgmgr  cfgmgr
cfgmgr  cfgmgr  cfgmgr  cfgmgr Method error (/usr/lib/methos/defaio ):
0514-068 Cause not known.
sh: /usr/lib/methods/defaio:  not found

******** stderr ***********
/usr/sbin/mkitab: could not open /etc/inittab

----------------------------------------------------------------------
invoking top level program -- "/etc/methods/startsmt"
```

```
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/startsmt:  not found

----------------------------------------------------------------------
invoking top level program -- "/etc/methods/startsgio"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/startsgio:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/fdarcfgrule"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/fdarcfgrule:  not found

----------------------------------------------------------------------
invoking top level program -- "/etc/methods/darcfgrule"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/darcfgrule:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defssar"
return code = 0
****************** stdout ***********
ssar
****************** no stderr ***********
----------------------------------------------------------------------
attempting to configure device 'ssar'
invoking /usr/lib/methods/cfgssar -2 -l ssar
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/deftmssar"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/deftmssar:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/cfgfan"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgfan:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -C -c dlc -s dlc -t tokenring -F
name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -C -c dlc -s dlc -t sdlc -F name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
```

```
----------------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -C -c dlc -s dlc -t x25_qllc -F
name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -C -c dlc -s dlc -t fddi -F name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -C -c dlc -s dlc -t ethernet -F
name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -C -c dlc -s dlc -t IEEE_ethernet
-F name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/defaio"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/defaio:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -C -c port -s tsd - cfgmgr  cfgmgr
cfgr  cfgmgr Method error (/usr/lib/methods/cfgvcons -l ttyvcons0 ):
0514-068 Cause not known.
sh: /usr/lib/methods/cfgvcons:  not found

 cfgmgr  cfgmgr  cfgmgr  cfgmgr Method error (/etc/methods/load_blockset_ext ):
0514-068 Cause not known.
sh: /etc/methods/load_blockset_ext:  not found

 cfgmgr t tsp -F name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/usr/lib/methods/cfgvcons -l ttyvcons0"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /usr/lib/methods/cfgvcons:  not found

----------------------------------------------------------------------
invoking top level program -- "/usr/sbin/lsdev -C -c driver -s sdlc -t scie -F
name"
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------------------------------------------------------------
invoking top level program -- "/etc/methods/load_blockset_ext"
return code = 127
****************** no stdout ***********
****************** stderr ***********
sh: /etc/methods/load_blockset_ext:  not found
```

```
+ /usr/lib/methods/showled 0x622
 showled + exit 0
+ [ 1 -ne 1 ]
+ PHASE=2
+ + bootinfo -p
PLATFORM=rs6k
+ [ ! -x /usr/lib/boot/bin/bootinfo_rs6k ]
+ [ 2 -eq 1 ]
+ + bootinfo -t
BOOTYPE=5
+ [ 0 -ne 0 ]
+ [ -z 5 ]
+ chramfs -t
/sbin/rc.boot[321]: chramfs:  not found
+ init -c unlink /usr/sbin/chramfs
+ 1> /dev/null
Could not load program init
Symbol setpcred in init is undefined
Symbol _FloatingReleaseLicense in init is undefined
Error was: Exec format error
+ unset NIM_DEBUG
+ export NIM_DEBUG=set -x
+ unset fd_invoker
+ set -x
+ export NSORDER=local
+ + bootinfo -b
PHY_BOOT_DEV=ent0
+ . /SPOT/niminfo
+ export NIM_NAME=sp5n13
+ export NIM_HOSTNAME=sp5n13.msc.itso.ibm.com
+ export NIM_CONFIGURATION=standalone
+ export NIM_MASTER_HOSTNAME=sp5nb01.msc.itso.ibm.com
+ export NIM_MASTER_PORT=1058
+ export NIM_REGISTRATION_PORT=1059
+ 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=sp5n13.msc.itso.ibm.com
+ export
SPOT=sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/aix432/spot/spot_aix432/usr
+ export NIM_BOSINST_DATA=/NIM_BOSINST_DATA
+ export NIM_CUSTOM=/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script -a
location=sp5nb01.msc.itso.ibm.com:/export/nim/scripts/sp5n13.script
+ export NIM_BOS_IMAGE=/NIM_BOS_IMAGE
+ export NIM_BOS_FORMAT=mksysb
+ export NIM_HOSTS= 192.168.6.13:sp5n13.msc.itso.ibm.com
192.168.6.1:sp5nb01.msc.itso.ibm.com  192.168.5.150:sp5en0.msc.itso.ibm.com
+ export NIM_MOUNTS=
sp5en0.msc.itso.ibm.com:/spdata/sys1/install/aix432/lppsource:/SPOT/usr/sys/inst.
images:dir
sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/pssp/13.noprompt:/NIM_BOSINST_DATA:
file
sp5nb01.msc.itso.ibm.com:/spdata/sys1/install/images/bos.obj.ssp.432:/NIM_BOS_IMA
GE:file
+ export ROUTES= 192.168.5.0:255.255.255.0:192.168.6.1
+ RC_CONFIG=/etc/rc.bos_inst
+ . /etc/rc.bos_inst
+ set -x
+ /SPOT/usr/sbin/no -o tcp_keepintvl=150
+ /SPOT/usr/sbin/no -o tcp_keepidle=1200
+ cp /usr/lpp/diagnostics/obj/CDiagDev /etc/objrepos/CDiagDev
+ cp /usr/lpp/diagnostics/obj/TMInput /etc/objrepos/TMInput
+ cp /usr/lpp/diagnostics/obj/MenuGoal /etc/objrepos/MenuGoal
+ cp /usr/lpp/diagnostics/obj/FRUB /etc/objrepos/FRUB
+ cp /usr/lpp/diagnostics/obj/FRUs /etc/objrepos/FRUs
+ cp /usr/lpp/diagnostics/obj/DAVars /etc/objrepos/DAVars
+ cp /usr/lpp/diagnostics/obj/CDiagAtt /etc/objrepos/CDiagAtt
+ cp /usr/lpp/diagnostics/obj/CDiagAtt.vc /etc/objrepos/CDiagAtt.vc
+ mkdir -p /etc/lpp/diagnostics/data
+ /usr/lib/methods/showled 0xfff
 showled + eec/usr/lp/bosinst/bi_main
```

Appendix D. 43 reasons why host responds is red

Host responds can be red for the following reasons.

D.1 General problems

1. The node is powered off.
2. The node is not connected.
3. The node has not booted.
4. PSSP is not installed.
5. /var is full.
6. There is a time difference with the Control Workstation.

D.2 Networking problems

1. The node cannot contact the Control Workstation.
2. The netmask on the node is wrong.
3. You are using the wrong port (AUI or BNC).
4. Intermediate node with ipforwarding = 0.
5. The Ethernet adapter is down.
6. There is a duplicated MAC address.

D.3 SDR problems

1. The node is not defined in the SDR.
2. The IP address is wrong in the SDR.
3. The node is defined in the wrong partition.
4. The SDR daemon is not running.
5. There is a wrong SP Port in the SDR class.
6. Reliable and Initial host name.
7. There is a duplicate entry in the Syspar class.

D.4 GUI problems

1. The hardmon daemon is not running.

2. The hmrmd resource monitor is not running.
3. The Event Manager daemon is not running.

D.5 Performance problems

1. The CPU is overloaded.
2. There is not enough memory.
3. There is network congestion.

D.6 Hardware problems

1. There is a Broken Ethernet adapter.
2. There is a Broken Ethernet cable.
3. The Ethernet card is set to full-duplex.

D.7 Software problems

1. There are incompatible software levels.
2. RSCT file sets are not installed.
3. RSCT subsystems are not configured.

D.8 Configuration problems

1. The node is not defined in the system partition.
2. An additional Ethernet adapter is not defined.
3. There is a wrong Ethernet adapter for the secondary segment.
4. There are name resolution problems.
5. There is no /etc/services for hats/hags/haem.
6. There is an incorrect instance number in hats.
7. Entries are missing from /etc/inetd.conf.

D.9 RSCT problems

1. Group Services is not running.
2. Domain is not established by hags.

3. Event Management is not running on the Control Workstation.
4. The hats subsystem does not exist.
5. The sockets are full.

Appendix E. Debugging Perl scripts

It is far more easy to use the Perl debugger to track down problems within Perl scripts than any other method such as inserting print statements to get the "idea" of the program flow. If you are used to debuggers for other programming languages, you will soon find it comfortable to use the Perl debugger. People new to debuggers all together will find this appendix a good overview of the debugging process, and they will learn just what they need in order to solve difficult setup_server and wrapper problems.

The idea here is not to use the debugger to solve any problem with setup_server or wrappers, but to use it in those situations where the error message is not clear, or the user does not have a clear idea what or where the problem is. Basically, the debugger will help you in identifying the problem, and possibly isolate it to a single component.

Instead of explaining the details of the debugger, we will use an example of a setup_server problem to demonstrate of to approach similar problems with the help of the debugger.

Figure 187 on page 433 shows the error we get when running setup_server.

```
[sp5en0:/]# setup_server
setup_server: Running services_config script to configure SSP
services.This may take a few minutes...
rc.ntp: Starting ntp daemon(xntpd)
0513-029 The supfilesrv Subsystem is already active.
Multiple instances are not supported.
/etc/auto/startauto: The automount daemon is already running on this
system.
setup_CWS: Control Workstation setup complete.
mknimmast: Node 0 (sp5en0) already configured as a NIM master.
create_krb_files: tftpaccess.ctl file and client srvtab files
created/updated
on server node 0.
mknimint: 0016-201: Incorrect number of tuples in IP address.
mknimint: 0016-201: Incorrect number of tuples in IP address.
setup_server: 0016-279 Failure of internally called command:
/usr/lpp/ssp/bin/mknimint; rc= 2.
setup_server: Processing incomplete (rc= 2).
[sp5en0:/]#
```

Figure 187. An example—setup_server problem

From the error, we see that the mknimint wrapper is the one with problems since we get the following message:

```
mknimint: 0016-201: Incorrect number of tuples in IP address.
mknimint: 0016-201: Incorrect number of tuples in IP address.
```

Now, before we start debugging the mknimint wrapper, we need to find out how the wrapper is called from setup_server. We need to know which are the arguments given to the wrapper by setup_server.

In A.7, “mknimint” on page 338, we see that the only argument accepted by this wrapper is the node number, and because we are running this command on the Control Workstation (CWS), we should assume that the argument passed on this wrappers should be node 0, which means the CWS. However, to introduce the Perl debugger, we will debug setup_server to get the argument list from the debugger.

Let us start setup_server in debug mode by running the following command:

```
perl -d /usr/lpp/ssp/bin/setup_server
```

We should add the directory where the Perl code resides so that you do not need to specify the full path every time you invoke Perl. Let us do that by running the following command:

```
export PATH=$PATH:/usr/lpp/ssp/perl5/bin
```

Now, we run the command again:

```
[sp5en0:/]# perl -d /usr/lpp/ssp/bin/setup_server

Loading DB routines from perl5db.pl version 1.0402
Emacs support available.

Enter h or `h h' for help.

main::(/usr/lpp/ssp/bin/setup_server:163):
163:    require "/usr/lpp/ssp/install/bin/install_def.pl"; # Install
definitions
  DB<1>
```

Once in the debugger, you will see the current line (the line that will be executed next). In this case, the current line is 163 in the `setup_server` script. The debugger is ready to accept your first command as indicated by the prompt DB<1>. One thing you can do here is to press **H** for help. This command will give you a list of all commands available.

Since we are just interested in debugging the problem, we will use just a hand-full of commands from the big repertoire of commands. The following is a reduce set of commands available in the debugger:

Table 14. Perl debugger commands

Command	Description
`s [expr]`	Single step [in expr]
`n [expr]`	Next, steps over subroutine calls [in expr]
`<CR>`	Repeat last `n` or `s` command
`r`	Return from current subroutine
`c [line|sub]`	Continue; optionally inserts a one-time-only breakpoint at the specified position
`w [line]`	List window around line
`b [line] [condition]`	Set breakpoint; line defaults to the current execution line; condition breaks if it evaluates to true, defaults to '1'
`b subname [condition]`	Set breakpoint at first line of subroutine
`S [[!]pattern]`	List subroutine names [not] matching pattern
`x expr`	Evals expression in array context, dumps the result
`X [vars]`	List variables [vars] in current package or subroutine

So the first thing we do is to list a window around the current line as follows:

```
DB<7> w
160     # 21 Remove authentication ticket.     /bin/rm /tmp/tkt_rcmd             #
161     #                                                                        #
162     #-----------------------------------#---------------------------------#
163==>  require "/usr/lpp/ssp/install/bin/install_def.pl"; # Install definitions
164:    require "/usr/lpp/ssp/install/bin/install_lib.pl"; # Install subroutines
165:    require "/usr/lpp/ssp/bin/sminstmsgs.pl";          # Install messages
166
167     #----------------------------------------------------------------------#
168     # Common Variables - The following variables are common (global) between  #
169     #                     main and all subroutines.  All other variables       #
  DB<7>
```

As you can see, the current line is indicated by the "==>" symbol. Since we do not want to debug every single line until we find the line where the mknimint wrapper is called, we use the following command to set up a breakpoint at that line:

```
DB<8> /mknimint/
728:      system("$MKNIMINT -l $local_node_numb");   # Make my NIM interfaces
```

So, we look for the mknimint wrapper, and then we set up the breakpoint as follows:

```
DB<9> b 728
```

Then, we let the execution roll until it hits the breakpoint:

```
DB<10> c
setup_server: Running services_config script to configure SSP services.This may take
a few minutes...
rc.ntp: Starting ntp daemon(xntpd)
0513-029 The supfilesrv Subsystem is already active.
Multiple instances are not supported.
/etc/auto/startauto: The automount daemon is already running on this system.
setup_CWS: Control Workstation setup complete.
mknimmast: Node 0 (sp5en0) already configured as a NIM master.
create_krb_files: tftpaccess.ctl file and client srvtab files created/updated
on server node 0.
main::make_interfaces(/usr/lpp/ssp/bin/setup_server:728):
728:      system("$MKNIMINT -l $local_node_numb");   # Make my NIM interfaces
  DB<10>
```

We have stopped at line 728. Here, we could list the variables to be used in the system call as follows:

```
DB<12> x "$MKNIMINT -l $local_node_numb"
0  '/usr/lpp/ssp/bin/mknimint -l 0'
```

This is how setup_server calls the mknimint wrapper for the CWS. Well, we knew this, but through the debugger, we got the exact syntax. Now, we can start debugging the mknimint wrapper directly.

Let us start the wrapper in debug mode as follows:

```
[sp5en0:/]# perl -d /usr/lpp/ssp/bin/mknimint -l 0

Loading DB routines from perl5db.pl version 1.0402
Emacs support available.

Enter h or `h h' for help.

main::(/usr/lpp/ssp/bin/mknimint:78):
78:     require "/usr/lpp/ssp/install/bin/install_def.pl";
  DB<1>
```

There are several ways we can identify the section in the wrapper where the error is produced. One of them could be to browse the script in a different window and look for the error message. Or, we can use the search facility from the debugger to look for the error message. The error message we received was:

```
mknimint: 0016-201: Incorrect number of tuples in IP address.
mknimint: 0016-201: Incorrect number of tuples in IP address.
```

So, we can search for the "tuples" word as follows:

```
DB<1> /tuples/
374:                          # normalize the ipaddr (put leading zeros in tuples)

  DB<2> /tuples/
646:                          # normalize the ipaddr (put leading zeros in tuples)

  DB<3> /tuples/
374:                          # normalize the ipaddr (put leading zeros in tuples)
```

As you can see, there are only two lines where the word tuples is used. So, the next step is to display a window around those lines as follows:

```
DB<4> w 374
371:      foreach(@netstat){
372                           # we need the interface (e.g. en0) and ipaddr from
netstat
373:            ($interface, $junk1, $junk2, $ipaddr, $rest) = split(/\s+/);
374                           # normalize the ipaddr (put leading zeros in tuples)
375:            $normal_ip=&ipfill($ipaddr);
376                           # if normalizing failed, bail out
377:            if ($normal_ip == 0) {
378:                    return $NOK;
379             }
380
  DB<5> w 646
643                      }
644             }
645
646                           # normalize the ipaddr (put leading zeros in tuples)
647:            $normal_ip=&ipfill($ipaddr);
648                           # if normalizing failed, bail out
649:            if ($normal_ip == 0) {
650:                    return $NOK;
651             }
652
```

Then, we set up a breakpoint in each line as follows:

```
DB<6> b 373
DB<7> b 647
```

Finally, we let wrapper run into the first breakpoint as follows:

```
DB<8> c
main::find_up_enets(/usr/lpp/ssp/bin/mknimint:373):
373:            ($interface, $junk1, $junk2, $ipaddr, $rest) = split(/\s+/);
DB<8> w
370:      chop(@netstat=grep(!/\*/,(grep(/en/,grep(!/ink/,@RUNITRC_OUT)))));
371:      foreach(@netstat){
372                           # we need the interface (e.g. en0) and ipaddr from
netstat
373==>b         ($interface, $junk1, $junk2, $ipaddr, $rest) = split(/\s+/);
374                           # normalize the ipaddr (put leading zeros in tuples)
375:            $normal_ip=&ipfill($ipaddr);
376                           # if normalizing failed, bail out
377:            if ($normal_ip == 0) {
378:                    return $NOK;
379             }
```

The first command was `c` to continue the execution in normal mode. The execution stopped at the first breakpoint found at line 373. Then, we display a window (command `w`) around the current line.

Now that we are in the location we "think" the error is located, we proceed with step-by-step execution using the `n` command, which steps over subroutines. We get the following output:

```
DB<8> n
main::find_up_enets(/usr/lpp/ssp/bin/mknimint:375):
375:            $normal_ip=&ipfill($ipaddr);
  DB<8> n
main::find_up_enets(/usr/lpp/ssp/bin/mknimint:377):
377:            if ($normal_ip == 0) {
  DB<8> n
main::find_up_enets(/usr/lpp/ssp/bin/mknimint:382):
382:            if($rc=&runitrc($host_name, "$LSATTR -E -l $interface -a netmask -F
value 2> /dev/null")){
  DB<8> n
main::find_up_enets(/usr/lpp/ssp/bin/mknimint:390):
390:            chop($netmask=@RUNITRC_OUT[0]);
  DB<8> n
main::find_up_enets(/usr/lpp/ssp/bin/mknimint:391):
391:            $netmask=~s/$host_name: //;
  DB<8> n
main::find_up_enets(/usr/lpp/ssp/bin/mknimint:392):
392:            $effective_ip=&ipeffaddr($normal_ip,&ipfill($netmask));
  DB<8> n
mknimint: 0016-201: Incorrect number of tuples in IP address.
mknimint: 0016-201: Incorrect number of tuples in IP address.
main::find_up_enets(/usr/lpp/ssp/bin/mknimint:393):
393:            if ($effective_ip == 0) {
```

As you can see, we get the error message after the execution of line 392. So, let us take a look at the variables used by the ipeffaddr() subroutine. Issue the following:

```
DB<9> x $normal_ip
0  '192.168.005.150'
  DB<10> x $netmask
0  ''
```

So, the netmask variable is empty. From the step-by-step execution, we see that the netmask variable was taken from the output of the `lsattr` command as follows:

```
DB<8> n
main::find_up_enets(/usr/lpp/ssp/bin/mknimint:382):
382:            if($rc=&runitrc($host_name, "$LSATTR -E -l $interface -a netmask -F
value 2> /dev/null")){
  DB<8> n
main::find_up_enets(/usr/lpp/ssp/bin/mknimint:390):
390:            chop($netmask=@RUNITRC_OUT[0]);
```

So, let us check the `lsattr` output provided by the RUNITRC_OUT array as follows:

```
DB<12> x @RUNITRC_OUT
0  'sp5en0:
'
```

This means that the output of the `lsattr` command was null. Let us run the command on another window and check out the output as follows:

```
[sp5en0:/]# lsattr -E -l en0 -a netmask -F value

[sp5en0:/]# lsattr -E -l en0
mtu           1500          Maximum IP Packet Size for This Device     True
remmtu        576           Maximum IP Packet Size for REMOTE Networks True
netaddr       192.168.5.150 Internet Address                           True
state         up            Current Interface Status                   True
arp           on            Address Resolution Protocol (ARP)          True
netmask                     Subnet Mask                                True
security      none          Security Level                             True
authority                   Authorized Users                           True
broadcast                   Broadcast Address                          True
netaddr6                    N/A                                        True
alias6                      N/A                                        True
prefixlen                   N/A                                        True
alias4                      N/A                                        True
rfc1323                     N/A                                        True
tcp_nodelay                 N/A                                        True
tcp_sendspace               N/A                                        True
tcp_recvspace               N/A                                        True
tcp_mssdflt                 N/A                                        True
```

The netmask attribute in the ODM is empty. We have found the problem.

When the netmask value in the ODM is empty, the TCP/IP interface takes the default mask for the address class; so, all TCP/IP applications will work fine as long as the default netmask is the correct value for the network topology.

As you can see, a very interesting and complicated problem from the outside. However, the Perl debugger gave us the key to isolate and find this error.

Appendix F. 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 Parallel System Support Programs (PSSP) Version 3, Release 1 projects, including the education of RS/60000 SP professionals responsible for installing, configuring, and administering PSSP Version 3, Release 1. The information in this publication is not intended as the specification of any programming interfaces that are provided by Parallel System Support Programs. See the PUBLICATIONS section of the IBM Programming Announcement for PSSP Version 3, Release 1 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, North Castle Drive, Armonk, NY 10504-1785.

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

AIX	AS/400
AT	BookManager
CT	ESCON
Global Network	HACMP/6000
IBM ®	LoadLeveler
MQ	Netfinity
OS/390	POWERparallel
RS/6000	S/390
SP	System/390
400	

Appendix G. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

G.1 IBM Redbooks publications

For information on ordering these publications see “How to get IBM Redbooks” on page 447.

- *Inside the RS/6000 SP*, SG24-5145
- *PSSP 2.4 Technical Presentation*, SG24-5173
- *RS/6000 SP High Availability Infrastructure*, SG24-4838
- *RS/6000 SP: Problem Determination Guide*, SG24-4778
- *RS/6000 SP PSSP 2.2 Survival Guide*, SG24-4928
- *RS/6000 SP PSSP 2.2 Technical Presentation*, SG24-4868
- *SP Perspectives: A New View of You SP System*, SG24-5180
- *Understanding and Using the SP Switch*, SG24-5161
- *The RS/6000 SP Inside Out*, SG24-5374
- *HACMP Enhanced Scalability Handbook*, SG24-5328

G.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at `http://www.redbooks.ibm.com/` for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	**Collection Kit Number**
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037

CD-ROM Title	**Collection Kit Number**
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

G.3 Other resources

These publications are also relevant as further information sources:

- *AIX V4.3 Problem Solving Guide and Reference*, SC23-4123
- *IBM RS/6000 SP: Planning, Volume 2, Control Workstation and Software Environment*, GA22-7281
- *PSSP: Administration Guide*, SA22-7348
- *PSSP: Command and Technical Reference, Volume 1* and *Volume 2*, SA22-7351
- *PSSP: Diagnosis Guide*, GA22-7350
- *PSSP: Installation and Migration Guide*, GA22-7347
- *PSSP: Managing Shared Disks*, SA22-7349
- *PSSP: Messages Reference*, GA22-7352
- *RS/6000 SP: Planning, Volume 1, Hardware and Physical Environment*, GA22-7280
- *RSCT: Event Management Programming Guide and Reference*, SA22-7354
- *RSCT: Group Services Programming Guide and Reference*, SA22-7355
- *Technical Presentation for PSSP 2.3*, SG24-2080 (Available online)

G.4 Referenced Web site

This Web site is also relevant as a further information source:

- `http://www.rs6000.ibm.com/resource/aix_resource/sp_books/pssp/index.htm`

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** http://www.redbooks.ibm.com/

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax 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.

List of abbreviations

AIX	Advanced Interactive Executive
AMG	Adapter Membership Group
ANS	Abstract Notation Syntax
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
DB	Database
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
FS	File System
GB	Gigabytes
GL	Group Leader
GPFS	General Purposes File System
GS	Group Services
GSAPI	Group Services Application Programming Interface
GVG	Global Volume Group
HACMP	High Availability Cluster Multiprocessing
HACMP/ES	High Availability Cluster Multiprocessing Enhanced Scalability
hb	Heart Beat
HiPS	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	Journaled File System
LAN	Local Area Network
LCD	Liquid Crystal Display
LED	Light Emitter Diode
LP	Logical Partition
LRU	Last Recently Used
LSC	Link Switch Chip
LV	Logical Volume
LVM	Logical Volume Manager
MB	Megabytes

MIB	Management Information Base
MPI	Message Passing Interface
MPL	Message Passing Library
MPP	Massive Parallel Processors
NFS	Network File System
NIM	Network Installation Management
NSB	Node Switch Board
NSC	Node Switch Chip
OID	Object ID
ODM	Object Data Management
PAIDE	Performance Aide for AIX
PE	Parallel Environment
PID	Process ID
PP	Physical Partition
PSSP	Parallel System Support Programs
PTC	Prepare To Commit
PTPE	Performance Toolbox Parallel Extensions
PTX	Performance Toolbox for AIX
PV	Physical Volume
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
R/VSD	Recoverable/Virtual Shared Disk
RVSD	Recoverable Virtual Shared Disk
SBS	Structure Byte String
SCSI	Small Computer System Interface
SDR	System Data Repository
SMIT	System Management Interface Tool
SSA	Serial Storage Architecture
VG	Volume Group
VSD	Virtual Shared Disk

Index

IBM Redbooks evaluation

PSSP Version 3 Survival Guide
SG24-5344-00

Your feedback is very important to help us maintain the quality of IBM Redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
_ Customer _ Business Partner _ Solution Developer _ IBM employee
_ None of the above

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!)

SG24-5344-00
Printed in the U.S.A.

PSSP Version 3 Survival Guide

SG24-5344-00