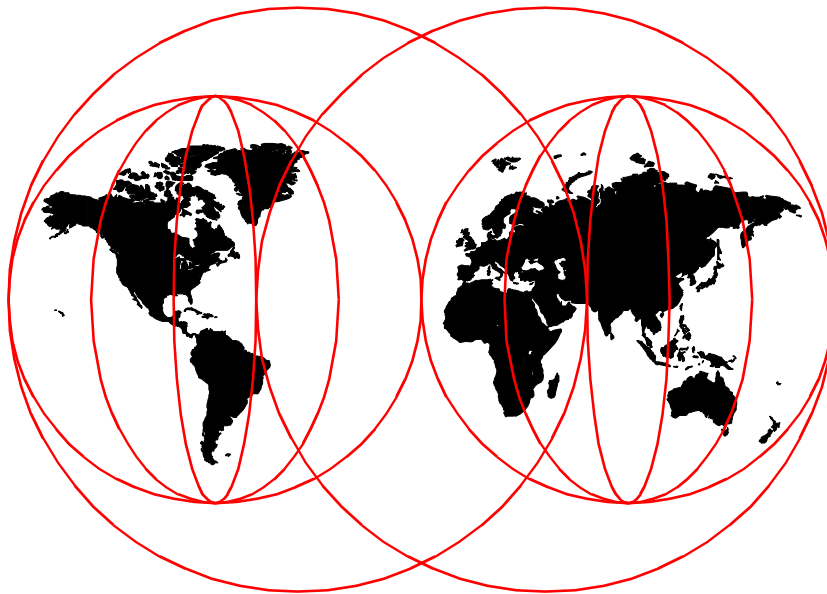




The RS/6000 SP Inside Out

*Marcelo R. Barrios, Mario Bono, Michael Hennecke
Teerachai Supapunpinyo, Bernard Woo, Steven Yap*



International Technical Support Organization

<http://www.redbooks.ibm.com>

SG24-5374-00

high availability. For each S70 attachment two RS-232 lines are required. For each Netfinity server one RS-232 line is required. Each of these RS-232 lines must be configured.

The transmit queue size for the SP administrative ethernet has to be tuned accordingly for better performance. Refer to Table 12 for the various settings.

Table 12. Adapter Type and Transmit Queue Size Settings

Adapter Type	Transmit Queue Size
PCI adapters	256
MCA adapters AIX 4.2.0 or earlier	Maximum possible
MCA adapters AIX 4.2.1 or later	512

After changing the transmit queue size for the appropriate adapter, you can proceed to configure the SP administrative ethernet IP address based on your planning sheet.

The maximum number of processes allowed per user must be increased from the default value of 40 to a recommended value of 256. This is done to accommodate the numerous processes spawned off during the installation.

The default network options have to be tuned for optimal performance over the network. The recommended values are stated in Table 13. To make the changes effective every time the control workstation is rebooted, these values must be set in the /etc/rc.net file. For immediate effect, use the command line. For example, to change thewall value, type:

```
# no -o thewall=16384
```

Table 13. Recommended Network Option Tunables

Parameters	Values
thewall	16384
sb_max	163840
ipforwarding	1
tcp_sendspace	65536
tcp_recvspace	65536
udp_sendspace	32768
udp_recvspace	65536
tcp_mssdflt	1448

A directory (preferably a file system) /tftpboot has to be created to store all the NIM boot images. Its size should be about 25 MB for each AIX level supported on the nodes. Having too little storage space in this directory will cause the `setup_server` command to fail.

Next comes the /spdata directory. We recommend that this directory be created as a file system on a volume group of its own because of the amount of disk space it requires. A minimum of 2 GB disk space is required to support one lppsource. Figure 108 shows the directory structure for /spdata.

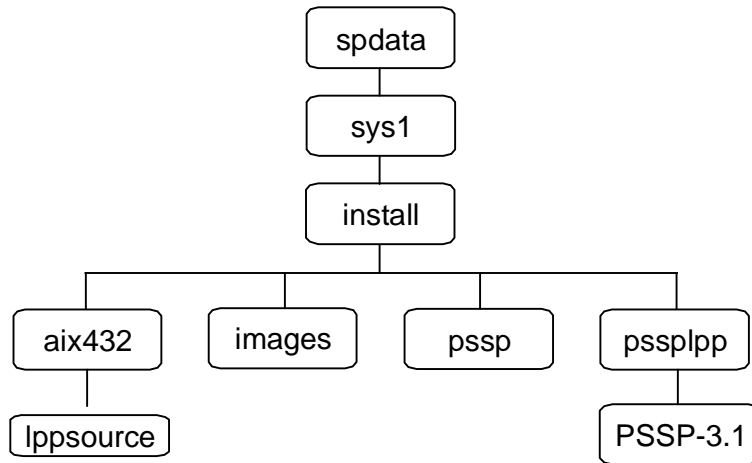


Figure 108. /spdata Directory Structure

Each one of the directories must be created. If you have to install nodes of other AIX levels, you must create the relevant lppsource directories under their own AIX directories. Here, only the aix432 directory and its related lppsource directory are created. If the other AIX levels require different PSSP codes, then the relevant PSSP directories must be created under the pssplpp directory. This shown in Figure 109 on page 265.

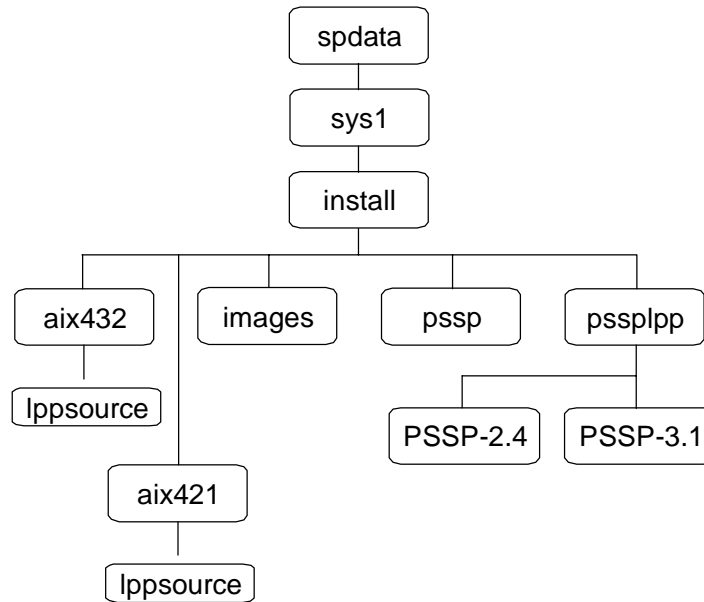


Figure 109. Multiple AIX And PSSP Levels

The `/spdata/sys1/install/aix432/lppsource` directory contains all the necessary AIX file sets for node installation. We recommend that you download all the AIX file sets. You must use the `bffcreate` command to copy these file sets from the source CDs or tapes to this directory. The command to copy from an AIX CD is as follows:

```
# /usr/sbin/bffcreate -v -d /dev/cd0 -t \
/spdata/sys1/install/aix432/lppsource -X all
```

The SMIT fastpath is:

```
# smitty bffcreate
```

Also required in the `lppsource` directory are the `perfagent.*` file sets. The `perfagent.server` file sets are part of the Performance Aide for AIX (PAIDE) feature of the Performance Toolbox for AIX (PTX). The `perfagent.tools` file sets are part of the AIX packaging. See Figure 11 on page 262 for the correct level for your installation.

Copy the `mksysb` image that you want to use for installing your nodes into the `/spdata/sys1/install/images` directory. Each `mksysb` must have an equivalent AIX `lppsource` to support it. You can use the minimum image that is shipped with the RS/6000 SP system, or you can create your own `mksysb` image. By

creating your own, you can install nodes that have the same file sets requirement instead of individually loading each one of them with the same file sets after the mksysb installation. The recommended method is to install a node with the default image and then install all the other required file sets. Next, take a mksysb image of that node and install the rest of the nodes that have the same requirements. You can do likewise for other groups of nodes with other specific needs.

To download the minimum image from the tape media, type:

```
# installp -a -d /dev/rmt0.1 -X spimg
```

Next, copy the PSSP images from the product tape into the /spdata/sys1/install/pssplpp/PSSP-3.1 directory:

```
# bffcreate -d /dev/rmt0 -t /spdata/sys1/install/pssplpp/PSSP-3.1 -X all
```

The ssp.usr.3.1.0.0 file set must be renamed to pssp.installp. After renaming the file, a new table of contents has to be generated as follows:

```
# mv /spdata/sys1/install/pssplpp/PSSP-3.1/ssp.usr.3.1.0.0  
/spdata/sys1/install/pssplpp/PSSP-3.1/pssp.installp
```

```
# inutoc /spdata/sys1/install/pssplpp/PSSP-3.1
```

10.2.3 Setting Up The PSSP Environment

After installing AIX and preparing its environment, the PSSP software is installed. On the control workstation, the minimum required file sets are:

- rsct.* (all rsct file sets)
- ssp.basic
- ssp.authent (if control workstation is Kerberos authentication server)
- ssp.clients
- ssp.css (if SP switch is installed)
- ssp.top (if SP switch is installed)
- ssp.ha_topsvcs.compat
- ssp.perlpkg
- ssp.sysctl
- ssp.sysman

Attention

If you are adding dependent nodes, you must also install the ssp.spmgr fileset.

The ssp.docs and ssp.resctr.rte file sets are recommended as part of the installed file sets. The ssp.docs file set provides online PSSP publications while the ssp.resctr.rte file set provides the front end interface to all online documentation and resources including SP-related Web site links.

The authentication services you want to implement on your system include RS/6000 SP, AFS, or MIT Kerberos Version 4. You can choose to implement one of them. The `/usr/lpp/ssp/bin/setup_authent` script will initialize the authentication services.

To finish off with the installation of the control workstation, the `install_cw` script is run. This command configures the control workstation as follows:

- It adds the PSSP SMIT panels to the ODM.
- It creates an SP object in the ODM which contains the `node_number` for the control workstation, which is always 0.
- It calls the script `/usr/lpp/ssp/install/bin/post_process` to do post-installation tasks such as:
 - Starting the `hardmon` daemon, SDR daemons
 - Calling the `setup_logd` script to start the monitor's logging daemon
 - Updating the `/etc/services` and `/etc/inittab` files.
 - It also ensures that `/tftpboot/tuning.cust` exists. If the file does not exist, it copies from `/usr/lpp/ssp/install/config/tuning.default`.
 - Finally, it sets the authentication server attribute in the SDR to reflect the kind of authentication server environment.

To verify that the SDR is properly installed, type:

```
# SDR_test
```

To verify if system monitor is properly installed, type:

```
# spmon_itest
```

10.3 Frame, Node And Switch Installation

After completing the control workstation installation and set up, the next step is to configure the frames, nodes and switches.

10.3.1 Site Environment And Frame Information

SDR information can only be entered on the control workstation. You can use the SMIT panels or the command line. If you are already using the SP Perspectives, there are icons which will lead you to the SMIT panels. You are

advised to use the SMIT panels, as they are less prone to errors and more user-friendly. The SMIT fastpath is:

```
# smitty enter_data
```

The first task is to set up the Site Environment Information. Select **Site Environment Information** on the SMIT menu. Figure 110 shows this SMIT screen. Enter the following information:

- Default network install image name (directory path not required)
- NTP information
- Automounter option
- User administration information
- File collection information
- SP accounting information
- AIX lppsource name (directory path not required)

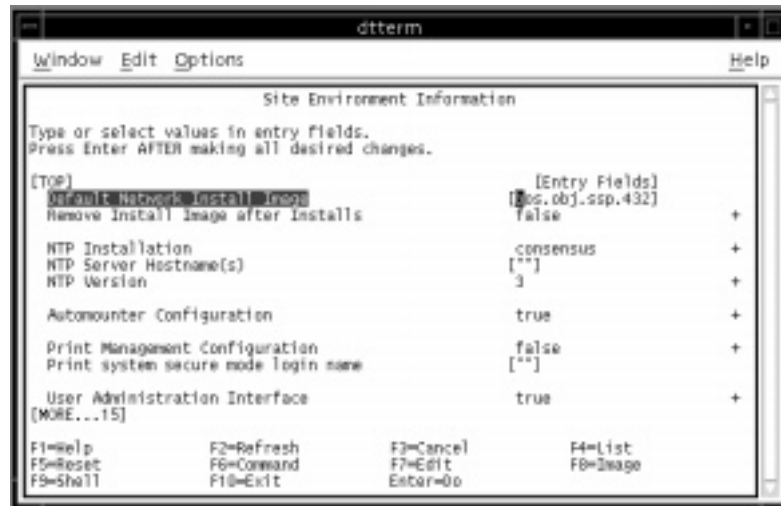


Figure 110. Site Environment Information SMIT Panel

Next, enter the SP Frame Information as shown in Figure 111 on page 269. In this panel, specify the starting frame number, the frame count (the number of consecutive frames), the starting frame tty port and whether you want to re-initialize the SDR. Do not enter non-SP frames in this panel. The re-initialization of the SDR is required only when the last set of frames information is entered. Take note that if your tty numbers do not run consecutively, you need to enter the frames information in separate sessions.



Figure 111. SP Frame Information SMIT Panel

If you have non-SP frames attached, enter their information in the Non-SP Frame Information SMIT panel as shown in Figure 112 on page 270. If the non-SP frame is an S70 or S7A, make sure that the starting frame tty port you specify is the one connected to the operator panel on the server. The s1 tty port field is the tty connected to the serial port on the server. The starting switch port number is the switch node number. The frame hardware protocol is where you specify if the non-SP frame is an S70/S7A server or a Netfinity server. For an S70/S7A server, select SAMI; for a Netfinity server, select SLIM. Re-initialize the SDR if the information you entered is for the last set of frames.

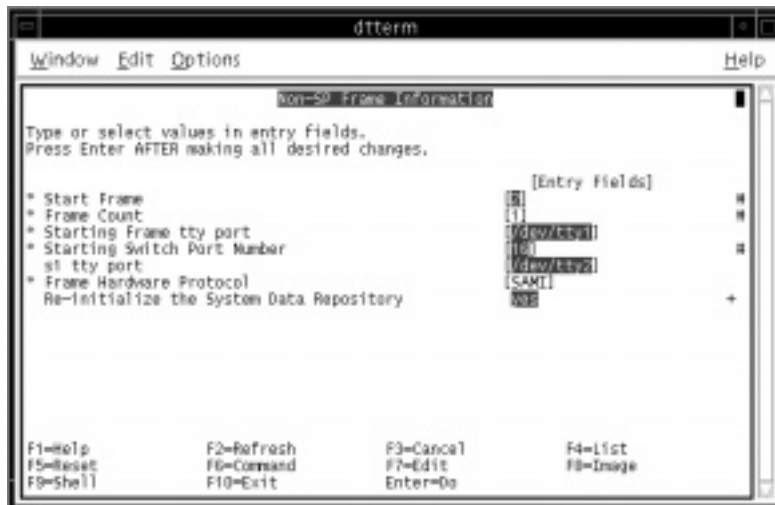


Figure 112. Non-SP Frame Information SMIT Panel

At this point, your frames must already be powered up. Verify that the System Monitor and Perspectives have been correctly installed. The SMIT fastpath is:

```
# smitty SP_verify
```

Select the System Monitor Configuration option. Figure 113 on page 271 shows the output if everything is installed properly. If it fails, check that your tty configuration is correct. Also, ensure that the physical RS-232 lines are properly connected. Delete the frame information and reconfigure again. Make sure that you do not remove or change the RS-232 connections before you delete the frame or you will end up having to hack the SDR to salvage the situation. Other causes of error include Kerberos authentication problems, SDR or hardmon daemons not running and so on. Rectify the errors before proceeding.

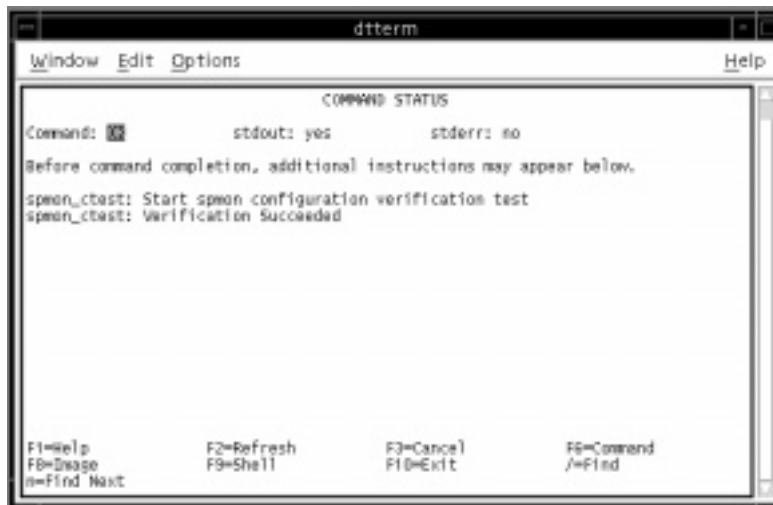


Figure 113. Successful Verification With spmon_ctest

Verify the frame information using the following command:

```
# spmon -d
```

If any error is found, check the RS-232 cables. You should not have this problem, though, as the previous test will have detected it.

Next, verify the supervisor microcode of your system. The SMIT fastpath is:

```
# smitty supervisor
```

Figure 114 on page 272 shows the result of selecting the List Status of Supervisors (Report Form) option. Any item that has the status Upgrade needs the microcode updated. Choose the **Update *ALL* Supervisors That Require Action (Use Most Current Level)** option to update microcode on all the necessary supervisor cards.

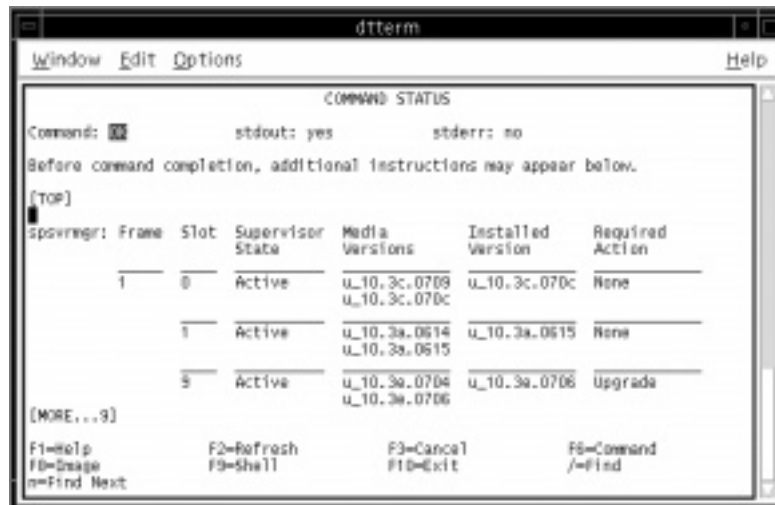


Figure 114. Supervisor Microcode Status

10.3.2 Node Database Information

The next information required is the node database information. The fastpath for this SMIT menu is:

```
# smitty node_data
```

Select the SP Ethernet Information. Provide the information required: the node you are installing, the starting node's en0 host name or IP address, the netmask, default route hostname or IP address, the type of ethernet and its characteristics and whether you want to skip IP address for unused slots. When specifying the starting node's en0 hostname, make sure the hostname is fully qualified. To be on the safe side, use the IP address. Ensure all IP addresses and hostnames are resolvable. You can issue the following command to ensure the correct hostname is specified:

```
# host <node's en0 IP address>
```

If you intend to use the node's slot number as a reference for assigning an IP address, set the Skip IP Address for Unused Slots to yes. This will skip the next sequential IP addresses when it encounters Wide or High nodes. Figure 115 on page 273 shows the SMIT panel available for setting up SP Ethernet addresses.

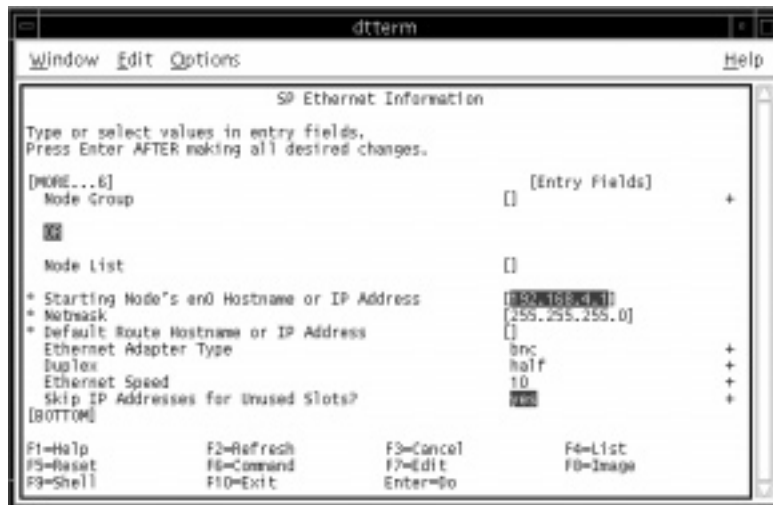


Figure 115. Setting Up The SP Ethernet Information

In order to establish bootp communication between the control workstation and the nodes, the hardware addresses of the nodes must be made known to the control workstation in the /etc/bootptab file. Select the **Get Hardware Ethernet Addresses** menu. Figure 116 shows the SMIT panel available for getting Ethernet hardware addresses.

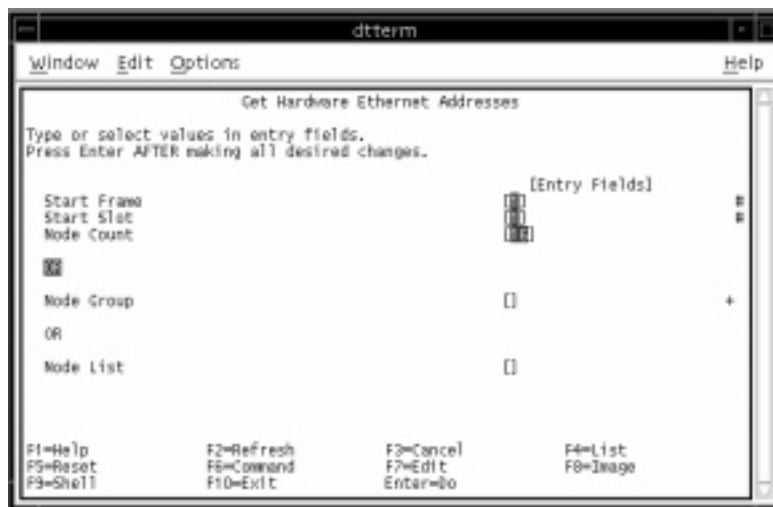


Figure 116. Acquiring The SP Ethernet Hardware Addresses

However, if you already have the hardware addresses available, enter them into the `/etc/bootptab.info` file. This speeds up the installation process since acquiring the hardware addresses can take up a significant amount of time. We recommend that you let the system acquire the hardware addresses. The small amount of time spent doing so will probably save time later by avoiding corrections if the wrong hardware addresses are specified in the `/etc/bootptab.info` file. The format of the `/etc/bootptab.info` file is:

```
<node number> <en0 hardware address>  
<node number> <en0 hardware address>
```

Attention

Acquiring hardware addresses for nodes will power those nodes down. Do not use this step on nodes running in a production environment.

If you have a switch or other network adapters (ethernet, FDDI, token ring) in your system and you want them to be configured during installation, you need to perform this step. To do so, select the Additional Adapter Information menu. To configure the SP Switch adapter, use `css0` for the Adapter Name. Specify the IP addresses and netmask just like for the `en0` adapter.

Attention

- To skip IP addresses for unused slots, do not use the switch node numbers for `css0` IP addresses.
- If you do not use switch node numbers for `css0` IP addresses, you must ensure ARP is enabled.

In addition, you can specify additional IP addresses for the adapter if you have IP aliasing. Figure 117 on page 275 shows the SMIT panel available for setting up additional network adapters.

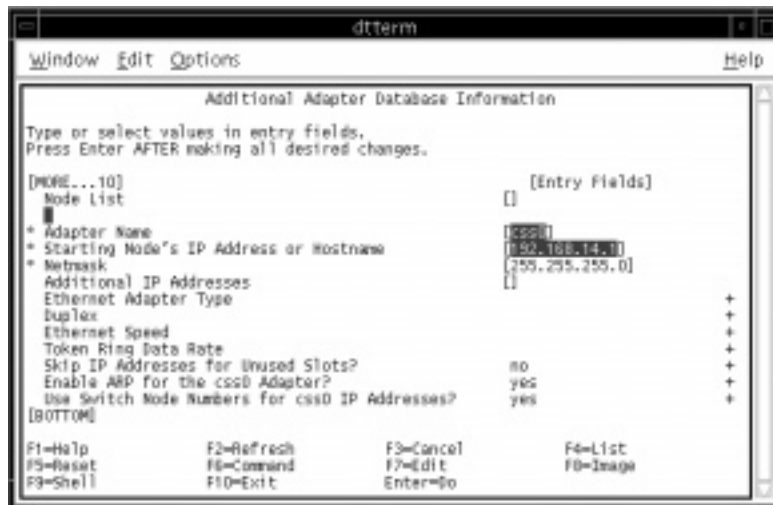


Figure 117. Setting Up Additional Adapter Information

Next, configure the default hostname of the nodes. By default, the en0's long hostname is used. You can use another adapter's hostname or change to short hostname. Figure 118 shows the SMIT panel for setting up hostnames.

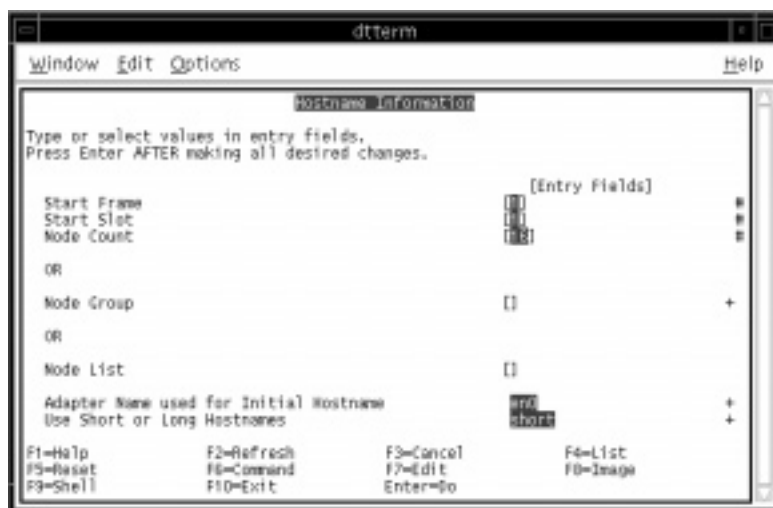


Figure 118. Changing Default Hostname Information

The next step creates the appropriate authorization files for the use of remote commands. The available methods are k4 (Kerberos Version 4) and std

(Standard AIX). k4 must be selected if std is an optional choice. We recommend that you enable all if you are not sure. The SMIT fastpath is:

```
# smitty spauth_config
```

Go to the Select Authorization Methods menu. Indicate the system partition name and the methods. Figure 119 shows the SMIT panel for setting up authorization methods.



Figure 119. Selecting The Authorization Method

Next, enable the selected authentication methods for use in conjunction with System Management tasks. The default is to enable authentication on all nodes as well as the control workstation if the Force change on nodes option is set to yes, it will force the authentication method to change to whatever you set, even though the information in the node may be the same. The available methods are k5 (Kerberos Version 5), k4 and std; k4 is required while k5 and std are optional. If you intend to use ftp, rlogin and telnet commands, k5 or std must be enabled as well. It is recommended that you enable all. Figure 120 on page 277 shows the SMIT panel available for enabling the authentication methods.



Figure 120. Enabling Authentication Methods

If you have a dependent node, add it at this point. First, ensure that you have the `ssp.spmgr` files set installed and that the UDP port 162 it uses for SNMP traffic does not clash with other applications that use the same port. If a conflict arises, modify the `spmgrd-trap` entry in `/etc/services` to use another port number.

To define the dependent node, use the SMIT fastpath:

```
# smitty enter_extnode
```

The required entries are administrative hostname, SNMP community name, extension node identifier, resolvable SNMP agent hostname, unused node number. Keep the administrative hostname and SNMP agent hostname the same to avoid confusion. Figure 121 on page 278 shows the SMIT panel available for defining dependent node information.



Figure 121. Defining Dependent Node Information

After adding the dependent node information, each node adapter for the dependent node must be defined. The SMIT fastpath is:

```
# smitty enter_extadapter
```

Specify the network address, netmask and the node number as shown in Figure 122 on page 278.



Figure 122. Defining Dependent Node Adapter Information

On the dependent node, you can choose to define the dependent node and dependent node adapter. You must configure the SNMP agent on the dependent node to communicate with the dependent node SNMP manager on the control workstation. Next, install the SP Switch Router and all IP interfaces connected to it.

At this point, the system partition-sensitive subsystems like hats, hags, haem and hr must be added and started. To do so, issue the following command:

```
# syspar_ctrl -A
```

Verify that all the system partition-sensitive subsystems have been properly added by issuing the following command:

```
# syspar_ctrl -E
```

Check that the subsystems are active with the following command:

```
# lssrc -a | grep <partition_name>
```

You may have to wait a few minutes before some of the subsystems become active.

The volume group information is required next. Go back to the Node Database Information SMIT menu and select the **Create Volume Group Information** option. In this screen, specify the node you are installing, the volume group name, the target disk, whether you want mirroring, the boot/install server, the network install image name, the lppsource name, the PSSP code version and whether you want to turn on quorum. Make sure that the image you use corresponds to the correct lppsource name. Mismatch of AIX levels can cause unpredictable results. Ensure also the PSSP level is supported for that level of AIX. Figure 123 on page 280 shows the SMIT panel for defining alternate volume groups.

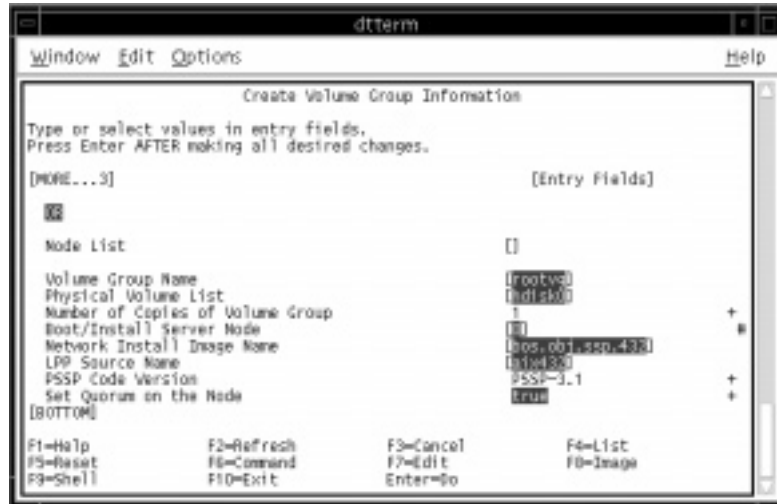


Figure 123. Setting Up Volume Group Information

With PSSP 3.1, you can install the rootvg on external SSA or SCSI disks. Refer to Table 14 for a list of supported SSA boot devices and Table 15 on page 281 for a list of supported SCSI boot devices.

Table 14. Supported External SSA Boot Devices

Node Type	External Boot Supported?	SSA Adapter Feature Codes			
		6214	6216	6217	6219
62 MHz Thin	N				
66 MHz Wide	N				
66 MHz Thin 2	N				
77 MHz Wide	Y	X	X	X	X
112 MHz 604 High	Y	X	X	X	X
120 MHz Thin	Y	X	X	X	X
135 MHz Wide	Y	X	X	X	X
200 MHz 604e High	Y	X	X	X	X
160 MHz Thin	Y	X	X	X	X
332 MHz SMP Thin	N				
332 MHz SMP Wide	N				

Node Type	External Boot Supported?	SSA Adapter Feature Codes			
		6214	6216	6217	6219
POWER3 Thin	N				
POWER3 Wide	N				

The supported external SSA subsystems are:

- 7133-010
- 7133-020
- 7133-500
- 7133-600

Table 15. Supported External SCSI Boot Devices

Node Type	External Boot Supported?	SCSI Adapter Feature Codes			
		2412	2416	6207	6209
62 MHz Thin	Y	X	X		
66 MHz Wide	Y	X	X		
66 MHz Thin 2	Y	X	X		
77 MHz Wide	Y	X	X		
112 MHz 604 High	Y	X	X		
120 MHz Thin	Y	X	X		
135 MHz Wide	Y	X	X		
200 MHz 604e High	Y	X	X		
160 MHz Thin	Y	X	X		
332 MHz SMP Thin	Y			X	X
332 MHz SMP Wide	Y			X	X
200 MHz POWER3 Thin	Y			X	X
200 MHz POWER3 Wide	Y			X	X

The supported external SCSI disk subsystems are:

- 7027-HSD IBM High Capacity Drawer
- 7131-105 SCSI Multi-Storage Tower

When specifying the Physical Volume List (target disk for rootvg), three types of format can be used. The first format specifies the device name such as hdisk0 or hdisk1. The second format specifies the hardware location of the disk (SCSI only), for example 00-00-00-0,0. When specifying more than one disk using this format, separate the location addresses by colons:

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

The third format specifies the parent-connwhere attribute (SSA only), for example ssar//0004AC5052B500D. To indicate more than one disk, separate the disks by colons:

```
ssar//0004AC5052B500D:ssar//0004AC5150BA00D
```

You can always go back to make changes to this volume group by selecting the **Change Volume Group Information** option.

Before continuing with the installation, perform a check on all the information you have entered into the SDR using the `splstdata.` command.

To list the site environment settings, use the following command:

```
# splstdata -e
List Site Environment Database Information

attribute          value
-----
control_workstation  sp4en0
cw_ipaddrs          9.12.0.4:192.168.4.140:
install_image        bos.obj.ssp.432
remove_image         false
primary_node         1
ntp_config           consensus
ntp_server           ""
ntp_version          3
amd_config           false
print_config         false
print_id             ""
usermgmt_config      true
passwd_file          /etc/passwd
passwd_file_loc      sp4en0
homedir_server       sp4en0
homedir_path         /home/sp4en0
filecoll_config      true
supman_uid           102
supfilesrv_port      8431
spacct_enable        true
spacct_actnode_thresh 80
spacct_excluse_enable false
```

```

acct_master          0
cw_has_usr_clients  false
code_version         PSSP-3.1
layout_dir           ""
authent_server       ssp
backup_cw            ""
ipaddrs_bucw        ""
active_cw            ""
sec_master           ""
cds_server            ""
cell_name            ""
cw_lppsource_name    aix432
cw_dcehostname       ""

```

To list the frame information, use the following command:

```
# splstdata -f
```

To list the node information, use the following command:

```
# splstdata -n
```

To list the adapter information, use the following command:

```
# splstdata -a
```

To list the boot/install information, use the following command:

```
# splstdata -b
```

To list the switch information, use the following command:

```
# splstdata -s
```

To list the node information for dependent nodes use the following command:

```
# splstnodes -t dependent node_number reliable_hostname \
management_agent_hostname extension_node_identifier snmp_community_name
```

To list the node adapter information, use the following command:

```
# splstadapters -t dependent node_number netaddr netmask
```

You can customize your nodes during the installation process. There are three files where you can specify your customization requirements. The three files are:

- **tuning.cust** - It is used to set the initial network tuning parameters. This file is called by `pssp_script` after installing the node. It must be placed in the `/tftpboot` directory for it to be effective. If during the installation process, the boot/install server cannot locate this file, the default

`/usr/lpp/ssp/install/config/tuning.default` file is copied to `/tftpboot/tuning.cust`. To help you with an initial setting, IBM provides three sets of values for three types of environment:

- `/usr/lpp/ssp/install/config/tuning.commercial` for typical commercial environment.
- `/usr/lpp/ssp/install/tuning.scientific` for typical engineering/scientific environment. This file must not be used for tuning the S70/S7A servers.
- `/usr/lpp/ssp/install/config/tuning.development` for development environment.
- **script.cust** - This file is also called by `pssp_script` just after the node is installed but before it reboots for the first time. This script is run in a limited environment, so many functions are not available yet. You can copy the `/usr/lpp/ssp/samples/script.cust` to `/tftpboot` as an initial reference.
- **firstboot.cust** - This file is called the first time a node is rebooted after a network installation. A sample copy of this file resides in `/usr/lpp/ssp/samples/firstboot.cust`. Copy this sample file to `/tftpboot/firstboot.cust` and modify the contents to suite your requirement.

The next step will configure the control workstation as a boot/install server. Prior to performing this step, ensure that the `/usr` file system or its related directories are not NFS-exported. The `/spdata/sys1/install/images` directory also must not be NFS-exported.

The command to set up the control workstation as a boot/install server is `setup_server`. The first time `setup_server` is run, it takes a considerable amount of time.

The `setup_server` Perl script is run on the control workstation when explicitly called, and on every node when they reboot. On the control workstation, or on a node set to be a boot/install server (stated in the SDR), this script configures it as a boot/install server. This command requires a ticket-granting-ticket to run. It performs the following functions:

- Defines the boot/install server as a Network Installation Management (NIM) master
- Defines the resources needed for the NIM clients
- Defines each node that this server installs as a NIM client
- Allocates the NIM resources necessary for each NIM client
- Creates the `node.install_info` file containing `netinstall` information
- Creates the `node.config_info` file containing node-specific configuration information be used during network boot

- Creates server key files containing the service keys for the nodes
- Copies the install images from the control workstation for nodes which are boot/install servers

Creation of the Network Installation Management (NIM) lppsource resource on a boot/install server will result in `setup_server` creating a lock in the lppsource directory on the control workstation. This lock is created by the `mknimres` command when the `setup_server` command calls it.

The general flow for `setup_server` is listed in Figure 124

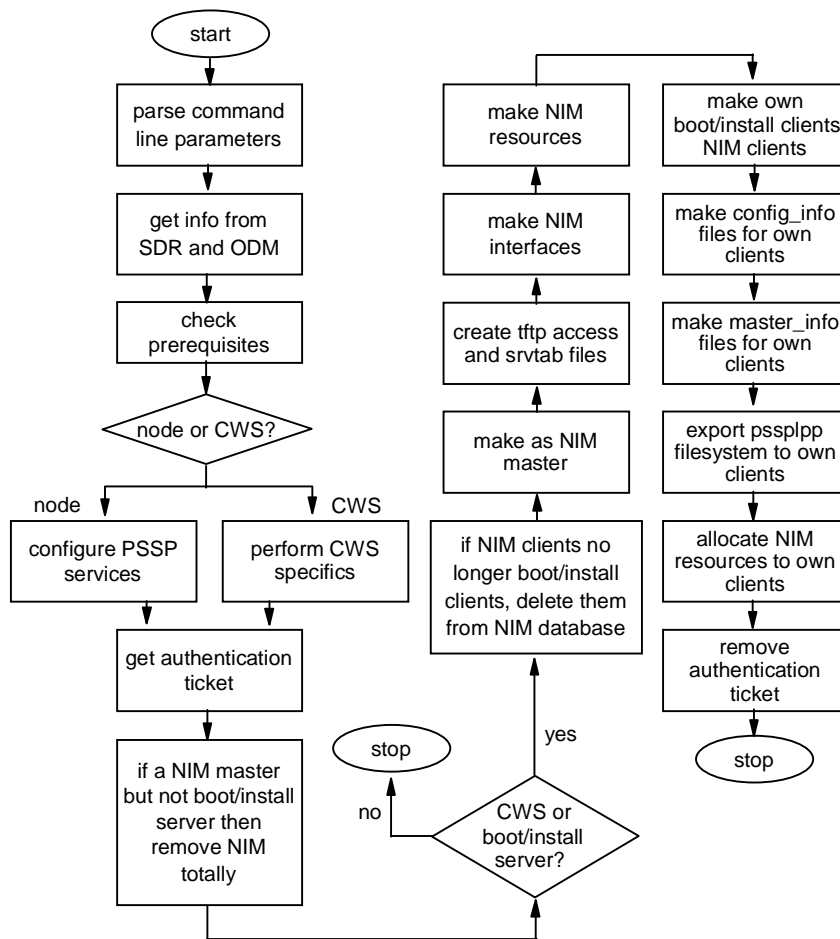


Figure 124. General Flow Of `setup_server` Command

After running `setup_server`, do a check to see if the System Management tools are properly set up using the following command:

```
# SYSMAN_test
```

10.3.3 Setting Up The Switch

If you have a switch, you must perform the next few steps. Depending on the number of switches, you need to select a correct switch topology file for your system. Refer to 9.4, “Switch Topology File” on page 218 for a description on this file and how to select the proper file. After determining the correct switch topology file to use, annotate the file. The SMIT fastpath for Eannotator is:

```
# smitty annotator
```

Enter the topology file name, specifying the full directory path. Also enter a fully-qualified name of a file in which to store the annotated topology file. Store the annotated file in the SDR. If the system cannot find a `/etc/SP/expected.top` file, it will read from the SDR to get the required information.



Figure 125. Annotating A Switch Topology File

The primary and primary backup nodes will already be defined. Verify this by running the `Eprimary` command. You will see an output like this:

```
1      - primary
1      - oncoming primary
```

```
15      - primary backup
15      - oncoming primary backup
```

The nodes assigned may be different in your environment, but all four roles are defined.

You must next set the switch clock source. Refer to 9.7, “Switch Clocks” on page 238 for details on selecting the correct Eclock topology file. To initialize the clock setting in SMIT, use the fastpath:

```
# smitty chclock_src
```

Important

If you have a running switch network, this command will bring the whole switch network down. The moment you select the topology file, it executes the `Eclock` command immediately. There is no warning message given.

You can partition your system now or you can do it after the installation. Refer to *PSSP: Administration Guide*, SA22-7348 for details on partitioning the system.

10.3.4 Node Installation

With all settings in place, the nodes can be installed. If your configuration has many nodes, we recommend that you perform installation of the nodes in batches of eight nodes or less at a time due to the performance issue. Perform installation on one node and assess the outcome. If the installation succeeds, you can then perform installation for the rest of the nodes. Install the node by netbooting it (a term used for network boot). Use the SP Hardware Perspective to perform the netboot function. From the Nodes pane, select the node to install by clicking it once. Select **Network Boot** from the Action menu bar. Click on the **Apply** button to perform the netboot function. Figure 126 on page 288 shows how this is done.

10.3.4.1 Network Boot

When netboot is performed, the `nodecond` command is issued. This command checks for the architecture of the node and invokes the appropriate scripts. If the node is an MCA node, it calls the `/usr/lpp/ssp/bin/nodecond_mca` script. If the node is a CHRP node, it calls the `/usr/lpp/ssp/bin/nodecond_chrp` script.

The `nodecond_chrp` script gets information from the SDR on the network type and attributes. Next, a serial port is opened via the `s1term` command. The node then gets powered off and then powered on again. When the RS/6000 logo appears, the script obtains the network address and performs a network boot by sending bootp packets to the node. When all these steps are done, NIM takes over with the installation process.

The `nodecond_mca` script does things a little differently. It first gets the network type and attributes from the SDR. It determines the node type to see whether it is a Thin, Wide or High node. Next, it powers off the node and opens up a serial port. If the node type is either Thin or Wide, it sets the key mode to secure. Next, it initiates a `hmmon` process to monitor the LED status.

The node is then powered on. When the LED reaches 200, the key mode is changed to service. On detecting an LED of 262, it lets the node know that the serial port is available. The IP address is determined and the network boot proceeds. NIM does the rest of the installation. For the High node, the key mode is switched to service. The BUMP processor is awoken and is set up. The node is then powered on. The script will then set the node up for network boot. The IP address is determined and the network boot proceeds. NIM takes over the rest of the installation.

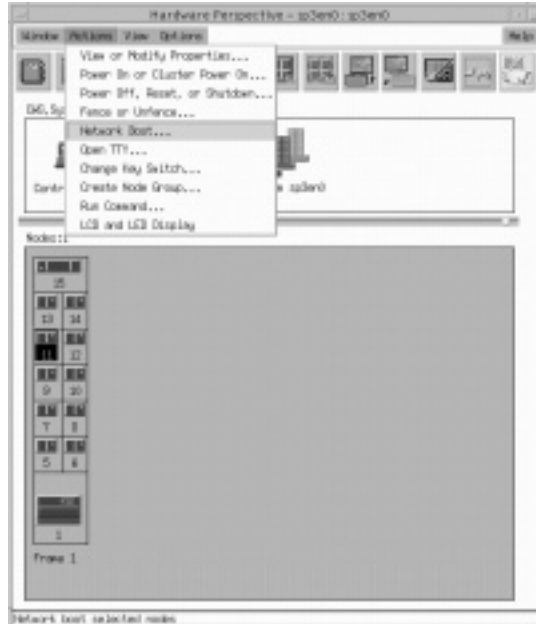


Figure 126. Network Boot Option

10.3.4.2 Customization

Once NIM finishes installing the AIX software on the node, and before the node is rebooted, it invokes the customization script `pssp_script`. This script performs the following functions:

- Creates the necessary directories `/var/adm/SPlogs/sysman` and `/tftpboot`
- Creates the log files to record all activities that occur during the customization phase
- Sets up environment, for example getting the Ethernet (en0) hardware address and node number for both the node as well as the boot/install server
- Configures the node's ODM without including the adapters at this point.
- Creates the `/etc/ssp` files
- Updates the `/etc/hosts` file with the latest information
- Transfers the `SDR_dest_info`, `script.cust`, `tuning.cust`, `spfbcheck` and `psspfb_script` files from the control workstation
- Sets up the Kerberos information
- Updates the `/etc/inittab` file to include setting up of default route and en0 is brought up.
- Adds entries to the `/etc/inittab` file to include the `spfbcheck` and `psspfb_script` scripts
- Installs all necessary PSSP files and adds the switch adapter information (if available) to the ODM
- Creates the dump device
- Initiates mirroring (if specified)
- Calls the `tuning.cust` script to tune the network parameters
- Calls the `script.cust` script

When `script.cust` finishes, the `psspfb_script` script is initiated. When everything is done, the `pssp_script` script exits. The first time a node is rebooted after installation, the `/etc/firstboot` script is called by the `fbcheck` script. The `/etc/firstboot` script contains information you specified earlier in the `firstboot.cust` file. The `fbcheck` script next renames the `/etc/firstboot` script so that it will not be executed the next time the node reboots.

Once the node is installed and set up, run the `SYSMAN_test` command again to verify that System Management tools are properly installed on the node. When this verification check succeeds, the rest of the nodes can be installed.

If you have a switch installed, you can start the switch network by issuing the `Estart` command. Run the `CSS_test` command to test the integrity of the switch. Verify that the `host_responds` and `switch_responds` are up and running. You can use the SP Hardware Perspectives to monitor this, or use the following command:

```
# sponon -d -G
```

10.3.4.3 Optional Tasks

The nodes are now installed. You may choose to perform additional tasks like mirroring the rootvg (if not already done), or install another rootvg onto another disk (if you have more than one disk drive).

The procedure to create mirroring after the rootvg is installed is simplified by the PSSP codes; see Figure 127. To create a mirrored copy of the rootvg from `hdisk0` to `hdisk1`, use the SMIT fastpath:

```
# smitty changevg_dialog
```

Specify the node on which you want mirroring to be performed. Enter `rootvg` as the Volume Group Name and specify the hard disks that `rootvg` will occupy (including `hdisk0`). Change the number of copies to two or three, depending on the mirroring copies you want.



Figure 127. Changing rootvg For Mirroring

After changing the rootvg characteristics, begin the mirroring process using the SMIT fastpath:

```
# smitty start_mirroring
```

Select the node and use Forced Extending the Volume Group in case the hard disk contains unwanted volume group information; see Figure 128. The mirroring process starts and takes about 30 minutes or more depending on the size of your volume group.



Figure 128. Initiate Mirroring

You can choose to install an alternate rootvg on another hard disk (maybe for testing purposes). You need to create the new volume group, naming it anything except rootvg (an example is rootvg1). The installation procedure is the same as for rootvg installation. With two rootvg residing on the two hard disks, you need to know how to switch them back and forth. The `spbootlist` command assists you in performing this task.

You will first change the Volume Group Information to the volume group from which you want to boot up. Use the following command if you want to use command line:

```
# spbootins -l <node number> -c <volume group name> -s no
```

Next, use the `spbootlist` command to change to bootlist on the node as follows:

```
# spbootlist -l <node number>
```

Perform a check on the bootlist of the node using the following command:

```
# dsh -w <node name> bootlist -m normal -o
```

If the change is correct, reboot the node. Your node will boot up from the rootvg you specified.

Chapter 11. System Monitoring

In order to successfully manage and administer an RS/6000 SP complex, the system administrator needs information. The type of information you require can be summarized in the responses to a series of hypothetical questions:

- Which nodes are up and available?
- How can I minimize application downtime to my users?
- Are any nodes experiencing environmental stresses?
- Is the SP Switch available to dependent applications?
- My manager wants to charge individual departments for their use of system resources — how can I generate the information he requires?
- A node has just failed on the switch network — how do I collect data to assist in problem determination?

The information presented in this chapter will help you answer these questions and guide you to other resources that will assist in system management activities. We will be covering the following topics in this chapter:

- SP Perspectives – the graphical user interface for system management on the RS/6000 SP
- SP Tuning and Performance Monitoring
- SP Accounting
- Problem Management

11.1 SP Perspectives

Scalable POWERparallel Perspectives for AIX (SP Perspectives) is a set of applications, each of which has 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.

By simply selecting an SP system object (a managed SP system resource such as a frame, node, or switch) by clicking on it with the mouse, you can select an action to perform on that object from the menu bar or tool bar. SP Perspectives uses this pattern of selecting an object, and then selecting an action to accomplish numerous system management tasks.

The AIX command `perspectives` starts the Launch Pad, from which you can launch the following applications:

- Hardware Perspective, for monitoring and controlling hardware

- Event Perspective, for managing system events and taking actions when events occur
- IBM Virtual Shared Disk Perspective, for managing shared disks
- The System Partitioning Aid
- The Performance Monitor Perspective
- access to other miscellaneous applications such as a set of pre-defined SP Perspectives applications, SMIT, the partition-sensitive subsystems command `syspar_ctrl`, and the SP Resource Center which gains access to the SP documentation through a Web browser interface.

The Launch Pad is shown in Figure 129.

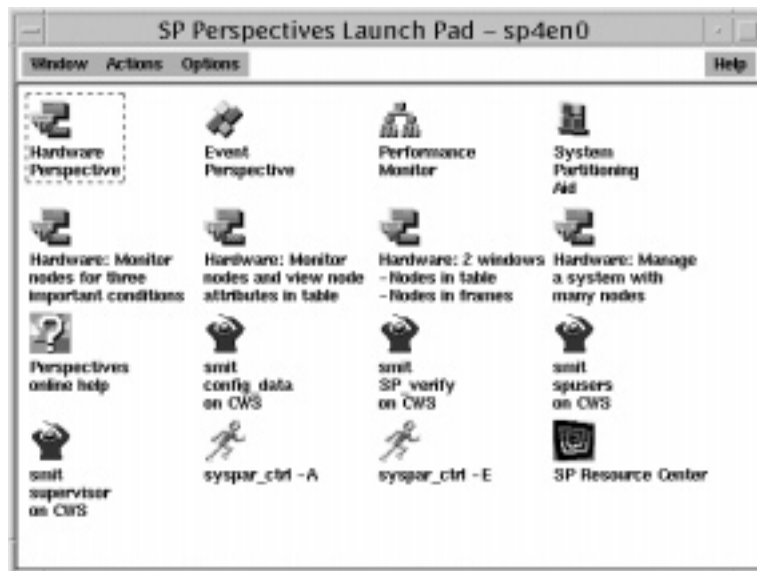


Figure 129. SP Perspectives Launch Pad

You can run the individual applications outside of the Launch Pad. The full pathnames of the individual applications are shown in Table 16.

Table 16. SP Perspective Application Pathnames

SP Perspective Application	Pathname
Hardware Perspective	/usr/lpp/ssp/bin/sphardware
Event Perspective	/usr/lpp/ssp/bin/spevent

SP Perspective Application	Pathname
IBM Virtual Shared Disk Perspective	/usr/lpp/ssp/bin/spvds
Performance Monitor	/usr/lpp/ssp/bin/spperfmon
System Partitioning Aid	/usr/lpp/ssp/bin/spsyspar
SP Resource Center	/usr/lpp/ssp/bin/resource_center

In previous releases of PSSP, another graphical interface (the original `spmon` GUI) was available and this was invoked by the `spmon -g` command. With the availability of PSSP v3.1, the `-g` flag is removed and the functionality that the original `spmon` graphical monitor provided is now available in SP Perspectives. (All other flags for `spmon` can still be used. For example, you are still able to issue the command `spmon -G -d`.)

A comprehensive review of SP Perspectives applicable to PSSP v3.1 can be found in the redbook *SP Perspectives: A New View of Your SP System*, SG24-5180. Refer to this book for extensive details on configuring and customizing the individual SP Perspective applications.

11.1.1 Hardware Perspective

The hardware perspective has two functions:

- Controlling hardware, for example, selecting a node, or several nodes, and performing an action on them such as power on/off, fence/unfence or network boot.
- Monitoring hardware, for example, opening a window which monitors important system conditions such as `host_responds`, `switch_responds` and individual node power LEDs.

Before we review these functions and how to perform them, we will look at the structure of the Hardware Perspective window as it appears when it is run either from the command line with the `sphardware` command or by double clicking the top leftmost icon **Hardware Perspective** on the launch pad.

The hardware perspective uses a concept of system objects. The system objects are defined as:

- Control workstation
- System
- System Partitions
- Nodes
- Frames and Switches

- Node Groups

These objects are displayed by default as icons, which are placed inside panes in the perspective window (icon view).

The hardware perspective allows four different kinds of pane. It refers to them as:

- CWS, System and Syspars (contains the control workstation, system and system partition objects)
- Nodes (contains node objects)
- Frames and Switches (contains frame and switch objects)
- Node Groups (contains node group objects)

Additional information on the features of the Hardware Perspective can be found in 5.3.2, “SP Perspectives” on page 119.

11.1.2 Event Perspective

Event Management is a distributed subsystem, a component of the RS/6000 Cluster Technology (RSCT). It matches information about the state of system resources with information about resource conditions that are of interest to client programs, which may include applications, subsystems, and other programs.

More detailed information about Event Management can be found in 8.6, “Event Management (EM)” on page 200. In that section we focus on the use of the Event Perspective to define and take action on events.

The Event Perspective allows you to define and manage event definitions within a system partition. In this sense it is not a GUI for displaying information; rather, it allows you to define monitors and triggers for other perspectives to display.

An *event definition* allows you to specify under what condition the event occurs and what actions to take in response. Using the Event Perspective, you can:

- Create an event definition
- Register or unregister an event definition
- View or modify an existing event definition
- Create a new condition

An event is triggered when a boolean expression involving a resource evaluates to true. A *resource* can be any entity in the system that can be observed, for examples processors, disk drives, memory, adapters, database

applications, processes, and file systems. A more detailed description of the Event Perspective, along with examples including defining and monitoring events, is given in *RS/6000 SP Monitoring: Keeping it Alive*, SG24-4873.

11.1.2.1 Getting Started

To use the event perspective to register event definitions with the problem management system, you must add your kerberos principal to the problem management access control list file `/etc/sysctl.pman.acl`. If you do not have an entry in this file, the Event Perspective will issue a warning as it initializes and certain actions cannot be performed. The file must exist on all nodes where an event definition is registered to a principal other than `root.admin`. A sample file is shown here.

```
#acl# /etc/sysctl.pman.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
```

You can initialize the Event Perspective from the launch pad, or alternatively from the command line using the `spevent` command. The main window of the Event Perspective is shown in Figure 130 on page 298.

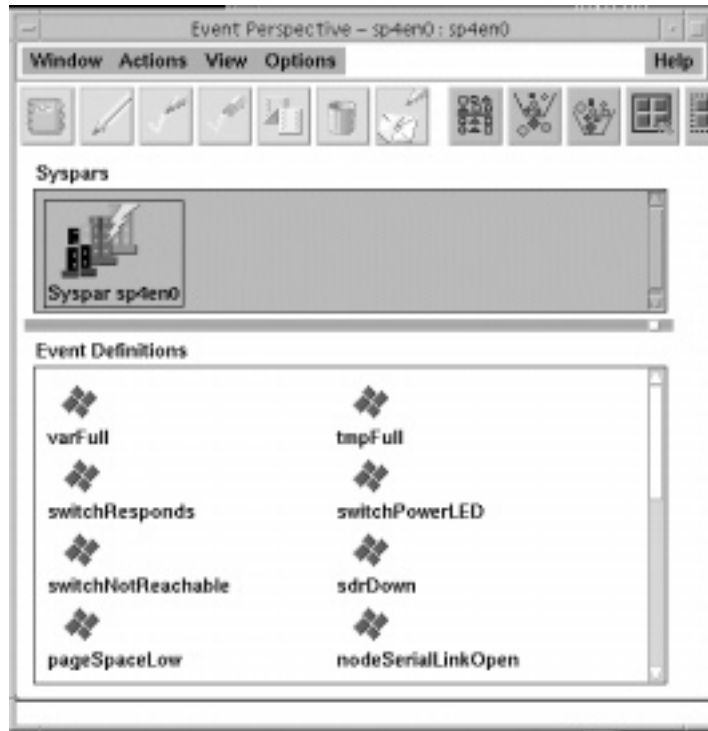


Figure 130. Event Perspective - Main Window

The bottom pane of the initial event perspective window shows some of the 19 pre-defined Event Definitions that are displayed when the Event Perspective is started. These definitions are summarized in Table 17.

Table 17. Pre-Defined Event Definitions

Event Definition Name	Event Definition Description
LCDhasMessage	The node's LED or LCD has a message.
errLog	A permanent error entry is added to the error log.
fileSystems	Monitor percentage usage of file systems.
frameControllerNotResponding	The frame controller is not responding.
framePowerOff	Frame power has been turned off.
hostResponds	The node is not responding.
keyNotNormal	Key mode switch is not in Normal position.

Event Definition Name	Event Definition Description
nodeEnvProblem	Environment LED is on; h/w problem detected.
nodeNotReachable	GS not able to communicate with node.
nodePowerDown	Node power is off.
nodePowerLED	Node power is off when powerLED is not 1.
nodeSerialLinkOpen	Serial link to the node (via TTY) is open.
pageSpaceLow	Node paging space use is greater than 85%.
sdrDown	SDR daemon has died.
switchNotReachable	Switch adapter not up on IP, or node is isolated.
switchPowerLED	Switch power is off when powerLED is not 1.
switchResponds	Switch adapter not responding, or node is isolated.
tmpFull	/tmp (LV=hd3) is running out of space.
varFull	/var (LV=hd9var) is running out of space.

By default, none of these events are active. You can start event monitoring using one or more of these defined events by using the following procedure:

1. Choose the event you wish to monitor by selecting its icon in the Events Definition pane. You can choose more than one event by pressing the left-CTRL button at the same time as you select the event(s).
2. Go to the Menu bar, and select **Actions->Register**.
3. The icons for the selected events will change from all-grey to colored. The definitions for the colors assigned to icons are shown in Figure 131 on page 300.

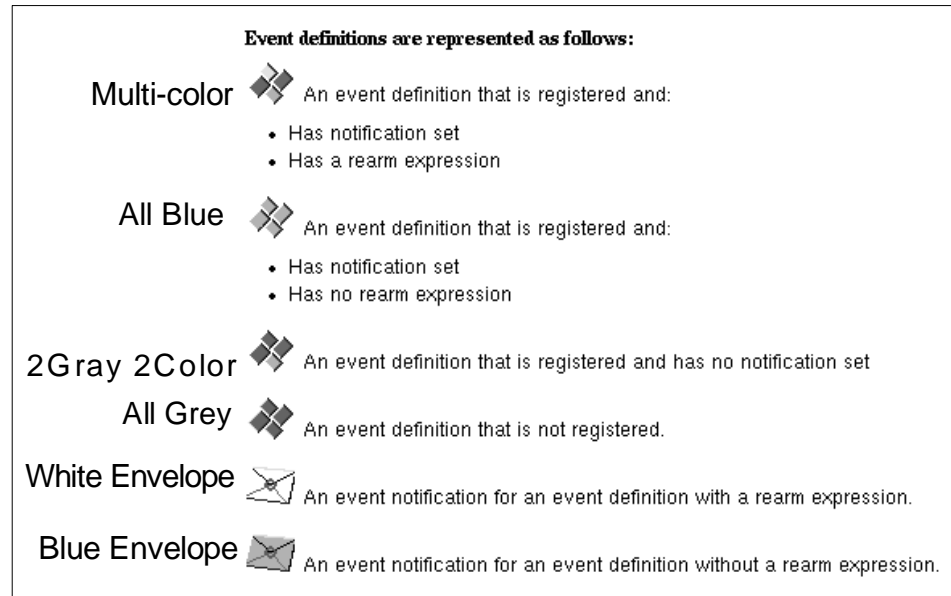


Figure 131. Icon Colors for Event Definitions

As an example, let us assume that the switchResponds event has been registered. Now if a node's switch adapter fails, or the Worm daemon dies, then the event icon will be replaced by the appropriate notification icon (in this case a white envelope, since a rearm expression has been set for this event). Once the node is back up on the switch, the icon will change back to its multi-colored status.

In addition to this visual alert, the system administrator will see an event notification log window pop up on his display. An example is shown in Figure 132 on page 301. There we can see both the original event notification when the switchResponds event was triggered, and the rearm notification once the problem was diagnosed and remedial action taken to fix it.

Notification	Event Definition	System	Resource	Time Observed
Event	switchResponds	sp5en0	NodeName=5	Wed Apr 21 14:15:32 1999
Event	switchResponds	sp5en0	NodeName=13	Wed Apr 21 14:15:32 1999

Buttons: Ok, Cancel, View Notification, Help

Figure 132. Sample Event Notification Log

11.1.2.2 Create and Monitor Your Own Events

PSSP v3.1 introduces a simple interface for creating an Event Condition and defining an Event Definition. These procedures are extensively described in *PSSP 3.1 Announcement*, SG24-5332.

As an example, we will monitor a specific application file system for utilization greater than 80%. The steps required to do this are as follows:

1. Ensure that the Conditions Pane has been added to the Event Perspective window.
2. From the Event Perspective with the Conditions Pane highlighted, select **Actions->Create** from the menu bar.
3. In the Create Condition notebook window that is generated, you will see two Resource Variable selection panes, as shown in Figure 133.

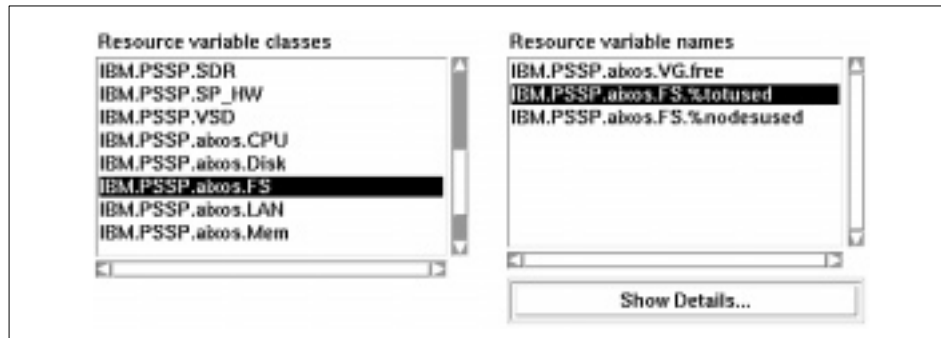


Figure 133. Event Definition - Resource Variable Selection

By scrolling through the classes pane, you find that the resource variable class associated with file systems is IBM.PSSP.aixos.FS and the variable name is IBM.PSSP.aixos.FS.%totused. Select these items, and complete the fields specifying the condition name and description, for example:

Name: apps_FS_used

Description: Monitors /apps filesystem on all nodes; triggers an event if utilization > 80%

4. Click the **Show Details** button to review a description of the resource variable and determine what to enter into the Event and (optional) Rearm Expression field. For this example, there are simple expressions:

Event Expression: $x > 80$

Rearm Expression: $x < 70$

5. Click **Ok**. The condition is created and appears in the Events Conditions pane.
6. Create an Event Definition based on this new condition. First, ensure that the Event Definitions pane is active. Then, from the menu bar, select **Actions->Create**. You should now see the Create Event Definition Window.
7. In the Condition Pane of this window, we can select the newly created Event Condition, apps_FS_used as shown in Figure 134.

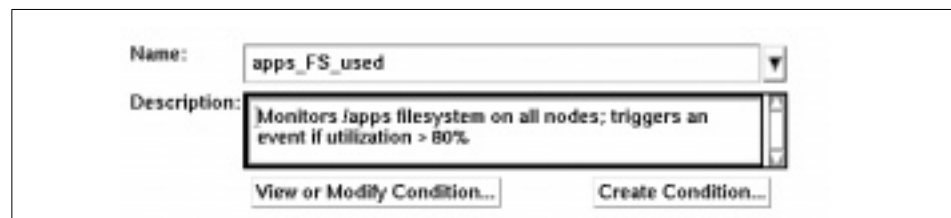


Figure 134. Event Definition - Condition Selection

8. Complete the Resource ID pane, which is shown in Figure 135 on page 303.

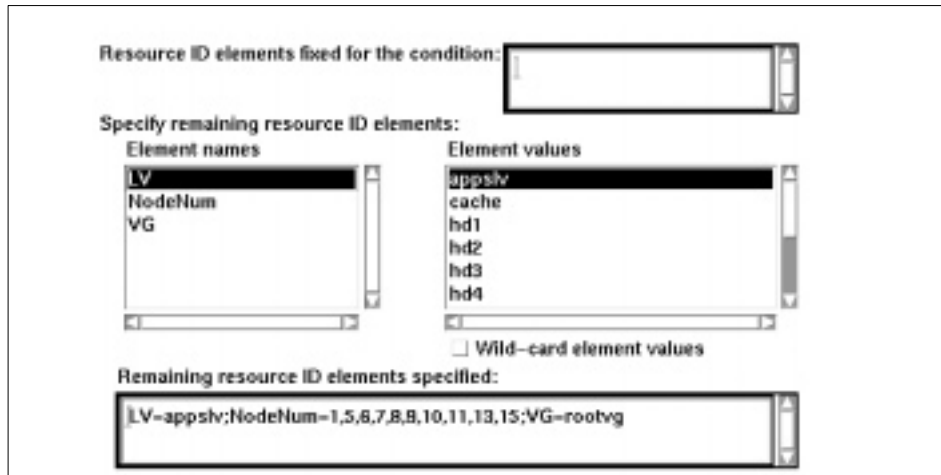


Figure 135. Event Definition - Resource Variable Identification

Here you can see that the resource elements IDs that have been selected are:

- The applsv logical volume, which resides on
 - The rootvg volume group that is in
 - Each node of the SP system (gaps in the node numbers indicate that there are no nodes in those slots)
9. Click on the **Create** button. The Event Definition will be saved and registered. When the file system utilization on any of these nodes becomes greater than 80% or falls below 70%, you will be notified by the Event Notification Log, as shown in Figure 136.

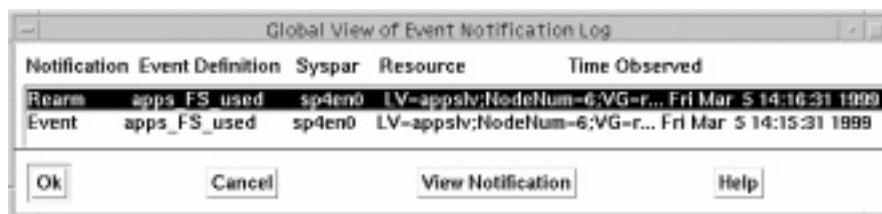


Figure 136. Event Notification Log

11.1.3 Virtual Shared Disk Perspective

IBM Virtual Shared Disk Perspective is the graphical user interface of PSSP that helps you perform shared disk management tasks. You can view, set, or change attributes and status, and you can control and monitor the operation of the shared disk management components. The IBM Virtual Shared Disk Perspective graphical user interface has the basic actions that let you perform most of your shared disk management work from within the graphical session.

Each action correlates to one or more virtual shared disk commands. You can run virtual shared disk, other PSSP, and AIX commands from within this interface as well. SMIT is also available to you for managing shared disks.

Explanation of the IBM Virtual Shared Disk Perspective graphical user interface is limited in this book to the brief introduction in this section. More information on VSDs can be found in 14.3, “Shared Disks” on page 394, and in *IBM Parallel System Support Programs for AIX Managing Shared Disks*, SA22-7349. A complete description of the capabilities of the VSD Perspective interface is given in the redbook *SP Perspectives: A New View of Your SP System*, SG24-5180.

Using this graphical interface requires your Kerberos principal to be defined in the virtual shared disk sysctl access control lists: `/etc/sysctl.vsd.acl` and `/etc/sysctl.acl`. The system administrator must run the command `sysctl svcrestart` for these changes to take effect. Once you have the correct authorization, you can start using the interface with either of two PSSP commands:

- Use the `perspectives` command to bring up the SP Perspectives Launch Pad window and then double-click on the **IBM VSD Perspective** icon to open the primary window.
- Use the `spvsd` command to open the IBM Virtual Shared Disk Perspective primary window directly.

Click **Help->Tasks** at the top right-hand corner of the primary window to see an online help system that explains how to use the IBM Virtual Shared Disk Perspective interface.

11.1.4 SP Command Line System Monitor

The SP System Monitor allows authorized operators and administrators to operate and monitor the system hardware. For information on using the command line interfaces to monitor and control the system hardware, refer to 5.3.1, “Command Line” on page 116.

11.2 SP System Tuning & Performance Monitoring

SP system tuning is generally an exercise in optimizing networks. SP system management, interprocessor communications, shared disk access, interconnection to the enterprise, and programming depend heavily on thoughtful tuning and exploitation of networks. Once the SP networks are optimized, you can tune an individual SP node the way you would any stand-alone RS/6000 machine.

11.2.1 RS/6000 SP Performance Considerations

Tuning the SP system is a complex task. A number of sources of information are already available that provide information on specific aspects of tuning the SP for optimal performance and general information on performance and tuning in a AIX environment. See Appendix C, "Related Publications" on page 525.

SP tuning and performance documentation is also available on the World Wide Web at the following URL:

<http://www.rs6000.ibm.com/support/sp/perf/>

The information at this site is updated with the latest performance and tuning data.

11.2.1.1 Large Configurations

Since an RS/6000 SP may grow to 512 nodes, the rules and assumptions used in tuning small networks of RS/6000 machines change in an SP environment. SP tuning requires strong change management disciplines, which stems from the complexity of large system configurations. While the vast majority of SP installations have less than 50 nodes, valuable lessons can be learned from large configurations.

In large SP configurations, a bottlenecked node (such as a communications gateway) can affect all other nodes in the system. Bottlenecks in large systems have correspondingly amplified negative impacts. For example, if the SP administrative Ethernet is overloaded by heavy NFS traffic caused by multiple concurrent node installs, the normally light load of the RS/6000 Cluster Technology (RSCT) daemons can exacerbate the network problem. The daemons will step up their communications to cope with the apparent unavailability of resources, thereby causing performance degradation.

11.2.1.2 SP Subsystems and Daemons

PSSP introduces many subsystems and daemons to manage the SP. In the default installation, these facilities present a light load to the system. As you start to exploit their function, be careful of potential performance impacts. For example, you could configure monitoring for hundreds of resource variables with RSCT, or mistakenly monitor one resource variable that triggers a steady stream of events (such as CPU usage greater than 0 on a node, and log the message). Generally, SP subsystems are efficiently designed so they can scale, but they can also be misused.

11.2.1.3 SP Administrative Ethernet

Use care in the design and loading of the SP administrative Ethernet. Many system management functions and enablers use the SP administrative Ethernet, such as node installation and customizing, LoadLeveler, Parallel Operating Environment (POE), and RSCT. The SP administrative Ethernet may also be connected to your external networks, and open to traffic in your enterprise.

Generally, SP systems with more than 16 nodes have multiple boot/install servers on physically separate Ethernet LAN segments. This maintains adequate network performance, particularly for intensive system management tasks such as node installation.

More information on the design of the SP administrative Ethernet can be found in 3.5, "Administrative Ethernet" on page 47.

11.2.1.4 SP System Tunable Network Options

The SP installation procedures direct you to select an appropriate tuning file based on your specific system environment. Sample tuning files are provided in the directory `/usr/lpp/ssp/install/config`. You can copy one of the sample files (or create your own tuning file based on one of these) to `/tftpboot/tuning.cust`; this file will be installed on each of the nodes. It will be executed each time the node is booted to ensure that your required tuning variables are in effect. The network tuning variables that are in this file are summarized in Table 18.

Table 18. Network Tuning Variables

Network Option variable	Description
thewall	The upper bound on the amount of real memory that can be used by the communications subsystem. The units are in 1K increments.
sb_max	Upper limit on the size of the TCP and UDP buffers in allocated space per connection.

Network Option variable	Description
ipforwarding	Specifies whether the kernel should forward IP packets. A value of 1 forwards packets.
tcp_sendspace	The default size of the TCP send space value in bytes.
tcp_recvspace	The default size of the TCP receive space value in bytes.
udp_sendspace	The default size of the UDP send space value in bytes. The safe maximum is 65536 (64K).
udp_recvspace	The default size of the UDP receive space value in bytes.
rfc1323	Turns on several enhancements to the tcp_sendspace and tcp_recvspace tunables for high speed networks: <ul style="list-style-type: none"> •The maximum TCP window size is increased from 64 Kbytes to 4 Gigabytes. •ACKs are time-stamped for better round trip time estimates. Sequence numbers are protected against wrapping.
tcp_mssdflt	The default maximum segment size used in communicating with remote networks.

11.2.1.5 High-Speed Interconnect Network: SP Switch

The SP Switch is the interconnect fabric of the nodes. Many network tuning parameters affect all network interfaces on a node. Nodes of a switched SP system always have at least one other network - the administrative Ethernet - and usually other networks connecting into the enterprise. Optimizing these global parameters for such disparate networks as a slow Ethernet and a high-speed SP Switch is not trivial, and usually involves trade-offs.

The SP Switch introduces specific tuning parameters. You can tune the switch's device driver buffer pools, *rpool* and *spool*, by changing the *rpoolsize* and *spoolsize* parameters on the node's switch adapter. These pools are used to stage the data portions of IP packets. Their sizes interact with the node's network options settings. A detailed discussion of tuning the *rpool* and *spool* buffers is presented in an online document at:

<http://www.rs6000.ibm.com/support/sp/perf>

The send pool and receive pool are separate buffer pools, one for outgoing data (send pool) and one for incoming data (receive pool). When an IP packet is passed to the switch interface, if the size of the data is large enough, a buffer is allocated from the pool. If the amount of data fits in the IP header

mbuf used in the mbuf pool, no send pool space is allocated for the packet. The amount of data that will fit in the header mbuf is a little less than 200 bytes, depending on what type of IP packet is being sent. The header size varies between using UDP or TCP, or having rfc1323 turned on.

To see the current send pool and receive pool buffer sizes, run the following command on your nodes:

```
[sp4n01:/]# lsattr -El css0
bus_mem_addr 0x04000000 Bus memory address      False
int_level    0xb          Bus interrupt level   False
int_priority  3           Interrupt priority    False
dma_lvl      9           DMA arbitration level False
spoolsize    524288      Size of IP send buffer True
rpoolsize    524288      Size of IP receive buffer True
adapter_status css_ready    Configuration status  False
[sp4n01:/]#
```

The default values for these buffer pools is set at 512 Kbytes or 524288 bytes. After reviewing your current usage of the buffer pools (by running the `vdid13` command on the node or nodes), and following the suggestions and recommendations in the online document, you can modify the `spoolsize` and/or `rpoolsize` parameters using the `chgcass` command. For example, to change both the send and receive buffer pool size on a node to 1 Mbyte, enter:

```
chgcass -l css -a rpoolsize=1048576 -a spoolsize=1048576
```

New values for the pool sizes must be expressed in bytes, and will not take effect until the node(s) are re-booted.

The small extra latency incurred by traversing intermediate switch connections in SP systems greater than 80 nodes may also be a consideration where consistent, precision results are required. (Remember that with configurations greater than 80 nodes, you need to add an intermediate switch frame to maintain switch performance.) For example, if a parallel job is dispatched to a set of nodes that all belong to one side of the intermediate switch frame, the execution time will be slightly faster than if the job is dispatched to the same number of identically configured nodes residing on either side of the intermediate switch frame.

11.2.1.6 Large Volumes of Data

SP systems typically have large volumes of data distributed across many nodes. Access to and movement of data within the system is critical to overall system performance. NFS, VSDs, and GPFS are all mechanisms for sharing

data in the SP. Each has particular tuning considerations and techniques, but fundamentally, an I/O request eventually comes down to a disk subsystem attached to a node. You must ensure that the I/O subsystem itself is not a bottleneck; otherwise, higher-level tuning will be fruitless.

Detailed tuning information for VSD and RVSD (the basis of GPFS) can be found in *IBM Parallel System Support Programs for AIX Managing Shared Disks*, SA22-7349.

11.2.1.7 External Networks

When you connect external networks to the SP, care must be taken to optimize the exchange of small data packets (typical of Ethernet, Token Ring, and similar slower-speed networks) and the large data packets of the SP Switch.

Adapter Transmit and Receive Queue Tuning

Most communication drivers provide a set of tunable parameters to control transmit and receive resources. These typically control the transmit queue and receive queue limits. These parameters limit the number of buffers or packets that may be queued for transmit, or limit the number of receive buffers that are available for receiving packets. These parameters can be tuned to ensure enough queueing at the adapter level to handle the peak loads generated by the system or the network.

Transmit queues

For transmit, the device drivers may provide a transmit queue limit. There may be both hardware queue and software queue limits, depending on the driver and adapter. Some drivers have only a hardware queue, some have both hardware and software queues. Some drivers internally control the hardware queue and only allow the software queue limits to be modified. Generally, the device driver will queue a transmit packet directly to the adapter hardware queue. On an SMP system, or if the system CPU is fast relative to the speed of the network, the system may produce transmit packets faster than they can be transmitted on the network. This will cause the hardware queue to fill. Once the hardware queue is full, some drivers provide a software queue and will then queue to the software queue. If the software transmit queue limit is reached, then the transmit packets are discarded. This can affect performance because the upper level protocols must then timeout and retransmit the packet.

Prior to AIX 4.2.1, the upper limits on the transmit queues were in the range of 150 to 250, depending on the specific adapter. The system default values were quite low, typically 30. With AIX 4.2.1 and later, the transmit queue limits were increased on most of the device drivers to 2048 buffers. The default

values were also increased to 512 for most of these drivers. The default values were increased because faster CPUs and SMP systems can overrun the smaller queue limits.

For adapters that provide hardware queue limits, changing these values will cause more real memory to be consumed because of the associated control blocks and buffers associated with them. Therefore, these limits should only be raised if needed, or for larger systems where the increase in memory use is negligible. For the software transmit queue limits, increasing these does not increase memory usage. It only allows packets to be queued that were already allocated by the higher layer protocols.

Receive Queues

Some adapters allow you to configure the number of resources used for receiving packets from the network. This might include the number of receive buffers (and even their size) or may simply be a receive queue parameter (which indirectly controls the number of receive buffers). The receive resources may need to be increased to handle peak bursts on the network.

When to Increase the Receive/Transmit Queue Parameters

Normally you would consider changing the queue size if one of the following situations arises in your network:

- The CPU is much faster than the network and multiple applications may be using the same network. This would be common on a larger multi-processor system (SMP).
- Running with large values for `tcp_sndspace` or `tcp_rcvspace` as set in the network options, or running applications that might use system calls to increase the TCP send and receive socket buffer space. These large values can cause the CPU to send down large numbers of packets to the adapter, which will need to be queued. A similar situation exists for `udp_sndspace` and `udp_rcvspace` for UDP applications.
- Network traffic is characterized by large data bursts.
- A high traffic load of small packets may consume more resources than a high traffic load of large buffers. This is because the large buffers take more time to send on the network, so the packet rate is slower for larger packets.

SP Considerations

One of the final RS/6000 SP installation steps is setting all network adapters in SP nodes to their maximum transmit queue size. With AIX release 4.2.1 and later, the default transmit queue limit has been increased to 512.

Attention

Be aware that the transmit queue size values recommended in *IBM PSSP for AIX Installation and Migration Guide, GA23-7347* at **Step 59: Tune the Network Adapters** are incorrect.

Check the value of this adapter attribute on all newly installed nodes using an appropriate `dsh` command, such as:

```
# dsh -a lsattr -El ent0 | grep xmt_que_size
```

If the value for the queue size is not 512, then change it using:

```
# dsh -a chdev -P -l ent0 -a xmt_que_size=512
```

Reboot the nodes to make the change effective. A value of 512 should be sufficient for initial operation, and can be increased if you change the `tcp_sendspace` or `tcp_recvspace` as set in the network options.

An Ethernet MTU is usually 1500 bytes. 512 MTUs represent 768,000 bytes, more than ten times the size of a single packet (MTU) on the SP Switch, which is 65,520 bytes.

11.2.2 Performance Management Tools

As part of the RS/6000 family of AIX machines, the RS/6000 SP inherits a set of mature performance management tools. In addition, IBM has enhanced the tools to scale with the SP.

11.2.2.1 AIX Facilities

AIX provides all the traditional UNIX resource monitoring tools. Each node's resources can be interrogated at the operating system level. Examples of these tools include the commands `sar`, `vmstat`, and `iostat`. If you wish to use these commands, the `bos.acct` fileset has to be installed on the nodes.

AIX Version 4 includes Performance Diagnostic Tool (PDT). PDT assesses the current state of a system and tracks changes in workload and performance. It attempts to identify incipient problems and suggest solutions before the problems become critical.

For the most part, PDT functions with no required user input. PDT data collection and reporting are easily enabled, and then no further administrator activity is required. Periodically, data is collected and recorded for historical analysis, and a report is produced and mailed to the `adm` userid. Normally, only the most significant apparent problems are recorded on the report. If

there are no significant problems, that fact is reported. PDT can be customized to direct its report to a different user or to report apparent problems of a lower severity level.

PDT optionally runs on individual nodes, assesses the system state, and tracks changes in performance and workload. It tries to identify impending problems and suggest solutions before they become critical.

The PerfPMR package was developed to ensure that reports of suspected performance problems in AIX were accompanied by enough data to permit problem diagnosis by IBM. This makes the shell scripts in PerfPMR useful to other performance analysts as well. PerfPMR is an optionally installable part of the AIX Version 4 Base Operating System.

The SP currently offers no specific reporting consolidation or meta-level commands in this area. AIX resource monitoring capabilities are well explained in *AIX Performance Monitoring and Tuning Guide*, SC23-2365.

11.2.2.2 Reporting Resource Data Information

Performance Agent, a component of Performance Toolbox/6000, is an element of the resource monitoring facility of RSCT. By itself, Performance Agent only supplies data from performance related resource variables to the System Performance Measurement Interface (SPMI); it offers little in the way of viewing or analyzing the data. IBM supplies program products specifically to exploit the performance data from Performance Agent.

Performance Toolbox (PTX/6000)

PTX/6000 comprises the Performance Manager (PM) and Performance Agent (PA), or `perfagent.server`. The latter is the exact same code that ships automatically with the SP. The PM component includes X-Windows and Motif-based applications that provide:

- Real-time, color graphic performance monitors for local and remote systems, such as `3dmon`
- Performance analysis tools
- Performance tuning controls

PTX/6000 has a long, successful heritage of monitoring the performance of RS/6000 stand-alone machines. PTX/6000 is a separately orderable and priced IBM Licensed Program Product (LPP).

With reference to Figure 137 on page 313, the `xmservd` daemon of PA feeds the performance statistics into shared memory. While the `aixos` resource monitor within the Event Management (EM) daemon can supply AIX-level resource variable data selectively, as per EM client event registrations,

xmservd can be further configured for specific resource monitoring needs. Performance Manager (PM) can simply act as another client of the SPMI interface (or Local Data Consumer in PTX/6000 vernacular), just as does the Resource Monitors of Event Manager. The diagram introduces another interface into shared memory: the Remote Statistics Interface (RSI). This interface allows a PM instance on another node (shown as a Remote Data Consumer, in this case the 3dmon monitoring program of PM) to access the local node's shared memory. This way, one PM installation can monitor a cluster of machines at the same time.

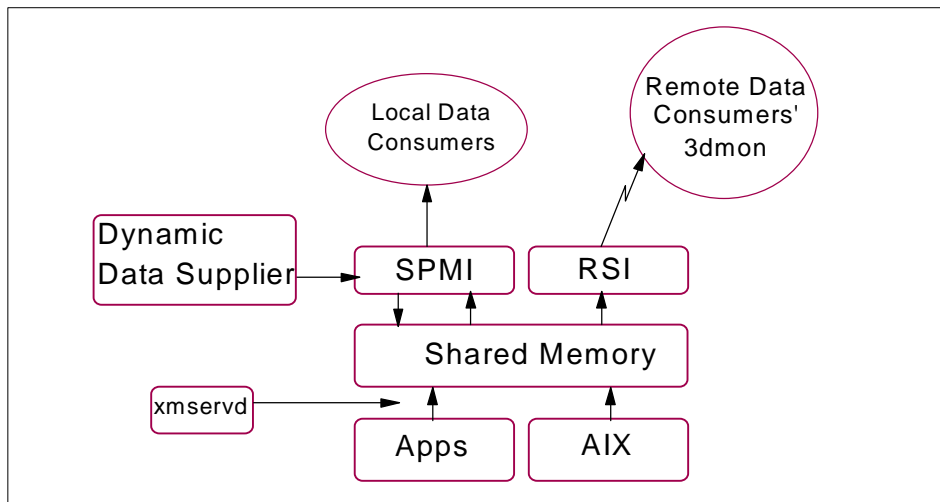


Figure 137. PTX Functional Overview

Although Performance Agent (PA) is an integral part of the SP's software implementation, PM offers no SP-specific function. Its usability does not scale: on larger SP installations (greater than 24 nodes), its graphical monitoring tools cannot illustrate on a screen all the performance statistics of all nodes. PM is not aware of specific SP hardware and software enablers.

Performance Toolbox Parallel Extensions (PTPE)

PTPE extends the function of PTX/6000 for use in an RS/6000 SP complex. PTPE is a scalable performance monitor, and when it is installed, it provides easy access to performance information. Any node or group of nodes can be monitored with a single point of control. The performance metrics and the information display criteria can be specified. Monitoring can be carried out in real time, or the performance data archived for later analysis. For detailed information on configuring and using PTPE, refer to *IBM PSSP for AIX*

Performance Monitoring Guide and Reference, SA22-7353 and the redbook RS/6000 SP Performance Tuning, SG24-5340.

PTPE performance statistics can be viewed:

- Averaged across the RS/6000 SP complex
- Averaged across a subset of nodes
- For individual nodes

PTPE setup, configuration and management have been integrated into Perspectives, or alternatively, can be carried out using AIX commands.

The implementation of PTPE is shown in Figure 138.

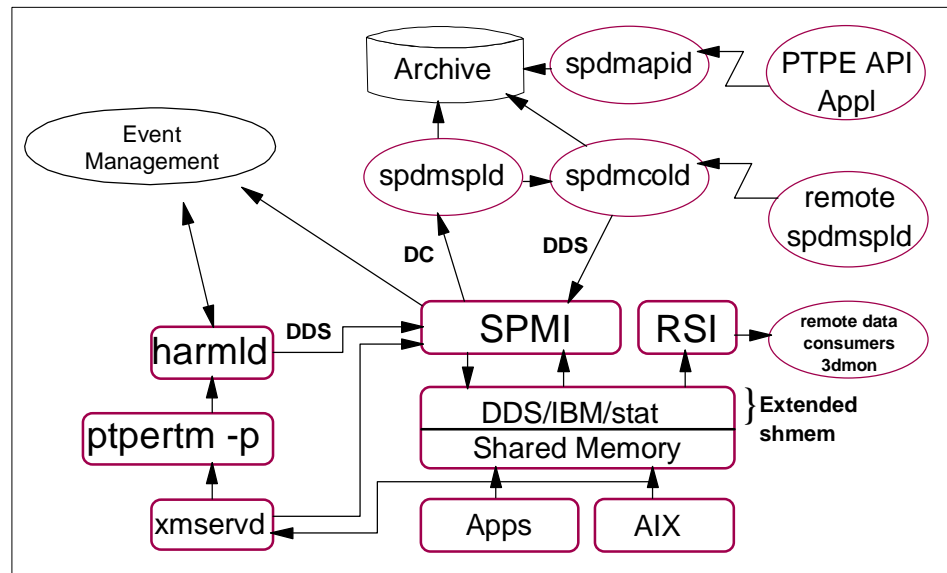


Figure 138. PTPE Architecture

In addition to the capabilities of PTX/6000, PTPE provides:

- **Collection of SP-specific data** - PTPE provides ptperrm, an additional data supplier that complements the data that xmservd collects. The SP-specific performance data is currently implemented for:
 - The SP Switch
 - LoadLeveler
 - VSD
- **SP runtime monitoring** - The system administrator should ideally have a global view of SP performance behavior. This is accomplished by grouping

nodes into specific groups. With reference to Figure 139 on page 315, similar nodes of the first tier, or Collectors, can be grouped and their performance data summarized by their respective Data Manager node in the second tier. In this way, large SP systems can be easily monitored from a single presentation application by viewing node groups instead of individual nodes. The Data Managers are administered by the Central Coordinator in the third tier. The Central Coordinator aggregates the Data Managers' summary data to provide a total performance overview of the SP. Of course, the base PTX/6000 monitoring functions can be used to focus on any particular performance aspect of an individual node.

The PTPE display monitors on the Data Manager receive data from `spdmspld` (SP Data Manager Sampler daemon) on the local node. To acquire data from remote Collectors, `spdmspld` communicates with `spdmcold` (SP Data Manager Collector daemon) on the remote nodes, which in turn get the data from their associated `spdmspld` daemon. The communications between these daemons is via a private protocol. Note that PTPE does not use the same mechanism as PTX/6000 to collect data from remote nodes.

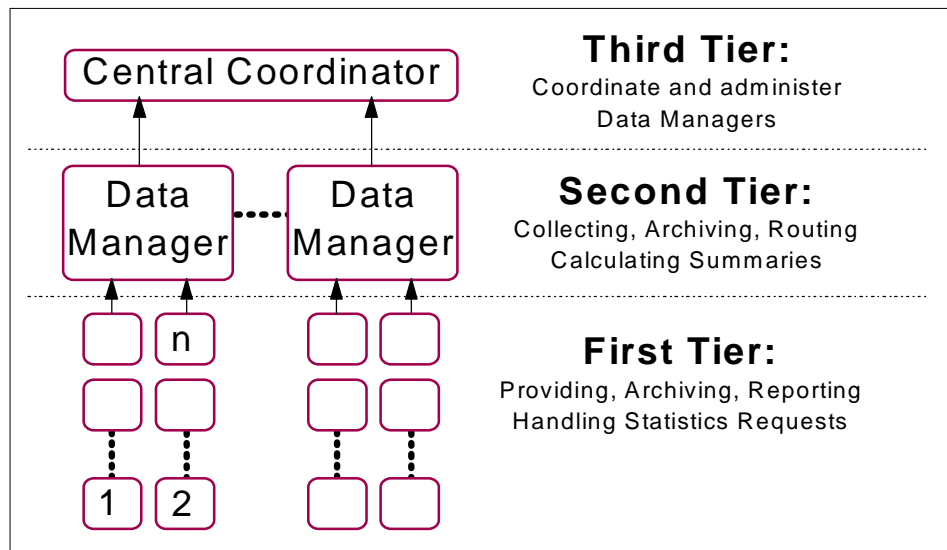


Figure 139. PTPE Monitoring Hierarchy

- **Data Analysis and Data Relationship Analysis** - PTPE provides an API to allow analysis applications to sift through all requested data. The data archive created by PTPE exists on every node, and is completely accessible through the API using the `spdmapid` daemon (see Figure 138

on page 314). In base PTX/6000, performance data is analyzed with the azizo utility, which is restricted to simple derivatives like maximum, minimum, and average. With the PTPE API, programs of any statistical complexity can be written to find important trends or relationships. Also, data captured for azizo use is far more limited with base PTX/6000.

PTPE Installation and Customization

To install and run the Performance Toolbox Parallel Extensions for AIX, the following IBM licensed program products must also be installed:

- AIX Version 4 Release 3.2
- RS/6000 Cluster Technology
- Performance Aide for AIX Version 4.1 (5696-899)
This must be installed on all nodes where PTPE is to run (the nodes you intend to monitor), as well as on the control workstation.
- Performance Toolbox-Network for AIX Version 4.1 (5696-900)
This must be installed on any nodes where you will display performance data (the nodes from which you intend to monitor).

There are three filesets in the PSSP installation media that make up PTPE:

ptpe.program The PTPE programs. This software will not run unless RSCT has been installed.

ptpe.docs Documentation material, for example man pages.

ssp.ptpegui This image should be installed on any node on which you plan to run SP Perspectives.

The software installation is straightforward, and details can be found in Chapter 3, “Installing PTPE” of *IBM PSSP for AIX Performance Monitoring Guide and Reference*, SA22-7353.

Before you can use Performance Toolbox Parallel Extensions for AIX, you must create a monitoring hierarchy as shown in Figure 139 on page 315. PTPE distributes the management of performance information among a number of data manager nodes rather than giving total responsibility to a single node. Although one central coordinator node is designated when the monitoring hierarchy is created, the data manager nodes act as intermediaries, absorbing most of the administrative overhead and greatly reducing data transfer operations. The resulting central coordinator node workload is far less than that required by a single point of control for all nodes and all data management functions.

You can create a monitoring hierarchy using either the SP Perspectives Performance Monitor or the `ptpehier` command. In our example, we will

consider the single frame SP system shown in Figure 140, and use the `ptpehier` command.

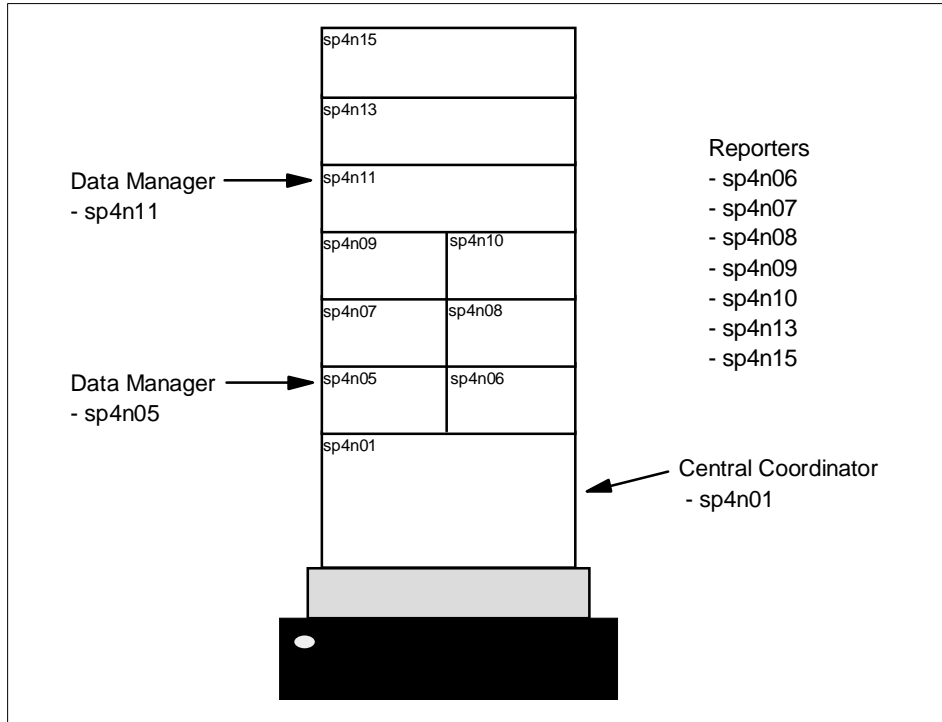


Figure 140. RS/6000 SP System with Defined Roles

Here we have decided that nodes `sp4n05` and `sp4n11` will be Data Managers, and `sp4n01` will be the Central Coordinator; the remaining nodes will be simply Reporters. To formalize this hierarchy, you create a text file (we call it `samp_hier`) in the `/tmp` directory, with the following format:

```

{
sp4n05.msc.itso.ibm.com
sp4n06.msc.itso.ibm.com
sp4n07.msc.itso.ibm.com
sp4n08.msc.itso.ibm.com
sp4n09.msc.itso.ibm.com
sp4n10.msc.itso.ibm.com
}
{
sp4n11.msc.itso.ibm.com
sp4n13.msc.itso.ibm.com
sp4n15.msc.itso.ibm.com
sp4n01.msc.itso.ibm.com
}

```

To initialize this hierarchy, you use the `ptpehier` command with a `-c` flag to specify the Central Coordinator node and `-i` flag to specify that the hierarchy will be read from standard input — in this case, the `/tmp/samp_hier` file. You can also let PTPe automatically create a hierarchy, based on your Ethernet local area subnetwork, or if you have a multi-frame SP system, then it can develop a hierarchy based on the node/frame configuration. In our example, we use the following commands to set up and confirm the hierarchy.

```

[sp4en0:/]# /usr/lpp/ptpe/bin/ptpehier -i -c sp4n01 < /tmp/samp_hier
ptpehier: Monitoring hierarchy successfully created.

[sp4en0:/]# /usr/lpp/ptpe/bin/ptpehier -p
ptpehier: The current monitoring hierarchy structure is:
    sp4n01.msc.itso.ibm.com
        sp4n11.msc.itso.ibm.com
            sp4n11.msc.itso.ibm.com
            sp4n13.msc.itso.ibm.com
            sp4n15.msc.itso.ibm.com
            sp4n01.msc.itso.ibm.com
        sp4n05.msc.itso.ibm.com
            sp4n05.msc.itso.ibm.com
            sp4n06.msc.itso.ibm.com
            sp4n07.msc.itso.ibm.com
            sp4n08.msc.itso.ibm.com
            sp4n09.msc.itso.ibm.com
            sp4n10.msc.itso.ibm.com

[sp4en0:/]#

```

All that remains is to initialize and start the data collection process using the `ptpectrl` command with the `-i` and `-c` flags:

```

[sp4en0:/]# ptpectrl -i
-----
ptpectrl: Beginning setup for performance information collection.
ptpectrl: Reply from the Central Coordinator expected within 255 seconds.
ptpectrl: Setup for collecting performance information succeeded.
-----
ptpectrl: Command completed.
-----
[sp4en0:/]#

[sp4en0:/]# ptpectrl -c
-----
ptpectrl: Starting collection of performance information.
ptpectrl: Reply from the Central Coordinator expected within 250 seconds. OK.
ptpectrl: Performance information collection successfully started.
-----
ptpectrl: Command completed.
-----
[sp4en0:/]#

```

Now you can use the familiar `xmperf` and `3dmon` commands to review the performance of the nodes in your SP complex. Since performance monitoring creates additional load on the nodes being monitored, you should shut down the monitoring process once you have collected sufficient performance data, using the `ptpectrl -s` command.

11.2.2.3 Performance Event Monitoring with RSCT Clients

Performance-related resource variables from Performance Agent are part of Event Manager's standard suite of monitored resources variables. You can always use your own programs or PMAN to interface to RSCT to monitor and take action on performance-related events. See 8.6, "Event Management (EM)" on page 200 and 11.4.2, "Problem Management (PMAN) Subsystem" on page 329 for more on this topic.

11.3 SP Accounting

The SP accounting facilities are based on the standard System V accounting system utilities of AIX. These accounting facilities are described in *AIX Version 4.3 System Management Guide: Operating System and Devices*, SC23-4126. Enabling SP system accounting support is optional. It is a separately-installed portion of the PSSP `ssp.sysman` fileset. SP system accounting extends the function of base AIX accounting in three ways:

- **Accounting-record consolidation** - Partial reduction of accounting data is done on each node before the data is consolidated on an accounting

master. (The accounting master will generally be the control workstation, but it can be any node.) Nodes are configured into groups called accounting classes. All the data from a given accounting class is consolidated. These accounting classes can be used to impose different charges for nodes with differing capabilities.

The administrator can use the SMIT menus and dialogs or the command line interface to specify the accounting configuration. The SP system management code automatically configures accounting on the selected nodes. The command `runacct` (a modified version of the AIX `runacct` command) is scheduled to run nightly on each node to consolidate the accounting data for that node. Data from all of the nodes is then consolidated on the accounting master node, which performs additional processing. The output of this processing is standard format data that can be used for charge-back purposes, usage monitoring or capacity planning. It also can be fed into existing accounting applications that you may have already implemented.

- **Node-exclusive-use accounting** - Standard accounting generates charges for resources used (processor, disk, and so on). If a user is given exclusive use of a set of nodes for the duration of a parallel job, standard accounting may not be an appropriate way to charge for processor usage. Instead, administrators may want to bill based on the wall-clock time that a processor is in use since it is unavailable to other users during that period (regardless of what processor cycles are actually consumed by the running job).

The PSSP software provides an optional mechanism with which to charge for nodes that have been assigned for exclusive use. If this support is enabled, an accounting record for a marker process is created before and after the job is run. All processor accounting records associated with that user and falling within the window of the marker records are discarded. Instead, a special charge is applied based on the actual number of seconds the job runs. The fee is specified by the administrator through the standard accounting charge-fee mechanism.

- **Parallel job accounting** - The LoadLeveler job-management program allows job-based accounting by accumulating data on resource usage from all processes triggered by a specific job.

11.3.1 Configuring SP Accounting

AIX accounting requires the creation of directories, files, and crontab entries, as well as accounting configuration activity. This is appropriate for a single system, but a daunting task for an SP with tens or hundreds of nodes. The PSSP software simplifies this task greatly. After installation of the accounting

portion of the SP, default values in the SDR configure accounting across the entire system. Accounting can be enabled or disabled globally by changing the site environment data. Node-by-node accounting activation, accounting classes creation, and exclusive-use support on a node are efficiently configured using SMIT menus or SDR commands to update node attributes collectively. The accounting installation process creates all necessary directories and files on the nodes for which accounting is enabled.

The following procedure helps you to tailor your accounting system by defining accounting class identifiers and accounting-enabled attributes on a SP level or on an individual node basis, if necessary. Accounting class identifiers need only be specified when you want more than one accounting class. You can also specify which nodes are to have accounting enabled. However, if all nodes are to be enabled, then only SP Accounting Enabled need be set to true and the `acct_enable` attribute can be left to default on each Node object in the SDR.

In addition, this procedure shows you how to specify exclusive use accounting, and how to specify an Accounting Master other than the default control workstation. It also shows how to set an SP Accounting Active Node Threshold value other than the default value of 80. This threshold specifies the minimum percentage of nodes for which accounting data must be present in order for processing to continue.

You can define or change these values after installation by using SMIT (the fastpath is `site_env_dialog`) or the `spsitenv` commands. An extract of the relevant SMIT screen is shown in Figure 141.

SP Accounting Enabled	true	+
SP Accounting Active Node Threshold	[80]	#
SP Exclusive Use Accounting Enabled	true	+
Accounting Master	[0]	

Figure 141. Site Environment- Accounting Setup

The `spsitenv` command could also be entered at a prompt on the control workstation:

```
# spsitenv acct_master=0 spacct_enable=true spacct_exclude_enable=true
```

You also need to further define SP accounting at the node level. This can be done using the SMIT fastpath `acctnode_dialog` or the `spacctnd` command. The following sequence of `spacctnd` commands will set the job charge value for nodes 1, 9 and 10 to 30.0, and these nodes will have the *default* accounting

class identifier. Nodes 5, 6, 7, 8, 11, 13 and 14 will have the *pjobs* accounting class identifier, exclusive use accounting will be enabled, and the job charge value will be set to 60

```
# spacctnd -j 30.0 1 1 1
# spacctnd -j 30.0 1 9 2
# spacctnd -c pjobs -x true -j 15.0 1 5 4
# spacctnd -c pjobs -x true -j 15.0 1 11 3
```

We can review the relevant attributes of the Node class in Figure 142 on page 322.

```
[sp4en0:/]# SDRGetObjects Node node_number acct_class_id acct_enable acct_job_charge
acct_excluse_enable
node_number acct_class_id acct_enable acct_job_charge acct_excluse_enable
1 default default 30.0 false
5 pjobs default 15.0 true
6 pjobs default 15.0 true
7 pjobs default 15.0 true
8 pjobs default 15.0 true
9 default default 30.0 false
10 default default 30.0 false
11 pjobs default 15.0 true
13 pjobs default 15.0 true
15 pjobs default 15.0 true
[sp4en0:/]#
```

Figure 142. SP Nodes - Accounting Setup

These accounting installation procedures result in the following activities being carried out on the accounting master node (usually the control workstation) and nodes.

1. All necessary directories and files on the nodes for which accounting is enabled and on the acct_master node are created.
2. The accounting startup command is added to the /etc/rc file on the nodes for which accounting is enabled.
3. The /var/adm/acct/nite/jobcharge file is created on each node for which accounting is enabled. This file contains the job charge value previously defined for the node.
4. The holidays file (/etc/acct/holidays) is placed in the user.admin file collection and is available on all nodes. Its source is the control workstation. If you do not use file collections, propagate this file to all nodes in the SP system, using whatever method is available in your environment.

Note: You may have to update the holidays file to reflect the correct information for the current year.

5. The crontabs file is updated to schedule the accounting processes.
6. The login and process account data files are initialized by the `nulladm` command.

```
/usr/sbin/acct/nulladm /var/adm/wtmp /var/adm/pacct
```

Tools and Considerations

Like AIX accounting, the SP merges accounting data by user ID and login name. This forces all SP nodes for which accounting is enabled to have identical user ID and login names; in other words, belong to the same user name space. Name space can be managed by NIS, SP User Management, or other mechanisms.

See *IBM Parallel System Support Programs for AIX Administration Guide*, SA22-7348, for more information on SP accounting implementation.

11.3.2 LoadLeveler Accounting

LoadLeveler is an application designed to automate workload management. In essence, it is a scheduler which also has facilities to build, submit and manage both serial and parallel jobs. The jobs can be processed by any one of a number of machines, which together are referred to as the LoadLeveler cluster. Any standalone RS/6000 can be part of a cluster, although LoadLeveler is most often run in the RS/6000 SP environment.

For more information about configuring and administering LoadLeveler on an SP system, see Chapter 17, "LoadLeveler" on page 497, and *IBM LoadLeveler for AIX: Using and Administering Version 2 Release 1*, SA22-7311.

LoadLeveler can collect a variety of accounting information, based on your local requirements. Refer to the relevant documentation for more information.

Job Resource Data - Machines

LoadLeveler can collect job resource usage information for every machine on which a job may run. A job may run on more than one machine because it is a parallel job or because the job is vacated from one machine and rescheduled to another machine.

To turn on accounting in this environment, specify

```
ACCT = A_ON A_DETAIL
```

in the LoadLeveler configuration file LoadL_config.

Job Resource Data - Serial and Parallel Jobs

Information on completed serial and parallel jobs is gathered using the UNIX *wait3* system call. Information on non-completed serial and parallel jobs is gathered in a platform-dependent manner by examining data from the UNIX process.

Accounting information on a completed serial job is determined by accumulating resources consumed by that job on the machine(s) that ran the job. Similarly, accounting information on completed parallel jobs is gathered by accumulating resources used on all of the nodes that ran the job.

To turn on accounting in this environment, specify the following keywords in the LoadLeveler configuration file LoadL_config.

```
ACCT = A_ON A_DETAIL
JOB_ACCT_Q_POLICY = num1
JOB_LIMIT_POLICY = num2
```

The value num1 is an accounting update frequency (in seconds) with a default value of 300, and num2 is also a number in seconds; the smaller of these two numbers controls how often resource consumption data on running jobs is collected.

Job Resource Data - Events

LoadLeveler can also collect job resource information based on events or times that you specify. For example, you may want to collect accounting information at the end of every work shift or at the end of every week or month.

To collect accounting information from all machines in this way, the *llctl* command is used with the *capture* keyword

```
llctl -g capture eventname
```

where eventname is any string of continuous characters that defines the event about which you are collecting accounting information. For example, to collect this type of information regularly, you can add crontab entries, such as:

```
00 08 * * * /u/loadl/bin/llctl -g capture third
00 16 * * * /u/loadl/bin/llctl -g capture first
00 00 * * * /u/loadl/bin/llctl -g capture second
```

Job Resource Information — User Accounts

You can also keep track of resources used on an account basis by requiring all users to specify an account number in their job command files. Users can specify this account number with the *account_no* keyword. This account number must be a member of a set of account numbers specified in the LoadLeveler administration file *LoadL_admin* with the *account* keyword.

Examples

Once LoadLeveler accounting has been configured, you can extract job resource information on completed jobs by using the `llsummary` command. For detailed information on the syntax of this command and the various output reports that it can generate, see *LoadLeveler for AIX: Using and Administering Version 2 Release 1*, SA22-7311.

LoadLeveler stores the accounting information that it collects in a file called *history* in the `/home/loadl/spool` directory of the machine that initially scheduled this job. Data on parallel jobs is also stored in the *history* files of the nodes that are part of the parallel job pool.

You can produce three types of reports using the `llsummary` command. These reports are called the *short*, *long*, and *extended* versions. As their names imply, the short version of the report is a brief listing of the resources used by LoadLeveler jobs, the long version provides more comprehensive detail with summarized resource usage and the extended version of the report provides the comprehensive detail with detailed resource usage. If you do not specify a report type, you will receive the default short version.

The short report displays the number of jobs, along with the total CPU usage according to user, class, group, and account number. The extended version of the report displays all of the data collected for every job.

An extract of a short report follows.

```

$ llsummary
  Name      Jobs  Steps      Job Cpu  Starter Cpu  Leverage
  user8     8     8     0+00:00:56  0+00:00:02    28.0
  usera91   5     5     0+00:00:01  0+00:01:10     0.0
  TOTAL    13    13     0+00:00:57  0+00:01:13     0.8

  Class     Jobs  Steps      Job Cpu  Starter Cpu  Leverage
  No_Class  13    13     0+00:00:57  0+00:01:13     0.8
  TOTAL    13    13     0+00:00:57  0+00:01:13     0.8

  Group     Jobs  Steps      Job Cpu  Starter Cpu  Leverage
  No_Group  13    13     0+00:00:57  0+00:01:13     0.8
  TOTAL    13    13     0+00:00:57  0+00:01:13     0.8

  Account   Jobs  Steps      Job Cpu  Starter Cpu  Leverage
  NONE      10    10     0+00:00:28  0+00:01:12     0.4
  test      3     3     0+00:00:29  0+00:00:01    29.0
  TOTAL    13    13     0+00:00:57  0+00:01:13     0.8

$

```

The short listing includes the following fields:

- Name** User ID submitting jobs
- Class** Class specified or defaulted for the jobs
- Group** User's LL group
- Account** Account number specified for the jobs
- Jobs** Count of the total number of jobs submitted by this user, class, group, or account
- Steps** Count of the total number of job steps submitted by this user, class, group, or account
- Job CPU** Total CPU consumed by user's jobs
- Starter CPU** Total CPU time consumed by LoadLeveler starter processes on behalf of the user jobs
- Leverage** Ratio of job CPU to starter CPU

11.4 Problem Management

The RS/6000 SP uses many techniques to report and act on errors and problems that are encountered during system operation. In general, problem management on the RS/6000 SP exploits the existing AIX facilities and the rich services provided by the RS/6000 Cluster Technology (RSCT) (known in pre-PSSP v3.1 releases as High Availability Infrastructure or RSCT). This topic is discussed in Chapter 8, "RS/6000 Cluster Technology" on page 185.

11.4.1 AIX, BSD and PSSP-Specific Error Logging

The RS/6000 SP uses both the AIX Error Logging facilities and the BSD syslog, as well as a number of function-specific log files to record error events on each node. Commands and SMIT panels are available to perform general log management. In order to manage Syslog and the AIX Error Log, the `ssp.sysman` fileset must be installed.

Error logging reports debugging information into log files for subsystems that perform a service or function on behalf of an end user. The subsystem does not communicate directly with the end user and therefore needs to log events to a file. The events that are logged are primarily error events.

Error logging for the SP uses BSD syslog and AIX Error Log facilities to report events on a node basis. The System Monitor and the SP Switch use this form of error logging.

Error log entries include a DETECTING MODULE string that identifies the software component, module name, module level, and the line of code or function that detected the event that was logged. The information is formatted based on the logging facility the user is viewing. For example, the AIX Error Log facility information appears as follows:

```
DETECTING MODULE
LPP=<LPP name>,Fn=<filename>, SID=<ID_level_of_the_file>,L#=<line number>
```

An extract from an actual error log entry is shown here:

```
DETECTING MODULE
LPP=PSSP,Fn=harml.d.c,SID=1.36,L#=2595,
```

The BSD syslog facility information appears as follows:

```
<timestamp, hostname, ID, PID>
LPP=<LPP name> Fn=<filename> <SID_level_of_the_file> L#=<line number>
```

An extract from the `/var/adm/SPlogs/SPdaemon.log` file is shown here:

```
Mar 4 20:08:58 sp4n01 Worm[10656]:
LPP=PSSP,Fn=TBSrecovery.c,SID=1.42,L#=1257,
```

11.4.1.1 Installing and Configuring the Error Log

Log management functions are built upon the `sysctl` facility, which uses the SP authentication services. Generating parallel AIX Error Log and BSD syslog reports and performing general log viewing require that the user issue the `kinit` command to be identified to the SP authentication services. All other log management commands additionally require that the user be defined as a principal in the `/etc/logmgt.acl` file. All users defined in this file

must also be placed in the authentication (PSSP or AFS) database as principals.

Important

Log management mostly consists of administrative tasks normally requiring root authority, and requires a user defined in the logmgt.acl file to execute commands as the root user.

Here is an example of the /etc/logmgt.acl file:

```
[sp4en0:/]# cat /etc/logmgt.acl
#acl#
#
# This sample acl file for log management commands
# contains a commented line for a principal
#
#_PRINCIPAL root.admin@HPSSL.KGN.IBM.COM
# Principal for trimming SPdaemon.log by cleanup.logs.ws
#
#_PRINCIPAL rcmd.sp4en0
#_PRINCIPAL root.admin@MSC.ITSO.IBM.COM
[sp4en0:/]#
```

The file contains entries for the user root as principal root.admin and giving root the authority to execute log management commands.

The log management server functions executed by sysctl are located in /usr/lpp/ssp/sysctl/bin/logmgt.cmds. During system initialization, an include statement for this file is added to the default sysctl configuration file /etc/sysctl.conf.

If you want to use an alternate sysctl configuration file, it must be updated with a statement to include the logmgt.cmds file, and the sysctld daemon must be restarted to activate the change.

Detailed information on configuring the error logs can be found in the redbook *RS/6000 SP: Problem Determination Guide*, SG24-4778, and in *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348.

11.4.1.2 SP Error Logs

The SP System uses the standard logs provided by both AIX and the public domain software it includes, as well as SP-specific logs. Some logs reside on the control workstation only, and some reside only on the SP nodes. Others

reside on both. A summary of the log files and their locations is shown in Chapter 4, “Error Logging Overview” of *IBM Parallel System Support Programs for AIX: Diagnosis Guide*, GA22-7350.

You may be asked to reference some of these files and send them to your IBM Support Center representative when diagnosing RS/6000 SP problems.

11.4.2 Problem Management (PMAN) Subsystem

The PMAN subsystem provides an infrastructure for recognizing and acting on problem events. PMAN is an Event Management (EM) client application, and is packaged with PSSP in the ssp.pman fileset and requires Kerberos authentication. The EM subsystem is discussed in 8.6, “Event Management (EM)” on page 200.

PMAN receives events from EM, and can react in one or more of the following ways:

- Send mail to an operator or administrator
- Notify all logged-on users via the `wall` command or opening a window on displays with a graphical user interface
- Generate a Simple Network Management Protocol (SNMP) trap for an enterprise network manager, such as Tivoli
- Log the event to AIX and BSD error logging
- Execute a command or script

This is represented diagrammatically in Figure 143.

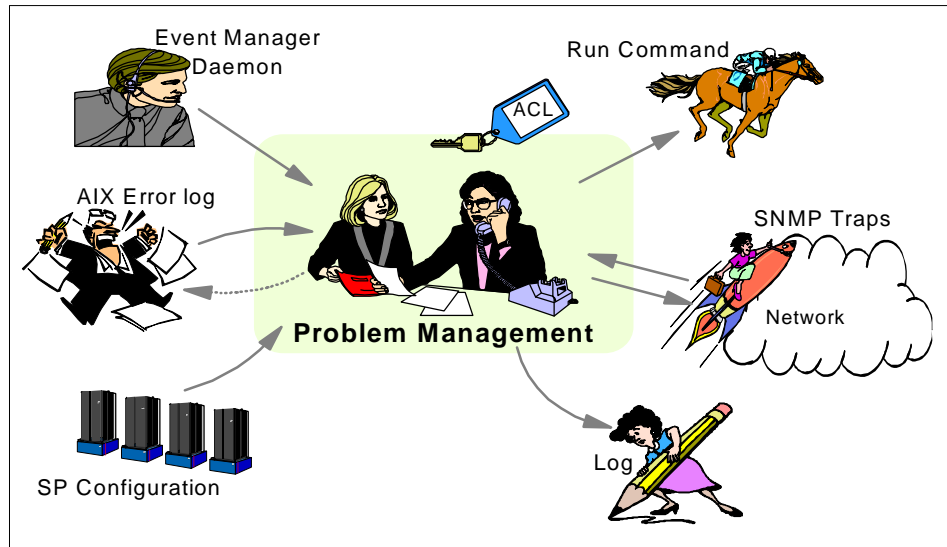


Figure 143. Problem Management Subsystem Design

Three daemons constitute the PMAN subsystem. The ways in which they inter-operate are shown in Figure 143. They are described as follows:

- **pmand** - This daemon interfaces directly to the EM daemon. The pmand daemon registers for events, receives them, and takes actions. PMAN events are stored in the SDR, and pmand retrieves them at initialization time.
- **pmanrmd** - If EM does not monitor the resource variable of interest, PMAN supplies its own resource manager daemon, pmanrmd, to access the resource variable's data. You can configure pmanrmd to periodically execute programs, scripts, or commands and place the results in one of EM's 16 user-defined resource variables. The pmanrmd daemon supplies the resulting data to the RMAPI. It also cooperates with pmand.
- **sp_configd** - This daemon creates SNMP traps from the event data in pmand, and is the interface for an enterprise network manager to access SP event, configuration, and resource variable data.

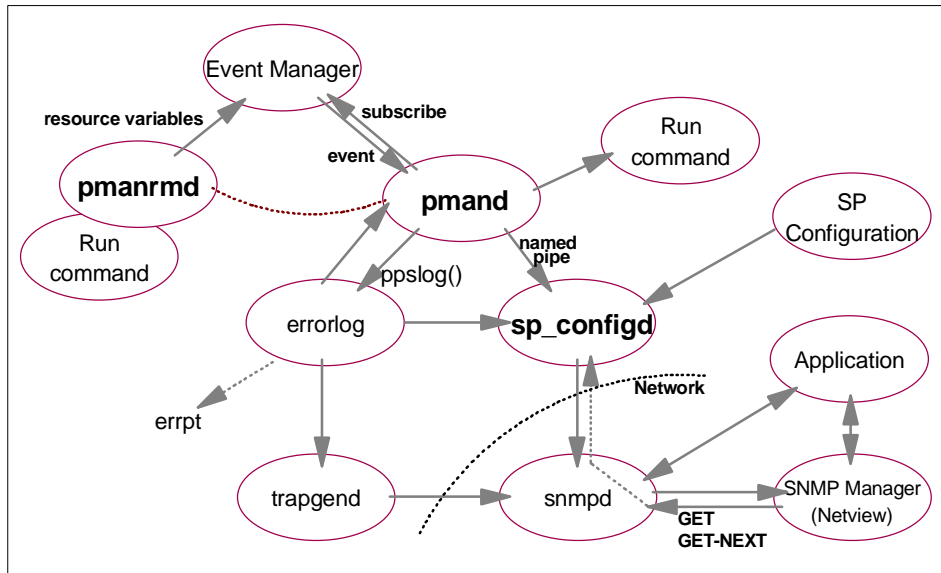


Figure 144. Problem Management Subsystem Daemons

The SP exploits AIX Error Notification Facility (AENF) to link the AIX error log and the PMAN subsystem. When a node is installed, an AENF object is created which sends all AIX Error Log entries to the pmand daemon. PMAN filters the entries based on subscriptions you define. The sp_configd daemon picks up SNMP-alertable AIX errors and passes them to snmpd for forwarding to the network manager (if this is installed in your environment).

By default, all PMAN daemons are started on all nodes and the control workstation. It is important to understand that PMAN is *not* a distributed application, but an Event Management (EM) client. The PMAN daemons on one node do not know about their counterparts on other nodes, and do not care. At initialization, each instance of pmand obtains PMAN event subscriptions from the SDR, so each node is aware of all PMAN events to be monitored. Although it is not mandatory to run the PMAN daemons everywhere, we do not recommend disabling them. PMAN must execute:

- Where you want an action to originate, given a specific event from EM
- Where you need custom resource monitoring, facilitated by the pmanrmd resource monitor

As an EM client, PMAN is an efficient infrastructure to respond to events across the system. For example, suppose your system-wide policy is to monitor for /tmp exceeding 95%. Instead of creating and maintaining the

required configuration on each node, use PMAN to subscribe to that event for all nodes. The EM distributed infrastructure monitors for you, and notifies your desired PMAN client (usually the node where the event occurs) to take appropriate action; furthermore, newly installed or re-installed nodes will automatically fall under the /tmp monitoring scheme. This is because the PMAN event is globally defined in the SDR, and PMAN is configured to run by default on all nodes.

The pmandefaults script sets the default events to be monitored, and is an excellent starting point for configuration of the PMAN subsystem. The script is documented in *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348. The script includes events of interest to most SP system environments, such as the following:

- The /var file system is more than 95% full.
- The /tmp file system is more than 90% full.
- An error log record of type PERM has been written to the AIX Error Log.
- The inetd daemon has terminated.
- The sdrd daemon has terminated (control workstation only).
- The sysctld daemon has terminated.
- The hrd daemon has terminated (control workstation only).
- The fsd (Worm) daemon has terminated (nodes only).
- Problem with noncritical power loss or fan failure (nodes only).

The subscribed events will result in an event notification being mailed to the root user on the control workstation when the specified event occurs. Events are defined for all nodes in the current system partition, and all events are monitored from the control workstation. An extract from this script follows.

```
#
# Watch /var space on each node in the partition
#
pmandef -s varFull \
  -e 'IBM.PSSP.aixos.FS.%totused:NodeNum=*;VG=rootvg;LV=hd9var:X>95'\
  -r 'X<70' \
  -c /usr/lpp/ssp/bin/notify_event \
  -C "/usr/lpp/ssp/bin/notify_event -r" \
  -n 0 -U root -m varFull
```

The syntax for this command is completely described in *IBM Parallel System Support Programs for AIX: Command and Technical Reference*, SA22-7351. The major flags you specify are:

- **-s** — This flag specifies that this is a subscribe request and the remaining flags define the Problem Management subscription. The name of this subscription is varFull.

-e — This flag specifies the Event Management resource variable, resource identifier and expression that define the event for which actions are generated for this Problem Management subscription. You can review the Resource Variable Descriptions by using the `haemqvar` command. For example, this command can be used to see how the resource variable can be used:

```
haemqvar IBM.PSSP.aixos.FS IBM.PSSP.aixos.FS.%totused "*" 
```

-r — This flag specifies the Event Management re-arm expression, which together with the resource variable, resource identifier and expression specified by the `-e` flag, defines the re-arm event for which actions are generated for this Problem Management subscription.

-c — This flag specifies a command to be executed when the event defined by the `-e` flag occurs. The command will be interpreted by the user's login program, so this command may contain additional arguments and shell metacharacters.

-C — This specifies a command to be executed when the re-arm event defined by the `Nr` flag occurs. The command will be interpreted by the user's login program, so the `RearmCommand` may contain additional arguments and shell metacharacters.

11.4.3 IBM Support Tools

There are some additional support tools available on RS/6000 SP environments.

11.4.3.1 Service Director

Service Director for RS/6000 can do automatic problem analysis and report hardware-related problems to IBM for service. Service Director is an application program that operates on all IBM RS/6000 machine types, including the RS/6000 SP. It is offered at no additional charge to customers as part of the IBM Warranty or IBM Maintenance Agreement.

Service Director is intended to be installed by trained IBM Customer Engineers/Service Support Representatives (CE/SSRs) or RS/6000 System Administrators. Basic AIX skills and the ability to use the System Management Interface Tool (SMIT) are required.

Service Director provides the following functions:

Automatic Problem Reporting — Reports problems to the IBM RETAIN (Problem Management System) via modem.

Manual Problem Reporting — Allows manual entry of problem information, then reports it to IBM RETAIN.

Automatic Problem Analysis — Product Support Applications (PSAs) will package additional information (if available) with the error information.

Definable Threshold Levels for Call Placement — Allows you to increase the threshold a specific error must reach before generating a service call. You can also tell Service Director to ignore specific errors or resources altogether.

Problem Reporting — Provides the ability for one RS/6000 to be the reporting server to IBM for all RS/6000s accessible from one of its TCP/IP networks. Provides a notify option that can send e-mail messages, informing users of Service Director activities.

Service Focal Point — Allows for a single focal point to view errors reported from any machine on the network.

Security — Service Director accesses only system data from diagnostics, system error logs, and Vital Product Data. The TTY configuration disables login capability on the port. The modem configuration disallows auto-answer. At *no* time does Service Director ever access customer data.

A customer-supplied modem and analog line is required. The automatic call function may not be offered in every country. Check with your IBM marketing representative for specific implementation conditions that apply in your country.

11.4.3.2 Gathering AIX Service Information

In many cases, IBM support personnel need to collect a substantial amount of detailed information to help isolate problems. To facilitate this process, AIX gives you the ability to collect service information for diagnosis by IBM Software Support. Included with the base AIX operating system is the `snap` command which assists you in compiling system configuration information quickly and easily. The `snap` command can be used to collect the following information:

- Configuration files
- Error and other log files
- Command line outputs

Once this information is compiled, you can view it and compress it for downloading to diskette or tape or for remote transmission. You may be asked by support specialists to execute the `snap` command to help them accurately identify your system problem.

Disk Space Requirements

Approximately 8 MB of temporary disk space is required when executing all of the `snap` options on an average system. If only one or two options are chosen, the disk space required will be substantially less, depending on the option. The program automatically checks for free space in the `/tmp/ibmsupt` directory or the directory specified with the `-d` flag. If there is not enough space, you will have to expand the file system. You can suppress this check for free space by using the `-N` option.

Output Directory

The default directory for the output from the `snap` command is `/tmp/ibmsupt`. If you want to name an optional directory, use the `-d` option with the path of the desired output directory. Each execution of the `snap` command appends to previously created files.

Options

The main options of the `snap` command are:

-g — Gathers the output of the `ls_lpp -L` command. Support specialists use the output to re-create your operating system environment if other problem determination techniques fail. The output is stored in `/tmp/ibmsupt/general/ls_lpp.L`.

Also, the `-g` flag gathers general system information and outputs it to `/tmp/ibmsupt/general/general.snap`.

-D — Gathers dump and `/unix` (assumes dump device to be `/dev/hd7`).

-a — Gathers information for all of the groups.

-c — Creates a compressed tar image of all files in the `/tmp/ibmsupt` directory tree (or other output directory).

Note: Other information that is not gathered by the `snap` command can be copied to the `snap` directory tree before executing the `tar/compress` option. For example, you may be asked by the support specialist to provide a test case that demonstrates the problem. The test case should be copied to the `/tmp/ibmsupt` directory. When the `-c` option of the `snap` command is executed, the test case will be included.

-o — Creates a tar file and downloads it to removable media.

-v — Displays the output of the commands executed by the `snap` command.

Before executing the `snap -c` or `snap -o` commands, any additional information required by the Support Center should be copied to the `/tmp/ibmsupt/testcase` directory (or an alternate directory).

The `snap -c` and `snap -o` commands are *mutually exclusive*. Do not execute both during the same problem determination session.

- The `snap -c` command should be used to transmit information electronically.
- The `snap -o` command should be used to transmit information on a removable output device.

11.4.3.3 Gathering SP-Specific Service Information

Specialized “snap-like” utilities are provided with PSSP software to collect log files and diagnostic information for specific SP hardware and/or software components. The generated compressed tar files can then be sent to your IBM Support Center representative for problem diagnosis and resolution.

css.snap

The `css.snap` script collects log files created by switch support code such as device drivers, the Worm, diagnostic outputs and so on, into a single package.

The completed output file is found in the `/var/adm/SPlogs/css` directory and will have the naming convention of `css.snap.<date-time>.tar.Z`.

The `css.snap` script is called automatically from the fault service daemon when certain serious errors are detected. However, it can also be issued from the command line when a switch or adapter related problem is indicated with:

```
# /usr/lpp/ssp/css/css.snap
```

Important

Be aware that `css.snap` uses a number of undocumented utilities to collect the information required. Some of these, such as the `read_regs` and `tbXdump` routines, can have a disruptive effect when used on a running system.

After running `css.snap` to collect diagnostic information, it is advisable to run `/usr/lpp/ssp/css/rc.switch` in order to reset or reload the switch adapter and eliminate the residual effects of these utilities.

vsd.snap

The `vsd.snap` script collects the data necessary to report SP VSD & RVSD related problems. There are a number of flags you can use to limit the considerable output that is generated by the default `-w ALL` option. The `vsd.snap` syntax is:

```
vsd.snap [-w VSD | -w RVSD | -w CFG | -w ALL] [-d output_directory]
```

where `-w` is used to select output information, and `-d` is used to specify the output directory; the default is `/tmp/vsd.snapOut`. The output file generated (a compressed tar file) has the filename format:

```
vsd.snap.node_number. ....out.tar.Z
```

where `.....` is a timestamp. You must ensure that there is sufficient free space available in the target file system for the output file to be created, otherwise the command exits with an appropriate message.

338 The RS/6000 SP Inside Out

Chapter 12. User Management

One of the challenges for the SP system administrator is managing the users in the system. Should a user be allowed to access to one node, some nodes or all nodes? The SP can be viewed as one logical unit, therefore users need to be defined across all nodes, with the same login name, the same login password, the same group characteristics, the same home directory and so on. This is only achieved by sharing the user database across the SP system. An SP system administrator therefore has the responsibility of maintaining consistent copies of files such as `/etc/passwd`, `/etc/group` and `/etc/security/passwd` across the nodes.

There are three methods available to maintain a consistent user database:

1. Manage each user individually over each node in an SP system. This is a time consuming effort because the system administrator has to be aware of any changes (such as a password change) to the user database on every node. Once a change is implemented on one node, the system administrator has to then update every node in the system.
2. Use the Network Information System (NIS), available with AIX. NIS is a three-tiered system of clients, servers and domain in which any changes to a set of defined common files (such as `/etc/passwd` and `/etc/group`) are automatically propagated throughout the system.
3. Use the SP File Collections facility provided with the PSSP software. File Collections defines and maintains sets of common files which are updated throughout the system at regular intervals.

It is recommended that SP system administrators choose method 2 or 3 to maintain a user database. Both options provide the system administrator the ability to manage the user database from a single point of control, for example, the control workstation. In an SP system with a large number of users and/or a large number of nodes, options 2 and 3 may be the only feasible choices. Option 1 requires a large manual effort, while options 2 and 3 make user management much more automated.

Two types of users can reside within an SP system. AIX users are those created through AIX on an individual node and reside only on that node. SP users are created through SP user management (SPUM) and can have access to every node. It is possible to have both types of users on a given node. This makes it difficult, if not impossible, to use file collections and NIS to manage the user database because the user database on each node is different from the other nodes. File collections and NIS are designed to manage one consistent copy of the user database across the system.

SP access control (SPAC) is provided with the PSSP software to give you the capability to use NIS or file collections to maintain a user database and restrict user login on particular nodes.

The automounter is a subsystem that mounts file systems on demand when they are first referenced and subsequently unmounts them after a period of inactivity. Its use enables users to maintain one home directory across the SP nodes.

This chapter begins with a look into the files within the AIX operating system which make up the user database. We then examine SPUM, SPAC and NIS. We conclude with a look at file collections and the automounter.

12.1 Managing User Databases

The purpose of user management is to maintain a user database across all nodes in an SP system. This enables the users to use the SP as one logical machine, despite its makeup of multiple, individual RS/6000s. User management tasks include:

- Adding user accounts
- Deleting user accounts
- Changing user account information, including the login password
- Listing user account information
- Controlling user logins

There are distinct sets of commands for managing AIX and SP users. To manage AIX users, use the AIX commands `mkuser`, `rmuser`, `chuser` and `lsuser`. To manage SP users, the PSSP software provides the SP user management (SPUM) commands. The two sets of command perform similar functions; the difference is the type of users they manage. SPUM is discussed in 12.1.2, “SP User Management (SPUM)” on page 341.

12.1.1 AIX: files (local)

To AIX, a user is defined by the existence of an entry in the `/etc/passwd` file. A user's password is kept in encrypted format in the `/etc/security/passwd` file. Users may be grouped together; group information is kept in the `/etc/group` and `/etc/security/group` files. Together, these four files form the foundation of an AIX user database.

Optional files to further tailor user accounts include `/etc/security/user`, `/etc/security/limits`, `/etc/security/environ`, `/etc/security/login.cfg`, `/etc/passwd.nm.idx`, `/etc/passwd.id.idx` and `/etc/security/passwd.idx`.

The `*.idx` files are password index files used to improve login performance.

The other files in `/etc/security` may or may not be used depending upon your system's requirements. A more in-depth look into user login control using the AIX files can be found in 7.2.4, "Login Control in AIX" on page 147.

If any of these optional files are used, they need to be included in the user database to be replicated across the system.

12.1.2 SP User Management (SPUM)

The PSSP software provides SP user management (SPUM) commands to enable an SP administrator to manage SP users. SPUM commands add and delete SP users, change account information, and set defaults for SP users' home directories. There are four SPUM commands: `spmkuser`, `spruser`, `spchuser` and `splsuser`. You do not have to use SPUM if you are familiar with the AIX commands and can manipulate them to execute the same functions as the SPUM commands.

You can enable SPUM both during and after the installation process. It is also possible to change the default SPUM settings after enablement. This is done through SMIT panels or the use of the `spsitenv` command. For the SMIT panel, run `smit enter_data` and select **Enter Site Environment Information**. Figure 145 on page 342 shows this SMIT panel.

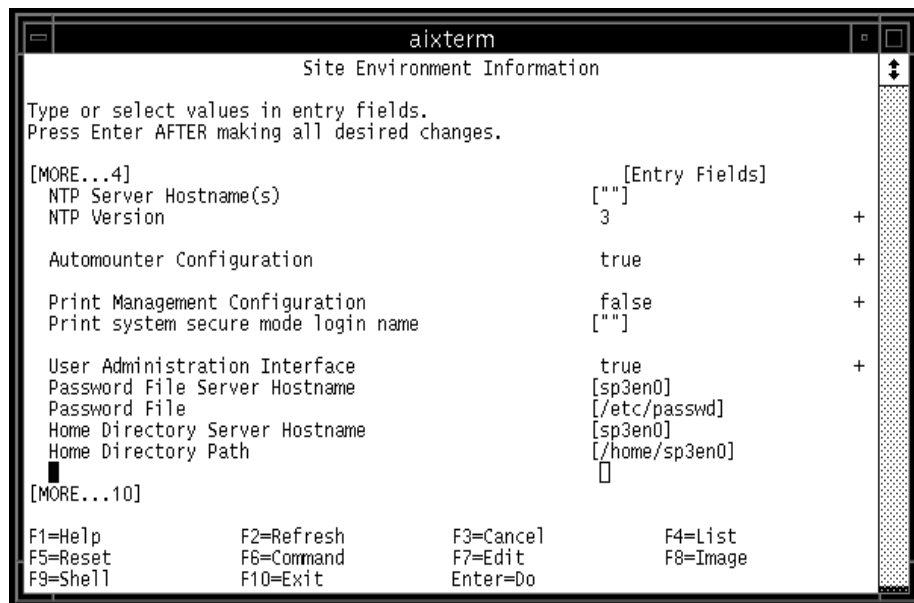


Figure 145. SMIT Panel for Controlling SPUM Settings

There are five fields for configuring SPUM:

- **User Administrative Interface** indicates whether you want to use SPUM or not.
- **Password File Server Hostname** indicates the hostname of the server which has the master password file. The default is set to the control workstation. You can set up another machine to serve the master password file, however, this machine cannot be one of the SP nodes.
- **Password File** is the path for the master password file. The default is `/etc/passwd`, the file that AIX uses. If you are using either file collections or NIS, you can choose to use another file.
- **Home Directory Server Hostname** is the name of the machine where the users' home directories reside. The default is the control workstation, although it can be set to any machine both inside and outside of the SP system. However, if the control workstation is not used, you need to ensure that security authentication has been set up. For the node, follow the authentication setup steps detailed in "Security Features of the SP System" in *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348. For a machine outside the SP system, you need to consider whether it is within the authentication realm of the SP or not. If

this machine is in the same realm, update the .klogin file of the machine to include the administrator's principal name. If not, you need to include a .rhosts file on the machine to permit the root user from other machines to perform the necessary actions against the home directories, for example, create a new directory when a new user is created.

- **Home Directory Path** is the default path of users' home directories. The default is /home/<control workstation name> and is stored in the SDR.

To add an SP user, you can use SMIT or the command `spmkuser`. To use SMIT, run `smit spmkuser`. Figure 146 shows this SMIT panel.

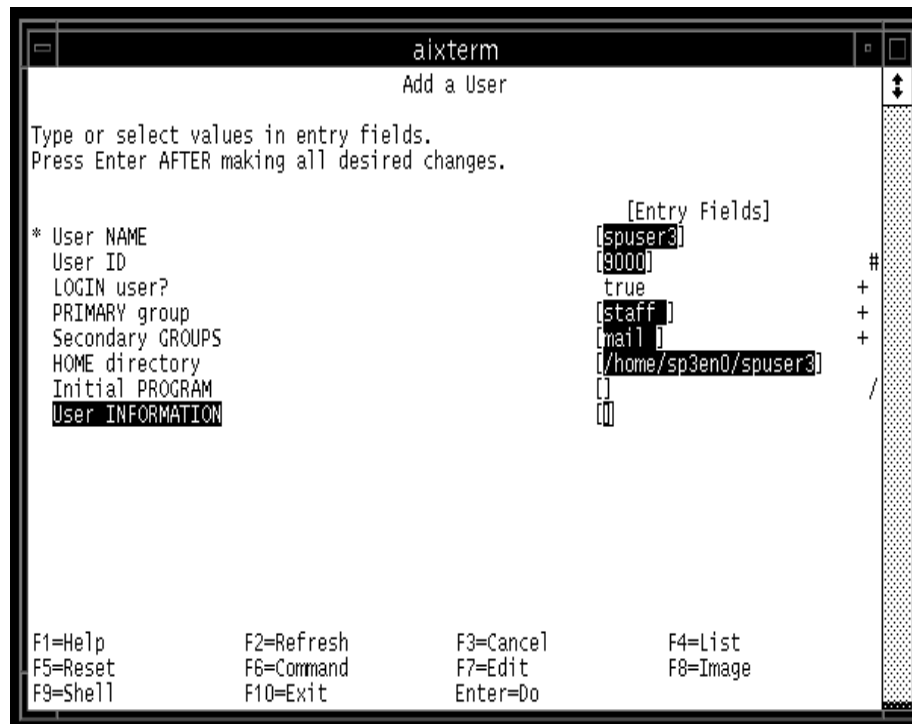


Figure 146. SMIT Panel for Adding SP Users

The only mandatory field is the name of the user. If all other fields are left blank, the system generates its own user ID, places the user in the staff group, does not put it into any secondary groups and creates a home directory based on the defaults specified when SPUM is turned on. If you specify a home directory path that is different than the default, this entry is used.

When a new user is added, the system generates a random password for the user and stores it in `/usr/lpp/ssp/config/admin/newpass.log`. This needs to be communicated to the user. The password in this file is not erased, even after the user has made changes to the password. It is therefore recommended that the contents of this file be deleted on a regular basis.

To delete users, you can use SMIT or `sprmuser`. To use SMIT, run `smit sprmuser`. Figure 147 shows this SMIT panel.

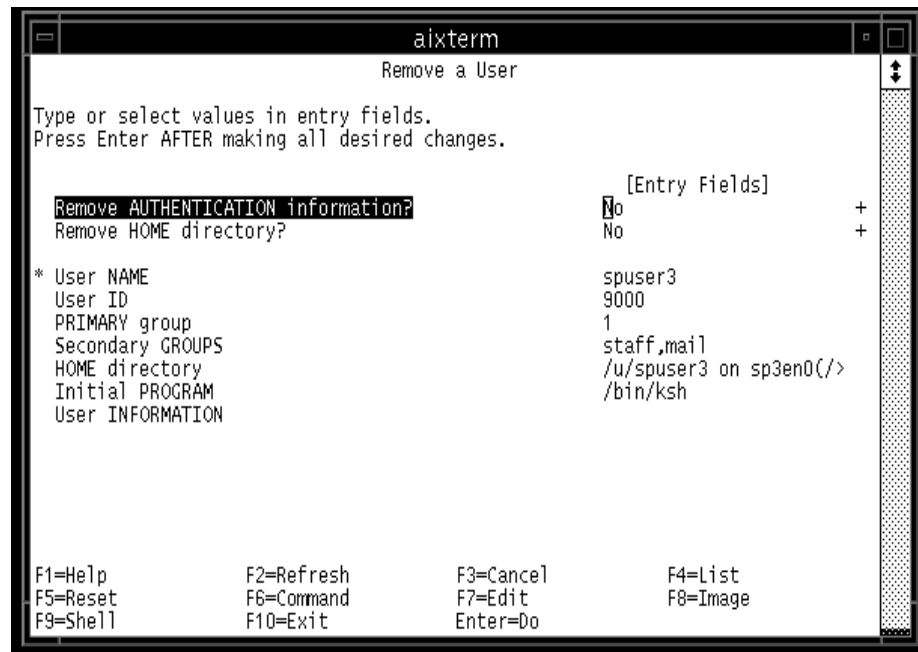


Figure 147. SMIT Panel for Deleting SP Users

The important options are the first two: **Remove AUTHENTICATION information?** and **Remove HOME directory?**. These are, by default, set to no to preserve data should the user need to be created again.

To change a user's information, you can again use SMIT or the command `spchuser`. Run `smit spchuser` to access the SMIT panel. Figure 148 on page 345 shows this SMIT panel.

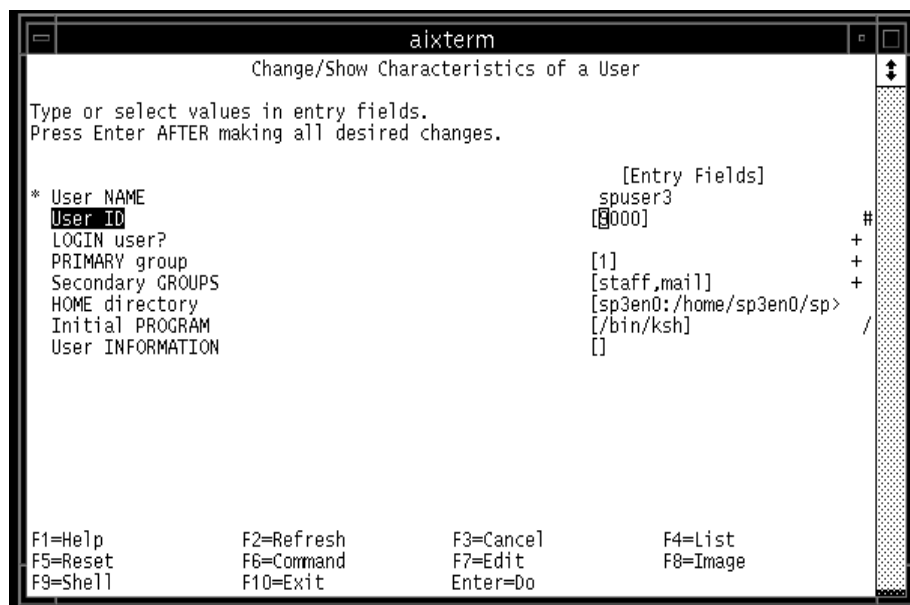


Figure 148. SMIT Panel for Changing SP User Information

Changes that you make here to an SP user can override the default settings specified when SPUM is initially enabled.

To list an SP user account and its information, you can run `spluser` or use SMIT by running `smit spchuser`. The same panel is used for listing and changing user information.

When SPUM is enabled, user password changes are handled in one of two ways, depending on whether NIS is used or not. If NIS is not in use, PSSP assumes that file collection is in use and restricts password changes to the machine that has the master copy of the `/etc/passwd` file. SPUM does this by linking the AIX password commands to SP password commands on the nodes. The commands that are modified are:

- `/bin/chfn`, which is linked to `/usr/lpp/ssp/config/sp_chfn`
- `/bin/chsh`, which is linked to `/usr/lpp/ssp/config/sp_chsh`
- `/bin/passwd`, which is linked to `/usr/lpp/ssp/config/sp_passwd`

The original AIX files are copied to new files which have the same name, but end in .aix. The SP password files tell users to log in to the appropriate machine to change their passwords.

User password changes when NIS is in use are discussed in 12.2, “Network Information System (NIS)” on page 348.

Attention

If both file collections and NIS are not used, but SPUM is enabled, user password changes are still restricted to the machine with the master /etc/passwd file. However, in this instance, after a change has been made, the system administrator needs to propagate the files /etc/passwd and /etc/security/passwd across the SP system.

12.1.3 SP Access Control (SPAC)

You may want to run either file collections or NIS to maintain a user database, but have a requirement to restrict SP user access on nodes. Or, if your SP system is used for a parallel application, there may be a need to restrict user login on the processing nodes to improve performance. In either case, you can use SP access control (SPAC). At the heart of SPAC is the command `spacs_cntrl`, which is run by the SP system administrator or by a job submission program on nodes where login control is required.

SPAC makes use of login and rlogin attributes in the /etc/security/user file to control a user's ability to log into particular nodes. It sets the attributes to either true or false depending on whether you want users to login (true) or not (false). The login attribute is used to control user local (serial line) logins while rlogin controls remote (network) logins. If you are using file collections, be sure to remove /etc/security/user from any file collections because it is unique among nodes. If you are using NIS, no action is needed because NIS by default does not handle this file.

Figure 149 on page 347 is a user's default setting as specified in /etc/security/user. Notice that the login and rlogin attributes are set to **true**.



```
aixterm
default:
  admin = false
  login = true
  su = true
  daemon = true
  rlogin = true
  sugroups = ALL
  admgroups =
  ttys = ALL
  auth1 = SYSTEM
  auth2 = NONE
  tpath = nosak
  umask = 022
  expires = 0
  SYSTEM = "compat"
  logintimes =
  pldwarntime = 0
  account_locked = false
  loginretries = 0
  histexpire = 0
  histsize = 0
  minage = 0
  maxage = 0
  maxexpired = -1
user (93%)
```

Figure 149. Sample of an Entry in /etc/security/user

SPAC controls the login and rlogin attributes by using `spacs_cntrl` with four keywords: block, unblock, allow and deny.

The block and unblock keywords are used by a system administrator to control login capabilities on a node. When a user is in block state on a node, both login and rlogin are set to equal false and all logins are disallowed. This is done by running `spacs_cntrl block <user>`. Similarly, if an user is in unblock state, login and rlogin are set to equal true and all logins are allowed. This is done by running `spacs_cntrl unblock <user>`.

Users may be controlled by `spacs_cntrl` individually or in a group. If you want to submit a group of users, run the script `/usr/lpp/ssp/samples/block_usr_sample` to build a file `/tmp/usr.input` which contain user IDs that can be blocked. For further details, refer to Chapter 6, "Managing User Accounts" in *IBM Parallel System Support Programs for AIX: Administration Guide, SA22-7348*.

The allow and deny keywords are used by a job submission program to control a user's state while executing a parallel job. The keywords are used on a transaction basis to update a user's state. A user can be in one of the following four states:

- Allowed login
- Denied login
- Allowed login after a deny request
- Denied login after an allow request

If a user's state is requested to change more than once using the allow and deny keywords, the file `spacs_data` is created to keep track of outstanding requests. When a change request is submitted, a check is made against the `spacs_data` file. If the request is the same as what is in the file, the request is not stored; instead, a counter is incremented. If the next request is opposite to what is in `spacs_data`, the counter is decremented, the user removed from `spacs_data`, and the `/etc/security/user` file is updated to reflect the change.

When a job submission program is using the allow and deny keywords to control user login on nodes, be careful that the system administrator does not run `spacs_cntrl block` or `spacs_cntrl unblock` on the same node. The block or unblock state automatically causes `spacs_cntrl` to clear the contents in the `spacs_data` file.

Further information for SPAC is in *IBM Parallel System Support Programs for AIX: Administration Guide, SA22-7348*.

12.2 Network Information System (NIS)

The main purpose of NIS is to centralize administration of files like `/etc/passwd` within a network environment.

NIS separates a network into three components: domain, server(s) and clients.

A NIS domain defines the boundary within which file administration is carried out. In a large network, it is possible to define several NIS domains to break the machines up into smaller groups. This way, files meant to be shared among for example, five machines, stay within a domain that includes the five machines, not all the machines on the network.

A NIS server is a machine that provides the system files to be read by other machines on the network. There are two types of servers: Master and Slave.

Both types keep a copy of the files to be shared over the network. A master server is the machine where a file may be updated. A slave server only maintains a copy of the files to be served. A slave server has three purposes:

- To balance the load if the master server is busy.
- To back up the master server.
- To enable NIS requests if there are different networks in the NIS domain. NIS client requests are not handled through routers; such requests go to a local slave server. It is the NIS updates between a master and a slave server that go through a router.

A NIS client is a machine which has to access the files served by the NIS servers.

There are four basic daemons that NIS uses: ypserv, ypbind, yppasswd and ypupdated. NIS was initially called yellow pages, hence the prefix yp is used for the daemons. The daemons work in the following way:

- All machines within the NIS domain run the ypbind daemon. This daemon directs the machine's request for a file to the NIS servers. On clients and slave servers, the ypbind daemon points the machines to the master server. On the master server, its ypbind points back to itself.
- The ypserv daemon runs on both the master and the slave servers. It is this daemon that responds to the request for file information by the clients.
- The yppasswd and ypupdated daemons run only on the master server. The yppasswd daemon makes it possible for users to change their login passwords anywhere on the network. When NIS is configured, the `/bin/passwd` command is linked to the `/usr/bin/yppasswd` command on the nodes. The yppasswd command sends any password changes over the network to the yppasswd daemon on the master server. The master server changes the appropriate files and propagates this change to the slave servers using the ypupdated daemon.

Important

NIS serves files in the form of maps. There is a map for each of the file that it serves. Information from the file is stored in the map, and it is the map that is used to respond to client requests.

By default, the following files are served by NIS:

- `/etc/ethers`
- `/etc/group`

- /etc/hosts
- /etc/netgroup
- /etc/networks
- /etc/passwd
- /etc/protocols
- /etc/publickey
- /etc/rpc
- /etc/security/group
- /etc/security/passwd
- /etc/services

Attention

By serving the /etc/hosts file, NIS has an added capability for handling name resolution in a network. Refer to *Managing NIS and NFS* by O'Reilly and Associates for detailed information.

To configure NIS, there are four steps, all of which can be done via SMIT. For all four steps first run `smit nfs` and select **Network Information Service (NIS)** to access the NIS panels, then:

Step 1 Choose **Change NIS Domain Name of this Host** to define the NIS Domain. Figure 150 on page 351 shows this SMIT panel. In this example, SPDomain has been chosen as the NIS domain name.

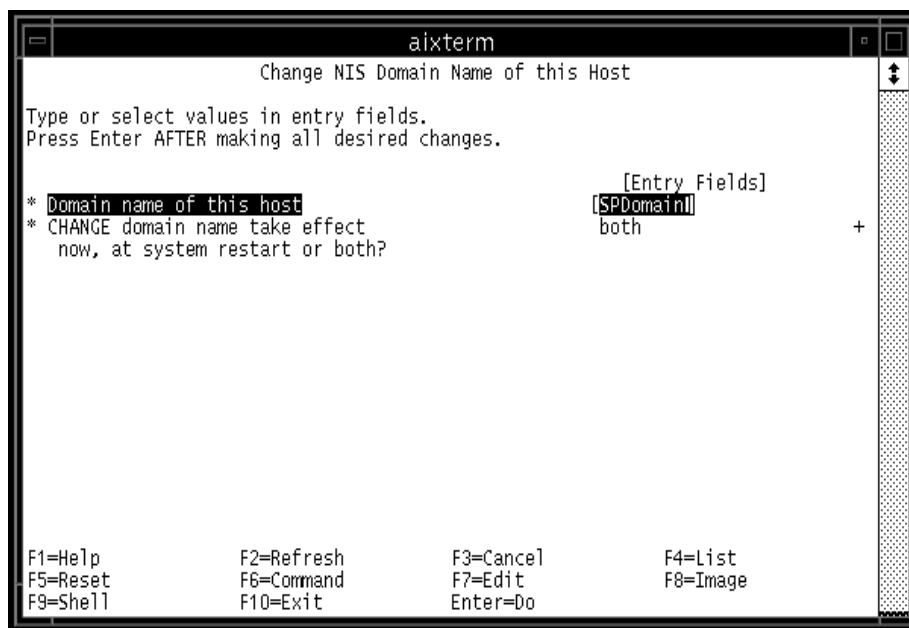


Figure 150. SMIT Panel for Setting a NIS Domain Name

Step 2 On the machine that is to be the NIS master (for example, the control workstation), select **Configure/Modify NIS** and then **Configure this Host as a NIS Master Server**. Figure 151 on page 352 shows the SMIT panel. Fill in the fields as required. Be sure to start the yppasswd and ypupdated daemons. When the SMIT panel is executed, all four daemons (ypbind, ypserv, yppasswd and ypupdated) are started on the Master server. This SMIT panel also updates the NIS entries in the local /etc/rc.nfs file.

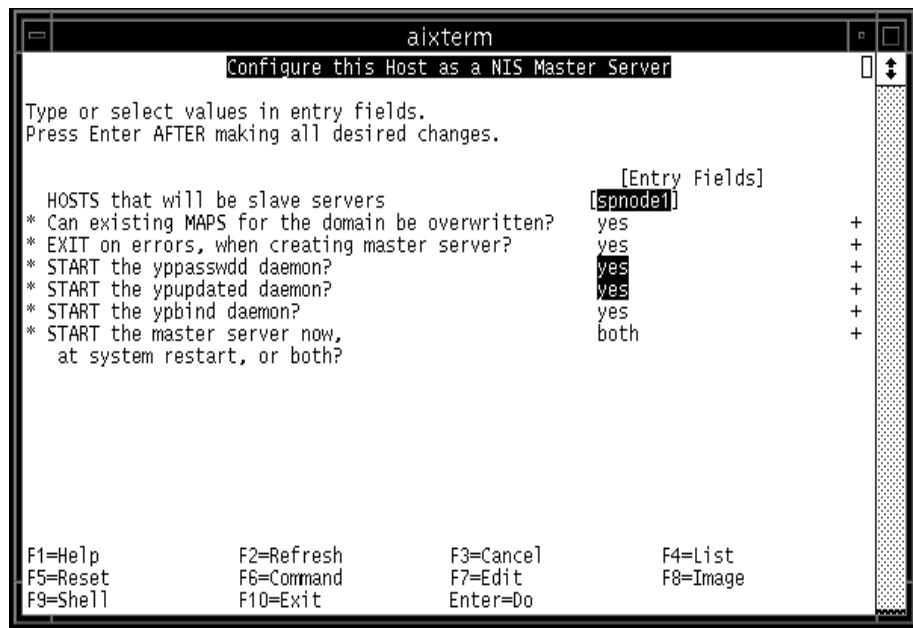


Figure 151. SMIT Panel for Configuring a Master Server

Step 3 On the machines set aside to be slave servers, go to the NIS SMIT panels and select **Configure this Host as a NIS Slave Server**. Figure 152 on page 353 shows the SMIT panel for configuring a slave server. This step starts the ypserv and ypbind daemons on the slave servers and updates the NIS entries in the local /etc/rc.nfs file(s).

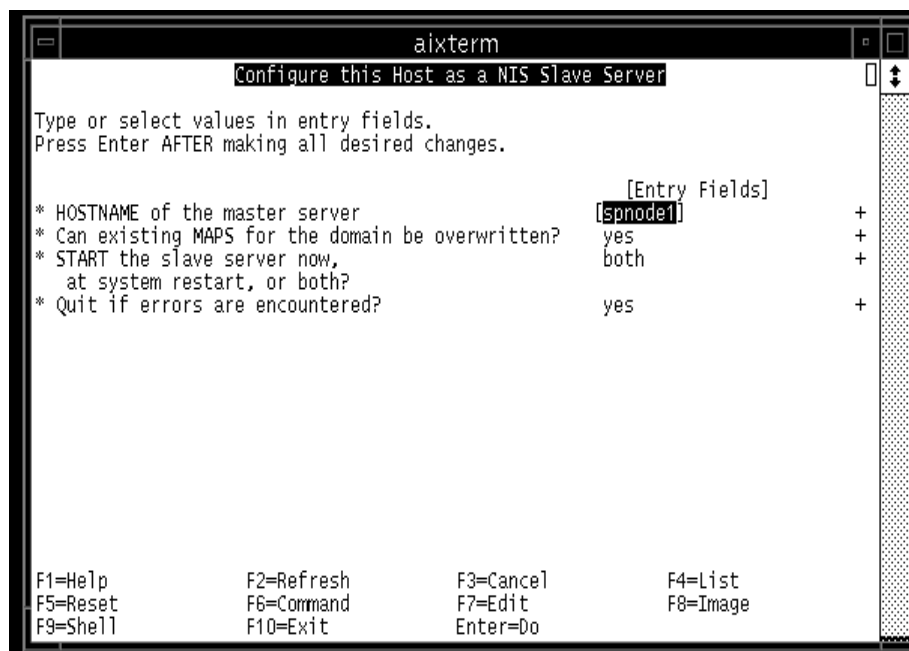


Figure 152. SMIT Panel for Configuring a Slave Server

Step 4 On each node that is to be a NIS client, go to the NIS SMIT panels and select **Configure this Host as a NIS Client**. This step starts the ypbind daemon and updates the NIS entries in the local `/etc/rc.nfs` file(s). Figure 153 on page 354 shows this SMIT panel.

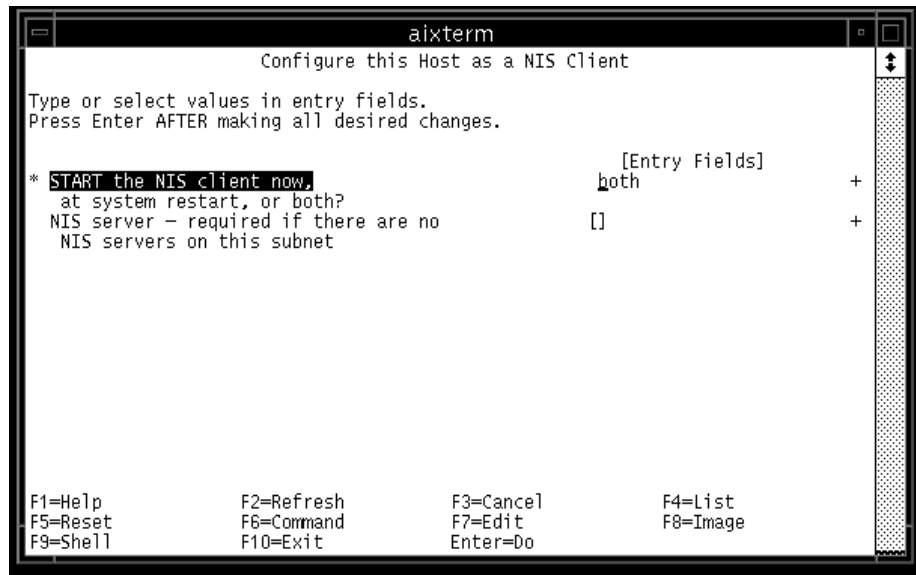


Figure 153. SMIT Panel for Configuring a NIS Client

Once configured, when there are changes to any of the files served by NIS, their corresponding maps on the master are rebuilt and either pushed to the slave servers or pulled by the slave servers from the master server. These tasks are done via the SMIT panel or the command `make`. To access the SMIT panel, select **Manage NIS Maps** within the NIS panel. Figure 154 on page 355 shows this SMIT panel.



Figure 154. SMIT Panel for Managing NIS Maps

Select **Build/Rebuild Maps for this Master Server** and then either have the system rebuild all the maps with the option **all**, or specify the maps that you want to rebuild. After that, return to the SMIT panel shown in Figure 154 and select either **Transfer Maps to Slave Servers** (from the master server) or **Retrieve Maps from Master Server for this Slave** (from a slave server).

12.3 File Collections

The File Collections facility is included with PSSP to group files and directories on multiple nodes into sets known as file collections. This simplifies their maintenance and control across an SP system. File Collections is installed by default on an SP system.

To turn off File Collections, run `smit enter_data`, select **Site Environment Information**, and choose **false** for the field **File Collection Management**. Figure 155 on page 356 shows this SMIT panel.

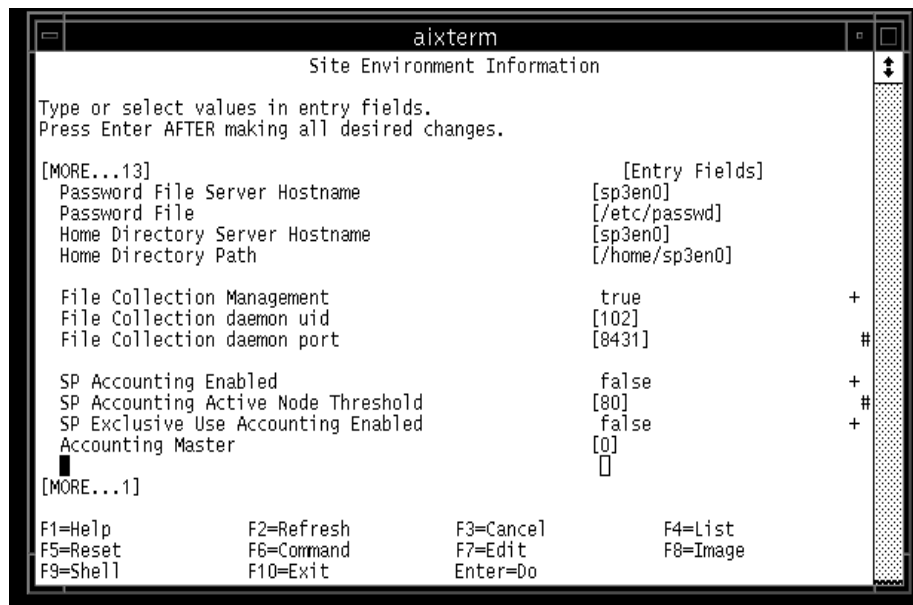


Figure 155. SMIT Panel for Setting File Collections

File collections are managed by the Perl program called `supper`, which in turn is based on the Software Update Protocol (SUP) and can be run as a command. A file collection has to be defined to `supper` so `supper` can recognize and maintain it. `supper` interacts with the file collection daemon `supman` to manage the file collections. `supman` is also installed as a unique userid for file collection operations and requires read access permission to all files that are to be managed as part of a file collection.

A file collection can be one of two types: primary or secondary. A primary file collection can contain a group of files or a secondary file collection. When a primary file collection is installed, the files in this collection are written to a node for usage. What if you want to use a node to serve a file collection to other nodes? This is made possible by using a secondary file collection. When a secondary file collection is installed on a node, its files do not get executed on the node, rather they are stored ready to be served out to other nodes.

File collections have two possible states: Resident or Available. A resident file collection is a group of files that are installed in their true locations and can be served to other systems. An available file collection is one that is not installed in its true location but is able to be served to other machines.

An installed secondary file collection can only be in the available state.

File collections uses a hierarchical structure to distribute files to nodes. The control workstation is defined by default to be the Master server, which is the machine where the "master" copy of a file is kept. It is only at this location that a file may be changed. Files are then distributed from the master server to all the defined boot/install servers to be distributed out to the nodes. If no boot/install servers are defined, file collections are served by the control workstation since the control workstation itself is a boot/install server.

You may change this hierarchy and use a boot/install server as a master server for one or some of the file collections. This way, you can maintain different copies of files on different groups of nodes. To implement this, you run `supper offline` on a boot/install server against a file collection. This prevents that file collection from being updated by the control workstation. Changes specific to the group of nodes served by the boot/install server can now be made on the boot/install server.

Attention

Password Changes

Recall that if NIS is not running, the password control files are changed so that all password updates are done on the master server. This may be a problem because you may not want all users to have access to the control workstation.

A workaround is to use the script `cw_restrict_login`, located in `/usr/lpp/ssp/config/admin`, to allow users to log into the control workstation only for the purposes of changing their passwords. The details for running this script can be found in Chapter 6, "Managing User Accounts" in *IBM Parallel System Support Program for AIX: Administration Guide*, SA22-7348.

12.3.1 Predefined File Collections

Four predefined file collections are shipped with PSSP:

- `sup.admin`

This file collection contains the files that define other file collections and the programs that load and manage file collections.

- `user.admin`

This file collection contains the AIX user account files: /etc/passwd, /etc/group, /etc/group, /etc/security/passwd, /etc/security/group and the login performance files /etc/passwd.nm.idx, /etc/passwd/id.idx, /etc/security/passwd.idx.

- power_system

This file collection contains files that are system dependent. By default, it only contains the node.root collection.

- node.root

This file collection contains files which are specific to the nodes. An example is a .profile different from the one on the control workstation that you want all users to use on the nodes.

The first three file collections are primary and resident on all nodes and boot/install servers. They are available to be served by the control workstation and boot/install servers. The node.root collection is a secondary file collection stored within the power_system file collection, available to be served by the control workstation and boot/install servers and resident on boot/install servers and nodes.

Attention

File Collections and NIS

It is possible to run both NIS and file collections, because NIS only handles the system administration files while file collections can be configured to handle other files. In this case, simply configure the user.admin file collection to not handle the user administration files of /etc/passwd,

12.3.2 File Collections File Structure

The file collection files are stored under /var/sysman. In particular, the master files which define and control file collections are stored in /var/sysman/sup. Each file collection has its own directory within /var/sysman/sup to store its unique characteristics. The actual files that make up the file collection may be stored elsewhere on the machine.

Figure 156 on page 359 summarizes the directory structure.

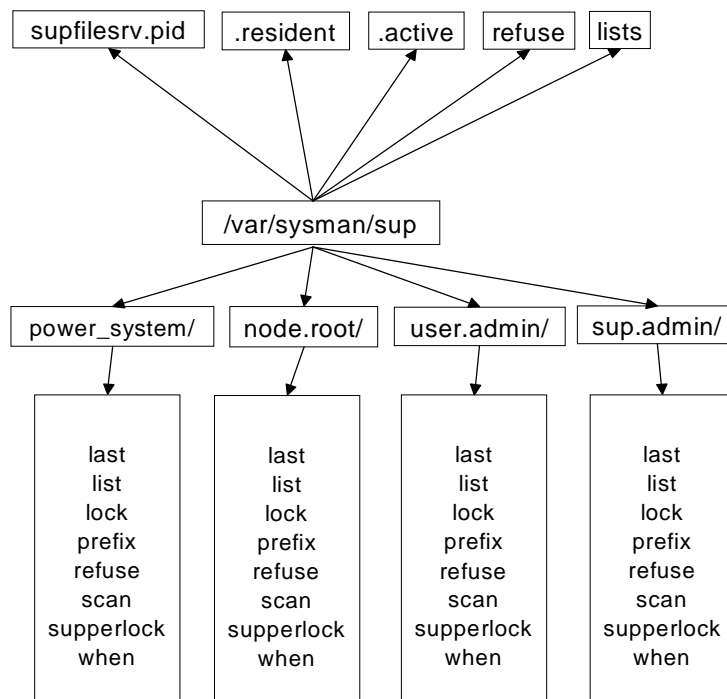


Figure 156. Summary of the File Collection Directory Structure

The files under /var/sysman/sup do the following:

- supfilesrv.pid holds the PID of the supfilesrv daemon, which serves the file collections on the system.
- .resident stores the names of all the file collections that can be served.
- .active describes the active volume group.
- refuse is a file that is useful on client systems which specifies the files to be excluded from file collection updates.
- lists is a file containing links to each of the individual file collection subdirectories.
- power_system, node.root, user.admin and sup.admin are all subdirectories set up to store the information regarding each file collection.

Within each file collection subdirectory, there is a set of “master files” which define the file collection:

last — is maintained by the system to hold a list of files and directories that have been updated.

list — holds the details of files that belong to the file collection. It is not necessary to define each individual file, you can use rules or wild cards. It is also not necessary to specify the full path of the files, just the path relative to the directory specified in the prefix file is sufficient.

lock — is an empty file that SUP uses to lock a collection and prevent multiple updates on a single file collection at one time.

prefix — specifies the starting point that `supper` uses when scanning the files that belong to a file collection.

refuse — is a system-generated file that contains a list of the files to be excluded from a supper update. The system creates this file based on the entries in the `/var/sysman/sup/refuse` file.

scan — is a system-created file that contains the list of files which make up the file collection, together with their permissions and time stamps. This file is populated when `supper` runs the scan process.

supper lock — is similar to the lock file and is used by supper to lock the file collection during updates.

when — is a system-maintained file that has the time of the last file collection update. It protects against accidental re-writes of files with older versions.

12.3.3 How File Collections Work

Here is how `supper` uses the file collection files. It can run one of three processes against a file collection: scan, install or update.

When `supper` runs the scan, it begins at the directory specified by the prefix file and traverses the directory tree to find all the files that meet the criteria in the list file. This process produces the scan file. The scan file is optional for the other supper processes. However, its presence can improve their performance since they no longer have to execute the equivalent of a scan.

The install process takes a file collection and copies the files onto the target machines for use, while the update process takes any changes in the files of a file collection and propagates them throughout the system. Both processes read from the scan file, if present, to identify the files to install or update. If a scan file is not present, `supper` performs a search to identify the files to install or update. An install or update also checks the `/var/sysman/sup/refuse` file to see if there are files to be excluded from the process.

During a `supper update`, the lock and supper lock files are used to block other processes from changing the files in the file collection. In addition, the subcommand checks the when file to ensure that it is not updating with an older version of the files.

Attention

The `supper update` process is a "pull" process. It is run from the clients to extract updated files from the master. For example, if a user changes their password on the control workstation, it is a node's responsibility to "pull" this change from the control workstation. Unlike NIS, the control workstation does not "push" a change.

When file collections is first installed, the supper update process is automatically included in the crontab file on the nodes in the SP. The frequency for an update is once per hour. If you want to change this, simply modify the crontab entry.

For the purpose of system administration, the process to run most frequently is the update process, to ensure that the files in a file collection are kept up-to-date. When you are updating files, make sure that you are writing to the copy of the file on the master server. Once the file is updated, run `supper scan <file collection name>` on the master server in case there have been changes to the number of files making up the file collection. Then, run `supper update <file collection name>` on the nodes to make sure that the nodes receive the change.

It is possible to run `supper` with a series of subcommands to extract information regarding a file collection. These are detailed in both *IBM Parallel System Support Programs for AIX: Administration Guide, SA22-7348* and *IBM Parallel System Support Programs for AIX: Command and Technical Reference, SA22-7351*.

12.3.4 Refusing Files in a File Collection

It is possible to exclude certain files from particular nodes during an update by `supper`. To do so, add the names of the specific files to the refuse file on the node's `/var/sysman/sup` directory. When `supper update` is run on the nodes, it checks `/var/sysman/sup/refuse` to see what files it needs to exclude from the update being executed.

As the files listed are passed over for update, the system writes the names of these files into `/var/sysman/<file collection name>/refuse`.

It may seem confusing to have two refuse files. However, this gives you the ability to confirm that the system has excluded the files you specified. Remember that you are to write only to `/var/sysman/sup/refuse`. The file `/var/sysman/<file collection name>/refuse` is written by the system and available for you to read to confirm that those are the files you requested to be excluded from the update.

12.3.5 Adding or Deleting Files in a File Collection

What if you want to add or delete a file within a file collection? These two situations require special considerations because you have to update the proper files in the file collection's directory.

The first step is to make sure the file is added or deleted on its server. You need to keep in the mind the exact location of the file relative to the directory that is specified in the prefix file. If you are adding a file, ensure that the file is readable by everyone else on the machine.

The second step is to run `supper scan` to update the scan file. This step may be optional since the scan file is not a requirement in a file collection. However, it is recommended that you do run this command because it increases the performance of the next command on large systems.

Run `supper update` on nodes to make this change effective.

12.3.6 Building a File Collection

To illustrate how to build a file collection, we are going to use an example where we create a file collection called `samplefc`. It includes four files in the `/sample` directory: `file1`, `file2`, `file3` and `file4`.

There are seven steps to building a file collection:

1. Identify the files that you want to collect. In the example, the files are `file1`, `file2`, `file3` and `file4`.
2. Create a File Collection directory on the master server and make sure that it is owned by `bin` and in the group `bin`. We therefore run:

```
cd /var/sysman/sup
mkdir samplefc
chown bin samplefc
chgroup bin samplefc
```

3. Create a list file in this directory to state the rules for including or excluding files in the file collection. The easiest way is to copy an existing list file

from another file collection and modify it to suit your needs. We copy and then edit the list file from another file collection to include only the line

```
always ./file*
```

This line in the list file instructs `supper` to always upgrade the files beginning with the word `file`.

4. Add a link to this file collection directory in the file `/var/sysman/sup/lists`.

```
ln -s /var/sysman/sup/samplefc/list /var/sysman/sup/lists/samplefc
```

5. Update the `/var/sysman/file.collections` file to include information regarding this file collection. We add the following lines:

```
# samplefc - a file collection to manage the files file1 file2 file3 and file4
primary samplefc - /sample - / sample 0 power yes
```

The first line is a comment. The fields in the second tells `supper` that this is a primary file collection, named `samplefc`, with no file system associated with it, which has files accessed through the `/sample` directory, with no specified file system size, having the scan process start at `/sample`, that all files in the directory should be evaluated, that this file collection runs on machines with the power architecture (RS/6000s) and that this file collection can be run on machines with different architecture.

There are further details in Chapter 5, “Managing File Collections” in *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348.

6. Update the `/var/sysman/sup/.resident` file to include your new file collection. We add in the line:

```
samplefc 0
```

where 0 indicates that this file collection is served by the control workstation.

7. Build the scan file in the file collection’s own directory by running:

```
supper scan samplefc
```

12.3.7 Installing a File Collection

There are two instances where it is necessary to install a file collection:

1. After you have created your own file collection
2. To set up boot/install servers to help distribute file collections to nodes

In both cases, it is a three-step process:

1. Run `supper update sup.admin` on the nodes and boot/install servers where you want to install the file collection. The `sup.admin` file collection contains

the files that identify and control all other file collections. You want the most up-to-date version.

2. Run `supper install <file collection name>` on each node or boot/install server that needs the file collection.
3. Add the `supper update <file collection name>` command to the crontab file to ensure that it is pulling down updates on a regular basis.

12.3.8 Removing a File Collection

A file collection cannot be globally removed; it can only be removed from the place where it is installed.

It is a two-step process:

1. Run `supper scan <file collection>` to create an updated scan file.
2. Run `supper remove <file collection>` to remove the file collection from the node.

If you want to remove the file collection from every node where it is installed, you have to run these steps on each node.

Attention

Do not remove any of the default file collections that are included when PSSP is installed. These are required by PSSP for its operation.

12.4 Managing the Automounter

The automounter is provided for an SP system administrator to manage the mounting of file systems across an SP system. The convenience of using the automounter is best illustrated with an example.

Consider an SP system with four nodes. You, as the system administrator, want to let all users have the capability to access all four nodes. You have chosen to use file collections to handle the management of the user database. What about the users' home directories? As with the user database, it is preferable to allow users to maintain one home directory across all four nodes.

To do this, one option is to store the home directories in an NFS file system on the control workstation and export it to the four nodes. Using NFS on its own, however, presents a problem. First, you have to decide whether each user's home directory is mounted as an individual file system, or whether the

entire /HOME directory is to be mounted on all nodes. Second, NFS requires the addition of the home directory entries into the file /etc/filesystems and you then have the added duty of maintaining this file whenever changes are made. Finally, NFS-exported directories are not automatically unmounted when they are no longer used. If you have chosen to mount each user's directory as a separate file system you have to find a way to unmount this directory every time they log out of a node to minimize local system hangs due to NFS server outages.

The automounter can take care of all of this for you. When the mounting of a file system is under automounter control, the automounter transparently mounts the required file system. When there is no activity to that file system for a prescribed period of time, the file system is unmounted.

The automounter uses map files to determine which file systems it needs to control. There is typically one map file per file system that you want to manage. It is also possible to use NIS maps instead of automounter maps to specify which file systems the automounter is to handle. The usage of NIS maps is discussed in greater detail in "Managing the Automounter" in *IBM Parallel System Support Programs for AIX: Administration Guide, SA22-7348*. The rest of this chapter is devoted to the use of automounter maps to handle the mounting of file systems.

There are three versions of the automounter available for use within an SP system. At PSSP 2.2 and below, the Berkeley Software Distribution (BSD) version of the automounter, called AMD, is used. From PSSP 2.3 onwards, the automounter included with the AIX operating system is used.

There are two versions of the automounter available with AIX. At AIX 4.3.0 and below, it is simply known as the "automounter". At AIX 4.3.1 and above, it is called the "AutoFS automounter". The biggest difference between the two automounters is that in AutoFS, there is a kernel implementation that separates the `automount` command from the `automountd` daemon. This makes it possible to change map information without having to stop and re-start the `automount` daemon process. For the purposes of our discussion, we are going to refer to the two automounters as automounter and AutoFS.

12.4.1 The BSD Automounter

At PSSP 2.2 and below, AMD is used to handle the mounting of user home directories and other file systems. AMD is different from the automounter and AutoFS in its configuration and capabilities. For example, while it is possible for all three to select multiple network interfaces to use for mounting a file system, only AMD permits you to prioritize the interfaces to always try to use

the switch interface to maximize performance. The automounter and AutoFS do not have this capability. This and other differences are detailed in *Inside the RS/6000 SP*, SG24-5145.

If you want to learn about the configuration of AMD, refer to *SP System Management: Easy, Lean and Mean*, GG24-2563.

12.4.2 The AIX Automounter

The automounter and the AutoFS automounter are kernel extensions which have an automountd daemon running in the background to service requests to mount file systems.

When a request is recognized, the automountd daemon uses NFS system calls to mount a directory onto the requesting machine. It then monitors the activities carried out against this file system. If it senses no activity against the file system after a specified amount of time (by default set to 5 minutes), automountd unmounts the file system from the requesting machine.

Both the automounter and AutoFS use map files residing on the server and clients to control file system mounts. There is a master map file, `/etc/auto.master` which contains entries for each file system to be controlled. Note that AutoFS by default looks for a map file named `/etc/auto_master` first. If it does not find an `/etc/auto_master`, then it looks for the `/etc/auto.master`. The use of the `/etc/auto_master` file is a requirement for AutoFS, and the `/etc/auto.master` is included in case users are migrating from the automounter and want to retain the old `/etc/auto.master`. Figure 157 on page 367 is an example of a `/etc/auto.master` file.

However, the automounter mounts the directories at `/tmp_mnt/<file system name>/<first sub-directory accessed>`, then creates a symbolic link from the local directory to the mounted file system.

In the example, `/share` is mounted at `/tmp_mnt/sample_fs/share` and then a symbolic link is created from `/sample_fs/dir_1` pointing to this directory.

The mount point `/tmp/*`, however, depends on which of `dir_2` and `dir_3` is first accessed. For example, if `dir_2` is accessed first, then the mount point for `/tmp/*` is `/tmp_mnt/sample_fs/dir_2/*`.

Then `/tmp/dir_2` is mounted as `/tmp_mnt/sample_fs/dir_2/dir_2`, with `/sample_fs/dir_2` symbolically linked to this directory.

Then `/tmp/dir_3` is mounted as `/tmp_mnt/sample_fs/dir_2/dir_3`, with `/sample_fs/dir_3` symbolically linked to this directory.

If `dir_3` is mounted first, then the mount point is `/tmp_mnt/sample_fs/dir_3/*`.

This can create problems for C-shell users because the C-shell `pwd` built-in command returns the actual path of a file directory. Since there is no guarantee that a file directory is always the same, C-shell commands cannot be based on the return value from `pwd`.

What if the client is the server? That is, we are trying to access a directory on the same machine on which it is residing. For example, we are trying to access `/sample_fs/dir_1` on the machine `sp3en0`. In this case, a local mount of `/share` to the mount point of `/sample_fs/dir_1` is carried out.

In an SP, the automounter and AutoFS write error messages to an error log file, `/var/adm/SPlogs/auto/auto.log`. In addition, you can use the daemon facility of `syslog` subsystem to record errors. By default, all `daemon.notice` and greater messages are written by `syslogd` to `/var/adm/SPlogs/SPdaemon.log`.

For more information on the AIX and AutoFS automounter, refer to *AIX 4.3 System Management Guide: Communications and Networks*, SC23-4127 and "Managing the Automounter" in *IBM Parallel Systems Support Programs for AIX: Administration Guide*, SA22-7348.

12.4.3 Examples (SP Setup, Migration, Coexistence)

This section is based on the use of AutoFS. The setup is the same whether you are using AutoFS or the automounter because they can both reference

the same map files. Differences are noted when they exist, otherwise, you can apply the same steps to both the automounter and AutoFS.

12.4.3.1 SP Setup

In an SP, AutoFS can be used to manage the mounting of both user home directories and other file systems across the system. The entry in the SDR which determines whether AutoFS is used or not is held in `amd_config`. You can change the value in this entry either by the command `spsitenv` or the **Automounter Configuration** field in the Site Environment Information SMIT panel. Figure 158 shows this SMIT panel.

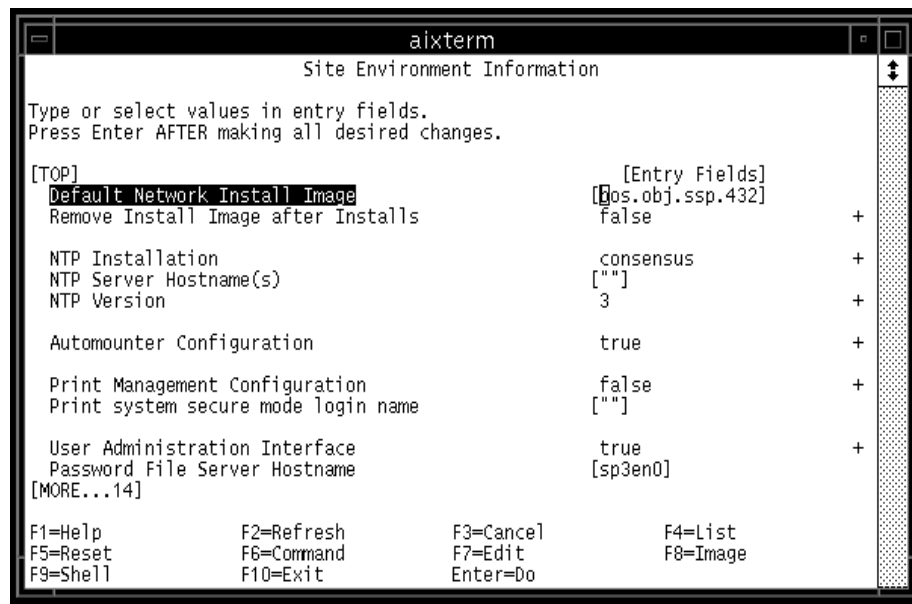


Figure 158. SMIT Panel for Turning On or Off the Automounter

When **Automounter Configuration** or `amd_config` is set to true, the system creates the necessary directories to store the configuration files for AutoFS and starts the automountd daemon. You will notice that the automountd daemon belongs to the `autofs` group. By default, AutoFS is set up to handle `/u` in an SP system, therefore, the `/etc/auto/maps/auto.u` file is installed and there is an entry for `/u` in the `/etc/auto.master` file.

If SPUM has also been turned on, the SP can then add, remove or change entries in the `auto.u` map file when SP users are added, removed or changed.


```

aixterm
# name (user "username" has a home directory /home/servername/username).
#
# username      servername:/home/servername:&
#
# This entry will link /u/username->/home/servername/username if the host is
# servername. If the host is not servername it will nfs mount
# servername:/home/servername on /tmp_mnt/u/servername and then link
# /u/username->/tmp_mnt/u/username
#
# If no entry is found for a user, and an entry with a key of "*" exists,
# that entry will be used as the default entry. For example, if the following
# is the last entry in this file:
# *          $HOST:/home:&
# and if no match is found for username, this entry will be used.
# It will link /u/username->/home/username on the current machine.
#
# The "netinst" entry is for the netinstall server. When automount is used,
# this entry must be defined for the netinstalls to work. Otherwise
# references to /u/netinst will not be resolved and netinstalls will fail.
#
netinst      $HOST:/home:&

spuser1 sp3en0:/home/sp3en0:&
spuser3 sp3en0:/home/sp3en0:&
[sp3en0:/etc/auto/maps]#

```

Figure 160. Sample /etc/auto/map/auto.u File

The control workstation (sp3en0) creates user directories under /home/sp3en0. For example, when spuser1 logs into a node, the user login facility searches for /u/spuser1, triggering the automountd daemon to mount /home/sp3en0/spuser1 over /u/spuser1.

Attention

If AutoFS is not enabled during the install process, and is subsequently enabled, then any user that has been created while AutoFS is disabled is not automatically added into the auto.u file. You have to manually add this into the file or use the `mkadment` command once the automounter is enabled.

12.4.3.2 Migration

If you are migrating from PSSP 2.2 and below to PSSP 2.3 and above, you have to convert the AMD map files into AutoFS map files. The two types of map files are not compatible. Since AutoFS and the automounter can share the same map files, there is no need for conversion above PSSP 2.3.

Assume that you are using AMD and have not customized any of the system created AMD configurations and map file at PSSP 2.2. During the migration, the system configuration process will detect that `amd_config` is set to true,

create a new `/etc/auto` directory structure, and add the default AutoFS configuration files. If SPUM is also enabled, then the system will convert your existing AMD map file into the automounter map file (`/etc/amd/amd-maps/amd.u` to `/etc/auto/maps/auto.u`).

The command which actually converts the map file is `mkautomap`. It, however, can only be used to convert the `amd.u` file.

If you have modified the `amd.u` file, it may not be properly converted by `mkautomap`. You have to check the `auto.u` file the command builds. If the file does not properly convert, you have to manually build the appropriate entries in the `auto.u` file.

If you have created your own AMD map files, you have to rebuild the equivalent automounter map files.

If you have customized AMD in any way, you have to consider whether AutoFS (automounter) is going to allow the same customization. AutoFS customization is described in "Managing the Automounter" in *IBM Parallel Systems Support Program for AIX: Administration Guide*, SA22-7348.

12.4.3.3 Coexistence

It is possible to run both AMD and AutoFS within the same SP system. The SP maintains both the AMD and auto sub-directories under `/etc`.

In this instance, the control workstation runs both versions of the automounter and the nodes run either one of the automounters. That is, if a node is running PSSP 2.2, it is going to continue to run AMD. If a node is running PSSP 2.3 and above, it is going to run either automounter or AutoFS, depending on the AIX level.

If you are at PSSP 2.3 and above, and you have some nodes that are AIX 4.3.0 and below and some nodes that are AIX 4.3.1 and above, you need to run the `compat_automountd`. This `automountd` can handle both automounter and AutoFS requests and is specifically included in for this purpose.

If you have to run a mixed environment of PSSP 2.2 and above, with AIX a mixture of AIX levels (both below 4.3.0 and above 4.3.1), you will run both `compat_automountd` in place of `automountd`, and AMD.

If SPUM is enabled, any SP user added, deleted or have information changed causes updates made to both the AMD and automounter map files.

When the user database gets sent to the nodes for updates (for example, by using File Collections), the nodes may receive and update both copies of the

maps. However, since only one of the automounters is running, only one of the maps is used for mounting the home directories.

374 The RS/6000 SP Inside Out

Chapter 13. Backup and Recovery

Having gone through the installation process listed in Chapter 10, “Installation and Configuration” on page 257, we now look at how we can protect what we have painstakingly installed. Failures due to hard disks, operating system corruption, irrecoverable data errors and so forth can cause much distress to an organization whose business depends on the availability of the system. No administrator will disagree on the importance of having good backups. Not only do they prevent data loss, they also reduce recovery time in the event of system failures.

There are various aspects of backup that need to be considered, namely, the operating system, user data and configurations.

Users familiar with AIX tend to agree that backing up of the AIX operating system is made easy with the implementation of the Logical Volume Manager (LVM). The SP system runs on AIX, which is why the backup procedure on the SP system is not much different from that of a non-SP system.

No doubt when we deal with the SP system, we are dealing with a cluster of RS/6000 systems. The strategies involved in backup and recovery can prove to be of utmost importance to an administrator dealing with hundreds of nodes. Poor management in this area can drastically increase backup time and result in unnecessary waste of archival space. Always ask yourself when you need to back up and what to back up. A quick guide as to when to back up the operating system would be when major changes are made. This includes installation of new software, upgrades, patches, changes to configurations, hardware changes especially to system planar, hard disks belonging to rootvg, power supplies and so forth.

This chapter briefly describes the backup and restore procedure for the RS/6000 SP system. There are products on the market that assist the backup of data. Examples include ADSM and Sysback from IBM, and Networker from Legato.

13.1 Backup/Recovery of rootvg

On an RS/6000 system, we have to ensure that proper backup is done for the volume group rootvg. On a failed system, the very first thing we need to recover is the rootvg volume from a reliable backup.

First we give a brief description of `mksysb` before we look into how we can back up the CWS using this command.

The `mksysb` command creates a bootable image by first putting a boot sector at the beginning of the media. This type of bootable image is usually done on tape media. If the `mksysb` image were created in a file, the file would not contain the boot image and would therefore not be bootable on its own.

Bootable images contain four parts, as shown in Figure 161.

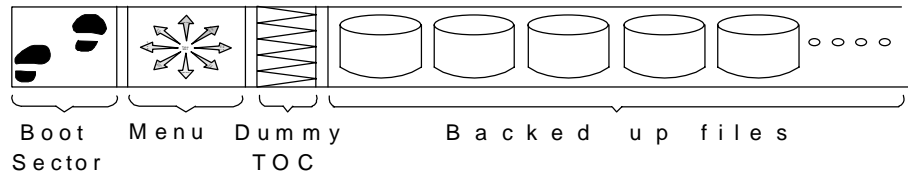


Figure 161. Contents of an `mksysb` Backup Image

The first part is the boot sector from which the node does a bootstrap boot, building the basic kernel in memory.

The second part is a menu that is displayed on the console of the node that is booting. This menu can be overridden with commands at the time the bootable image is created, or at the time of restoration by a certain file with a specific internal format.

The third part is a dummy table of contents, which is not used.

The last part contains the actual data files. This is the largest part. One of the first files that is in this data section is a file that contains the size, location and details about the various logical volumes and file systems that were mounted on the root volume group at the time of the `mksysb`. This file is looked at by the booting process to reconstruct the logical volume manager part of the system to what it was before so that the rest of the files on the tape can be restored into the same file systems that they were in.

A few points you have to consider about the `mksysb` command:

- It only backs up mounted file systems in the root volume group.
- If there is a database in a raw logical volume on the root volume group, it will not get backed up by this command.
- If there are other volume groups on a node, `mksysb` will not pick them up, and will not back them up, but the `savevg` command is designed specifically for non-root volume groups.

On a traditional RS/6000, the mksysb image is used for backup of systems, not cloning of systems - the mksysb image created on one type of machine will generally not be restorable on another type of machine. Note, however, that this is not true in the RS/6000 SP system: a mksysb image made on one node *will* restore on any other node.

13.1.1 Backup of the Control Workstation

The heart of the SP is the Control Workstation (CWS). It is through the CWS that node installation is made possible. This is because nodes, although they are basically RS/6000 machines put together in a frame, do not have consoles attached. The CWS provides these nodes with virtual consoles among other controls. Understanding the importance of the CWS, we now need to look at how we can take backups of this system. Although we talked about the CWS with so much hype, it is nothing more than a normal RS/6000 machine with additional software installed to function as a control master for the SP. We use the same `mksysb` and `savevg` commands to back it up. Care has to be taken, though, to back up the `/spdata` as well as the `/tftpboot` directories, as they could be created as file systems on a different volume group.

To back up the CWS, you can either use the command line or the SMIT screen. You have to be a root user to perform mksysb backups.

On the command line, use the following:

```
/usr/bin/mksysb -i -X /dev/rmt0
```

On the SMIT screen, use the following:

```
# smitty mksysb
```

See Figure 162 on page 378:



Figure 162. SMIT mksysb

The restore procedure of the CWS is dependent on the type of system you have as a CWS. Refer to your CWS system's installation guide about how to reinstall your system.

13.1.2 Backup/Recovery of Nodes

We now look at how we back up nodes. If we have gone through the installation of a new node, we understand that the image of the backup comes from a file and not from a bootable tape. This is made possible by the use of Network Install Manager (NIM). The PSSP package comes with a standard minimum mksysb image. The way this minimum image was made is similar to the method we will use to back up our nodes.

A mksysb to a file is very much the same as doing it onto tape media. Here, however, instead of specifying a tape drive, we specify a filename. The file can either reside on an NFS-mounted file system from the CWS or a local file system. For an NFS-mounted file system, the moment the mksysb is done, the file can be backed up to tape on the CWS. For a mksysb that is created on a local file system, you can either `ftp` or `rcp` it over to the CWS or any host that has a locally attached tape drive for back up.

Following is a simple example to illustrate the creation of a mksysb image of a node to a file system on the CWS. It is always best to ensure that the system is not in use before doing a backup.

On the CWS, export the file system for read and write access to the node:

```
[sp5en0:/] # /usr/sbin/mknfsxp -d '/spdata/sys1/install/images' -t 'rw'
-r 'sp5n09' '-B'
```

On the node, mount the NFS-exported file system from the CWS:

```
[sp5n09:/] # /usr/sbin/mount sp5en0:/spdata/sys1/install/images /mnt
[sp5n09:/] # df
Filesystem      512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4         16384           8224  50%      964   24% /
/dev/hd2        598016         141472  77%     9295  13% /usr
/dev/hd9var     65536           59920   9%      355   5% /var
/dev/hd3        65536           63304   4%       33   1% /tmp
/dev/hd1         8192            7840   5%       18   2% /home
sp5en0:/spdata/sys1/install/images 3907584 2687680 32%    8009
2% /mnt
```

Next, kick off the `mksysb` on the node.

```
[sp5n09:/] # /usr/bin/mksysb -i -X /mnt/mksysb.image.node9
```

The recovery of nodes is similar to the steps listed in the installation section and are not further elaborated on here. Refer to 10.3, “Frame, Node And Switch Installation” on page 267 for details on how to reinstall a node.

13.2 Backup/Recovery of System Database Repository (SDR)

We have been looking at backing up the system at an overall level. The CWS functions as a control for the whole SP complex. The System Database Repository (SDR) stores the configuration information about the whole system. Any changes made to the SDR need to be backed up. It is always advisable that, before any changes to the system configuration are made (like adding new nodes, frames, or changing hostnames, IP addresses), the SDR is backed up. If any errors were made, the SDR can easily be recovered without having to go through the whole `mksysb` restore procedure, which is time-consuming.

The command to back up the SDR is `SDRArchive`. It takes a snapshot of the SDR contents and saves it in a tar file located in the `/spdata/sys1/sdr/archives` directory on the CWS. The format of the backup is `backup.yyddd.hhmm`, where:

- `yy` denotes the last two digits of the year (for example, 99 for the year 1999).
- `dd` denotes the Julian date (039 is the 39th day of the year).
- `hh` denotes the hour in 24-hour format.

- mm denotes the minute.

It is quite normal for administrators to do many SDR archives and end up not knowing which archived file is usable. The `SDRArchive` command provides an `append_string` where you can add meaningful words to tell you more about the archived file. For example, if migrating PSSP to a higher version, you would want to archive your system's SDR with the following format:

```
# SDRArchive Before_Migrate
```

This will give you the following SDR archive file:

```
/spdata/sys1/sdr/archives/backup.99039.1001.Before_Migrate
```

If anything goes wrong with the SDR, we can restore the previous SDR information with the `SDRRestore` command:

```
# SDRRestore backup.99039.1001.Before_Migrate
```

The SDR should be backed up before making changes to it. Also, a few copies of SDR done on separate dates should be kept in case of corruption to any one of them. Although `SDRArchive/SDRRestore` is a handy tool to back up and restore the SDR, it should not be a replacement for a full system backup. There are misconceptions that the SDR is the heart of the SP and restoring it is all that is needed to restore a fully functional CWS. This is definitely an erroneous (and dangerous) idea. There are many components that make up the SP; the SDR is just one part of it.

13.3 Backup/Recovery of Kerberos Database

This section touches on the backup and restore of the Kerberos V4 database. For information on Distributed Computing Environment (Kerberos Version 5), refer to its related manuals.

On the Kerberos Authentication Server (KAS), which by default is the CWS, the Kerberos database is stored in the `/var/kerberos/database` directory. Just like the SDR mentioned earlier, backup of this database should be done prior to making changes to it. However, unlike the SDR where there are commands to facilitate the backup and restore, backing up Kerberos is done manually. The files that you would want to back up are as follows:

On KAS:

- `/var/kerberos/database/*`
- `/etc/krb-srvtab`

- /etc/krb.conf
- /etc/krb.realms
- /.k
- \$HOME/.klogin
- \$KRBTKFILE or /tmp/tkt<uid>

On the nodes:

- /etc/krb-srvtab
- /etc/krb.conf
- /etc/krb.realms
- \$HOME/.klogin
- \$KRBTKFILE or /tmp/tkt<uid>

For information relating to each file, refer to 7.4, “Managing Kerberos on the SP” on page 156.

To restore the Kerberos database, you will have to restore all the files you backed up.

There are instances where even restoring these files cannot help to recover the Kerberos database. In cases like those, you will need to rebuild the Kerberos database.

13.4 Backup/Recovery of User Specified Volume Group

Apart from the operating system, which by now we have an idea as to how to backup and restore, we next look at the data portion of the system. As a general rule, we normally advise administrators to separate their system files from their data files. The rootvg size should be kept to a minimum. Separate volume groups should be created to accommodate customers’ data. This helps in the sense that if anything goes wrong with the rootvg, only this volume group needs to be restored; the data residing on the other volume groups is kept intact.

Commands provided by AIX for backing up files are `tar`, `cpio`, `backup`, `rdump`. These can be used for journaled file systems (JFS). For databases, like Oracle, SAP, Sybase and so on, you will need to refer to the relevant database backup procedure; using conventional AIX backup commands may render your backups useless.

While `mksysb` is used for backing up the `rootvg`, `savevg` is used for backing up other volume groups. Like `mksysb`, only mounted journaled file systems are backed up with the `savevg` command.

The restoration of a `savevg` backup is accomplished with the `restvg` command. For information on usage of these commands, refer to AIX command documentation.

13.5 Proper Documentation

As with any setup, proper documentation is very important. We may have a very good backup procedure in place and good knowledge about the setup, but what is there to prevent us from forgetting about what we have done? What happens if the administrator leaves the company?

Any setup should be accompanied by documentation containing the procedure to setup, the structure of the setup, the problems faced during the setup, and so on. This documentation should be kept updated when there are changes to the system. The more you put into the documentation, the better position you are in to efficiently bring back a failed system.

Part 4. Application Enablers

384 The RS/6000 SP Inside Out

Chapter 14. Data Storage

SP nodes use AIX's Logical Volume Manager (LVM) as a building block for data storage. In turn, LVM uses a three-layered logical approach of volume group (VG), logical volume (LV), and file system to structure the data on the hard disks. For further details on LVM and how it works, refer to *AIX Storage Management*, GG24-4484.

Each SP node is capable of operating as a unique machine. This is because each node has its own operating system and stores its own data. Since each node uses LVM to store its data, however, the stored data is only accessible to the users and processes on the node. If users and processes on other nodes have to access this data, then file or disk sharing software has to be installed above LVM.

There are two strategies for sharing files. The first is to locate all shared files in file systems on one node, then use products like Network File Systems (NFS), Distributed File System (DFS) and Andrew File System (AFS). The second is to create a distributed environment, where data stored on any node is accessible by other nodes, using IBM's Virtual Shared Disks (VSD), Hashed Shared Disks (HSD), Recoverable Virtual Shared Disks (RVSD) or General Parallel File System (GPFS).

This chapter begins with a look at data stored locally on nodes, then examines the tools previously mentioned for sharing data among nodes, starting with NFS, DFS and AFS, then moving on to VSD, HSD and RVSD, and concluding with GPFS.

14.1 Local Disks

Each SP node stores data on disks physically attached to it. This data includes the AIX operating system, client portions of the PSSP software, application binaries and application data. This has both advantages and disadvantages.

Storing data on nodes permits an SP system to be used for applications like server consolidation. Each node has its own AIX operating system and can operate independently from other nodes to serve different applications. All nodes are centrally managed through one machine, the control workstation. Node locality is also scalable — as nodes are added, disks can be added linearly.

However, administrating these nodes can be a tedious task. There are multiple copies of the operating system and application software to install, backup and keep current. Each node can have unique configurations and tuned parameters that need to be maintained. If there are a large number of nodes, such tasks can be very time consuming.

Recall that an SP node uses LVM to handle the storage of data onto hard disks. LVM has many features which make it suitable for such a task. However, it does have one drawback: stored data is accessible to only the users and processes on the node on which the data is stored. In an SP system, this is a major inconvenience.

Consider one example: backups. The control workstation needs to make a backup copy of the data on one node, and then store it on a hard disk back on the control workstation itself. Data on the node is stored using LVM and can only be accessed by the local users and processes. If the control workstation wants to access this data, it uses a two-step process:

1. Use an application like telnet to log into the node and execute a backup.
2. Then use an application like ftp to download the backup image for storage on the control workstation.

If it is possible for the node and the control workstation to share data with each other, this process can be reduced to one step. The node “sees” the area where the control workstation stores backup images. All the control workstation has to do is initiate the backup on the node, and the node can store its backup image into this area.

Here is a second example. Consider a parallel application that runs on two nodes. If a user on one node wants to change a piece of data on the other node, LVM does not permit it. The user has to go through steps similar to the previous example to either obtain the data and process it locally, then move it back to the other node, or go to the other node and process it there. Either way, it is an inconvenience.

We can therefore conclude that LVM is insufficient as the sole means for handling data storage in an SP system.

In summary, the design that allows the storage of data on SP nodes has both advantages and disadvantages. Our concerns in this chapter center on the use of LVM within this design. We now look at the file sharing software which overcomes LVM's limitation.

14.2 Global File Systems

This section gives an overview of the most common *global* file systems. A global file system is a file system which resides locally on one machine (the file server), and is made globally accessible to many clients over the network. All file systems described in this section use UDP/IP as the network protocol for client/server communication (NFS Version 3 may also use TCP).

One important motivation to use global file systems is to give users the impression of a single system image by providing their home directories on all the machines they can access. Another is to share common application software which then needs to be installed and maintained in only one place. Global file systems can also be used to provide a large scratch file system to many machines, which normally utilizes available disk capacity better than distributing the same disks to the client machines and using them for local scratch space. However, the latter normally provides better performance, so a trade-off has to be made between speed and resource utilization.

Apart from the network bandwidth, an inherent performance limitation of global file systems is the fact that one file system resides completely on one machine. Different file systems may be served by different servers, but access to a single file, for example, will always be limited by the I/O capabilities of a single machine and its disk subsystems. This might be an issue for parallel applications where many processes/clients access the same data. To overcome this limitation, a *parallel* file system has to be used. IBM's parallel file system for the SP is described in 14.4, "General Parallel File System (GPFS)" on page 409.

14.2.1 Network File System (NFS)

Sun Microsystem's Network File System (NFS) is a widely used global file system, which is available as part of the base AIX operating system. It is described in detail in Chapter 10, "Network File System" of *AIX Version 4.3 System Management Guide: Communications and Networks*, SC23-4127.

In NFS, file systems residing on the NFS server are made available through an *export* operation, either automatically when the NFS startup scripts process the entries in the `/etc/exports` file, or explicitly by invoking the `exportfs` command. They can be mounted by the NFS clients in three different ways. A *predefined* mount is specified by stanzas in the `/etc/filesystems` file, an *explicit* mount can be performed by manually invoking the `mount` command, and *automatic* mounts are controlled by the `automount` command, which mounts and unmounts file systems based on their access frequency. This relationship is sketched in Figure 163 on page 388.

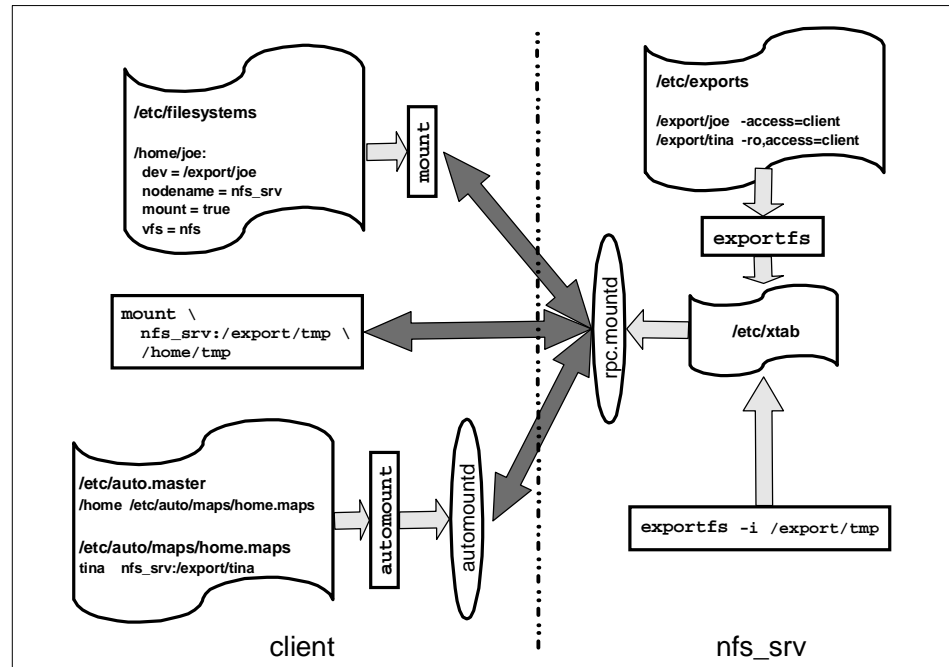


Figure 163. Conceptual Overview of NFS Mounting Process

The PSSP software uses NFS for network installation of the SP nodes. The control workstation and boot/install servers act as NFS servers to make resources for network installation available to the nodes, which perform explicit mounts during installation. The SP accounting system also uses explicit NFS mounts to consolidate accounting information.

NFS is often used operationally to provide global file system services to users and applications. Among the reasons for using NFS are that it is part of base AIX, it is well known in the UNIX community, it is very flexible, and it is relatively easy to configure and administer in small to medium-sized environments. However, NFS also has a number of shortcomings. We summarize them here to provide a basis to compare NFS to other global file systems.

Performance: NFS Version 3 contains several improvements over NFS Version 2. The most important change probably is that NFS Version 3 no longer limits the buffer size to 8 kB, improving its performance over high bandwidth networks. Other optimizations include the handling of file attributes and directory lookups, and increased write throughput by

collecting multiple write requests and writing the collective data to the server in larger requests.

- Security:** Access control to NFS files and directories is by UNIX mode bits, that means by UID. Any root user on a machine which can mount an NFS file system can create accounts with arbitrary UIDs and so can access all NFS-mounted files. File systems may be exported read-only if none of the authorized users needs to change their contents (like directories containing application binaries), but home directories will always be exported with write permissions as users must be able to change their files. An option for secure NFS exists, but is not widely used. Proprietary access control lists (ACLs) should not be used since not all NFS clients understand them.
- Management:** A file system served by an NFS server cannot be moved to another server without disrupting service. Even then, clients mount it from a specific IP name/address and will not find the new NFS server. On all clients, references to that NFS server have to be updated. To keep some flexibility, alias names for the NFS server should be used in the client configuration. These aliases can then be switched to another NFS server machine should this be necessary.
- Namespace:** With NFS, the client decides at which local mount point a remote filesystem is mounted. This means that there are no global, universal names for NFS files or directories since each client can mount them to different mount points.
- Consistency:** Concurrent access to data in NFS is problematic. NFS does not provide POSIX single site semantics, and modifications made by one NFS client will not be propagated quickly to all other clients. NFS does support byte range advisory locking, but not many applications honor such locks.

Given these shortcomings, we do not recommend the use of NFS in large production environments that require fast, secure, and easy to manage global file systems. On the other hand, NFS administration is fairly easy, and small environments with low security requirements will probably choose NFS as their global file system.

14.2.2 The DFS and AFS File Systems

There are mainly two global file systems which can be used as an alternative to NFS. The Distributed File System (DFS) is part of the Distributed Computing Environment (DCE) from the Open Software Foundation (OSF), now the Open Group. The Andrew File System (AFS) from Transarc is the base technology from which DFS was developed, so DFS and AFS are in many aspects very similar. Neither DFS nor AFS are part of base AIX; they are available as separate products. Availability of DFS and AFS for platforms other than AIX differs, but not significantly.

For reasons that are discussed later, we recommend using DFS rather than AFS except when an SP is to be integrated into an existing AFS cell. We therefore limit the following high-level description to DFS. Most of these general features also apply for AFS, which has a very similar functionality. After a general description of DFS, we point out some of the differences between DFS and AFS that justify our preference of DFS.

14.2.2.1 What is the Distributed File System

The DFS is a distributed application that manages file system data. It is an application of the Distributed Computing Environment (DCE) in the sense that it uses almost all of the DCE services to provide a secure, highly available, scalable, and manageable distributed file system.

DFS data is organized in three levels:

- Files and directories. These are the same data structures known from local file systems like the AIX Journaled File System (JFS). DFS provides a global namespace to access DFS files.
- Filesets. A DFS *fileset* is a group of files and directories which are administered as a unit, for example, all the directories that belong to a particular project. User home directories may be stored in separate filesets for each user, or may be combined into one fileset for a whole (AIX) group. Note that a fileset cannot be larger than an aggregate.
- Aggregates. An *aggregate* is the unit of disk storage. It is also the level at which DFS data is exported. There can be one or more filesets in a DFS aggregate. Aggregates cannot be larger than the logical volume in which they are contained.

The client component of DFS is the *cache manager*. It uses a local disk cache or memory cache to provide fast access to frequently used file and directory data. To locate the server that holds a particular fileset, DFS uses the *fileset location database (FLDB) server*. The FLDB server transparently accesses

information about a fileset's location in the FLDB, which is updated if a fileset is created or moved to another location.

The primary server component is the *file exporter*. The file exporter receives data requests as DCE Remote Procedure Calls (RPCs) from the cache manager, and processes them by accessing the local file systems in which the data is stored. DFS includes its own Local File System (LFS), but can also export other UNIX file systems (although with reduced functionality). It includes a *token manager* to synchronize concurrent access. If a DFS client wants to perform an operation on a DFS file or directory, it has to acquire a token from the server. The server revokes existing tokens from other clients to avoid conflicting operations. In this way, DFS is able to provide POSIX single-site semantics.

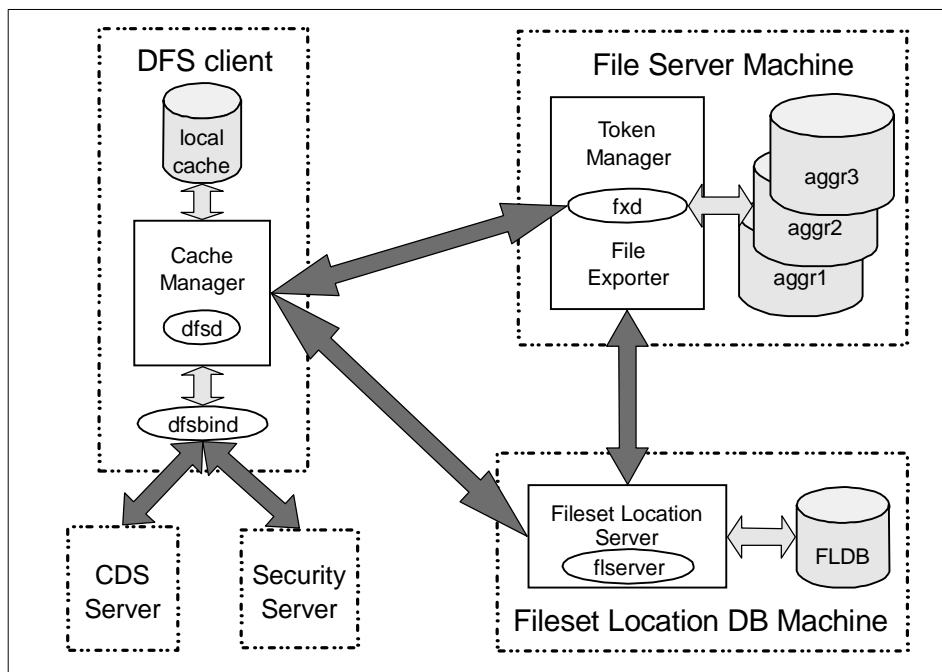


Figure 164. Basic DFS Components

Figure 164 shows these DFS components. This is an incomplete picture; in reality there are many more DFS components (such as the replication server) and various management services like the fileset server and the update server. More detailed information about DFS can be found in *Distributed Computing Environment for AIX, Version 2.2: Quick Beginnings*, SC23-4188.

and *IBM DCE for AIX: DFS Administration Guide and Reference*, provided with the product in soft copy only.

The following list summarizes some key features of DCE/DFS, and can be used to compare DFS with the discussion in 14.2.1, "Network File System (NFS)" on page 387.

- Performance:** DFS achieves high performance through client caching. The client to server ratio is better than with NFS, although exact numbers depend on the actual applications. Like NFS, DFS is limited by the performance of a single server in the write case. However, replication can help scale read-only access.
- Security:** DFS is integrated with the DCE Security Service, which is based on Kerberos Version 5. All internal communication uses the authenticated DCE RPC, and all users and services which want to use DFS services have to be authenticated by logging in to the DCE cell (except when access rights are explicitly granted for unauthenticated users). Access control is by DCE principal; root users on DFS client machines cannot impersonate these DCE principals. In addition, DCE Access Control Lists can be used to provide fine-grained control; they are recognized even in a heterogeneous environment.
- Management:** Since fileset location is completely transparent to the client, DFS filesets can be easily moved between DFS servers. Using DCE's LFS as the physical file system, this can even be done without disrupting operation. This is an invaluable management feature for rapidly growing or otherwise changing environments. Because there is no local information on fileset locations on the client, administering a large number of machines is much easier than maintaining configuration information on all of these clients.
- Namespace:** DFS provides a global, worldwide namespace. The file system in a given DCE cell can be accessed by the absolute path `././cell_name/fs/`, which can be abbreviated as `/:` (slash colon) within that cell. Access to foreign cells always requires the full cell name of that cell. The global name space ensures that a file will be accessible by the same name on every DFS client. The DFS client has no control over mount points: filesets are mounted into the DFS namespace by the servers. Of course, a client may

use symbolic links to provide alternative paths to a DFS file, but the DFS path to the data will always be available.

Consistency: Through the use of a token manager, DFS is able to implement complete POSIX single site read/write semantics. If a DFS file is changed, all clients will see the modified data on their next access to that file. Like NFS, DFS supports byte range advisory locking.

Operation: To improve availability, DFS filesets can be replicated, that is, read-only copies can be made available by several DFS servers. The DFS server processes are monitored and maintained by the DCE *basic overseer server* (BOS), which automatically restarts them as needed.

In summary, many of the problems related to NFS either do not exist in DFS, or have a much weaker impact. DFS is therefore more suitable for use in a large production environment. On the other hand, DCE administration is not easy and requires a lot of training. The necessary DCE and DFS licenses also add extra cost.

14.2.2.2 Differences Between DFS and AFS

Apart from the availability (and licensing costs) of the products on specific platforms, there are two main differences between DFS and AFS: the integration with other services, and the mechanism to synchronize concurrent file access. The following list summarizes these differences:

Authentication AFS uses Kerberos Version 4, in an implementation that predates the final MIT Kerberos 4 specifications. DCE/DFS uses Kerberos Version 5. For both, the availability of other operating system services (like telnet or X display managers) that are integrated with the respective Kerberos authentication system depends on the particular platform.

Authorization DFS and AFS ACLs differ, and are more limited in AFS. For example, AFS can only set ACLs on the directory level, not on file level. AFS also cannot grant rights to a user from a foreign AFS cell, whereas DFS supports ACLs for foreign users.

Directory Service DCE has the Cell Directory Service (CDS), through which a client can find the server(s) for a particular service. The DFS client uses the CDS to find the Fileset Location Database. There is no fileset location information on the client. AFS has no directory service. It relies on a local

configuration file (/usr/vice/etc/CellServDB) to find the Volume Location Database (VLDB), the Kerberos servers, and other services.

- RPC** Both DFS and AFS use Remote Procedure Calls (RPCs) to communicate over the network. AFS uses Rx from Carnegie Mellon University. DFS uses the DCE RPC, which is completely integrated into DCE, including security. AFS cannot use dynamic port allocation, DFS does so by using the RPC *endpoint map*.
- Time Service** DFS uses the DCE Distributed Time Service, discussed in Chapter 6, "Time Service" on page 131. AFS clients use their cache manager and NTP to synchronize with the AFS servers.
- Synchronization** Both DFS and AFS use a token manager to coordinate concurrent access to the file system. However, AFS revokes tokens from other clients when *closing* a file, whereas DFS revokes the token when *opening* the file. This means that DFS semantics are completely conforming with local file system semantics, whereas AFS semantics are not. Nevertheless, AFS synchronization is better than in NFS, which does not use tokens at all.

It is obvious that DFS is well integrated with the other DCE core services, whereas AFS requires more configuration and administration work. DFS also provides file system semantics that are superior to those of AFS. So unless an existing AFS cell is expanded, we recommend that you use DFS rather than AFS to provide global file services.

14.3 Shared Disks

A serial application can exceed the I/O capacity of a single node, even after the application's data has been spread across many disks and I/O adapters of the node. Parallel applications need access to data that is spread across many nodes to improve I/O performance.

If we can provide a node with the capability to access data residing on any other node in the SP system, we satisfy the access need of parallel applications and also offer a possibility for improved performance of serial applications.

IBM makes this possible with its family of offerings based on the Virtual Shared Disks (VSD) technology.

14.3.1 Virtual Shared Disks (VSD)

VSD allows data stored in raw Logical Volumes (LVs) on one node to be globally accessible over an IP network. A VSD may be viewed as an LV that other nodes may access as though it has been locally configured. There may therefore be multiple VSDs defined and configured on any given node.

This technology is designed with applications like Oracle's parallel database in mind. Oracle's database architecture is highly centralized. Any processing node must be able to "see" the entire database. Therefore, in Oracle's parallel database, all nodes must have access to all disks, regardless of where these disks are physically attached.

A node which has VSDs defined and configured is called a *server node*. A node which accesses other nodes' VSDs is called a *client node*. A node may be both a client node and a server node.

VSD is designed to run over any IP network. We do, however, strongly recommend that the switch network be used for superior performance.

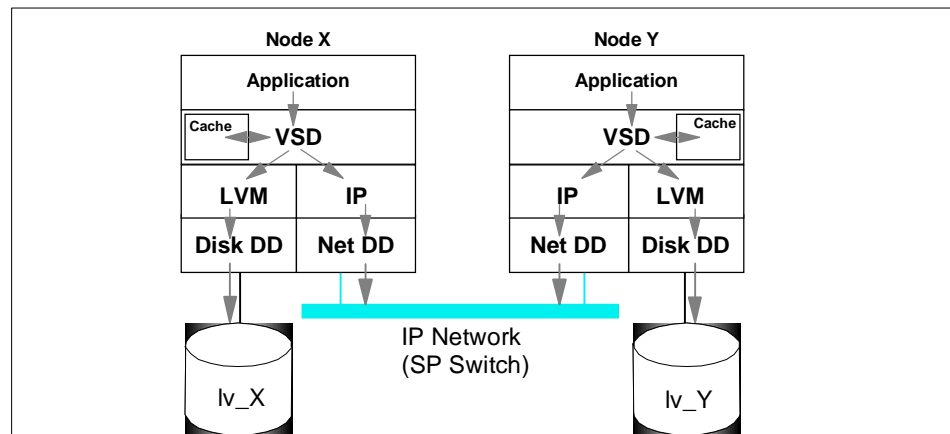


Figure 165. VSD Architecture

As shown in Figure 165, VSD provides a device driver loaded as a kernel extension which sits below the application layer, and above the LVM and IP device drivers. When the application requests a piece of data, this request is sent to the VSD device driver, which then looks for the data in the appropriate place.

For example, the application on Node X is looking for a piece of data and passes this request to the VSD device driver. Node X's VSD driver can then look for the data in one of three places:

1. VSD cache

The VSD cache stores all I/O requests in case that piece of data is required later. This area is shared by all VSDs defined and configured on one node. The data here is stored in 4 KB blocks, a size optimized for Oracle's parallel database program. Note that this is an optional feature. If your I/O requests involve operations much larger than 4 KB, using this cache may prove to be counterproductive because of the additional costs of fragmenting it into the 4 KB blocks.

2. lv_X

The VSD device driver passes the request to Node X's LVM and disk device drivers to fetch the data.

3. Node Y

The VSD device driver passes the request through the IP and network device drivers to Node Y. Node Y's IP and network device drivers then pass the request to Node Y's VSD device driver, which finds the data, either in Node Y's VSD cache (if applicable) or in lv_Y. The data is fetched, and then passed back to Node X in a similar fashion, making use of the IP and network device drivers.

VSD device drivers use their own stripped-down IP protocol for performance reasons. In addition, VSD uses unique buffers, buddy buffers and pbufs to handle the network transmissions. Detailed information on these buffers, including how to tune them, can be found in *IBM Parallel System Support Programs For AIX: Managing Shared Disks, SA22-7349*.

Because VSDs are LVs, they do not have a file locking mechanism to preserve data integrity. This task falls upon the application that is using the VSDs.

VSDs can be defined, configured and managed by the command line, or by SMIT panels, or by the graphical user interface Perspectives. Their definitions are stored in the SDR on the control workstation.

The VSD software is an optional feature of PSSP is installed on the nodes and the control workstation. The VSD software consists of the filesets shown in Table 19 at PSSP 3.1:

Table 19. VSD Fileset Names and Descriptions

Fileset	Description
vsd.cmi	VSD SMIT panels
vsd.vsdd	VSD device driver

Fileset	Description
vsd.sysctl	VSD sysctl commands

Additional information on VSD filesets can be found in *PSSP 3.1 Announcement*, SG24-5332.

To build VSDs in a SP system, perform the five general steps as follows:

1. Install the filesets on the nodes you intend to run VSD.
2. Designate the VSD nodes.

Every node that is going to be involved with VSDs, whether as a client or as a server, has to be designated as a VSD node in the SDR on the control workstation. In Perspectives, this is done by choosing the action **Designate as an IBM VSD Node**.

- For SMIT panels, run `smit vsd_data` and select the **VSD Node Information** option.
- From the command line, use `/usr/lpp/csd/bin/vsdnode`.

It is more advantageous to use the Perspectives interface or SMIT panels because both automatically bring up a list of tuning parameters that you can set. If the command line option is used, the settings have to be included when `vsdnode` is run.

Figure 166 on page 398 is an example of the Perspective panel used for designating VSD nodes (in this case, nodes 1 and 15).

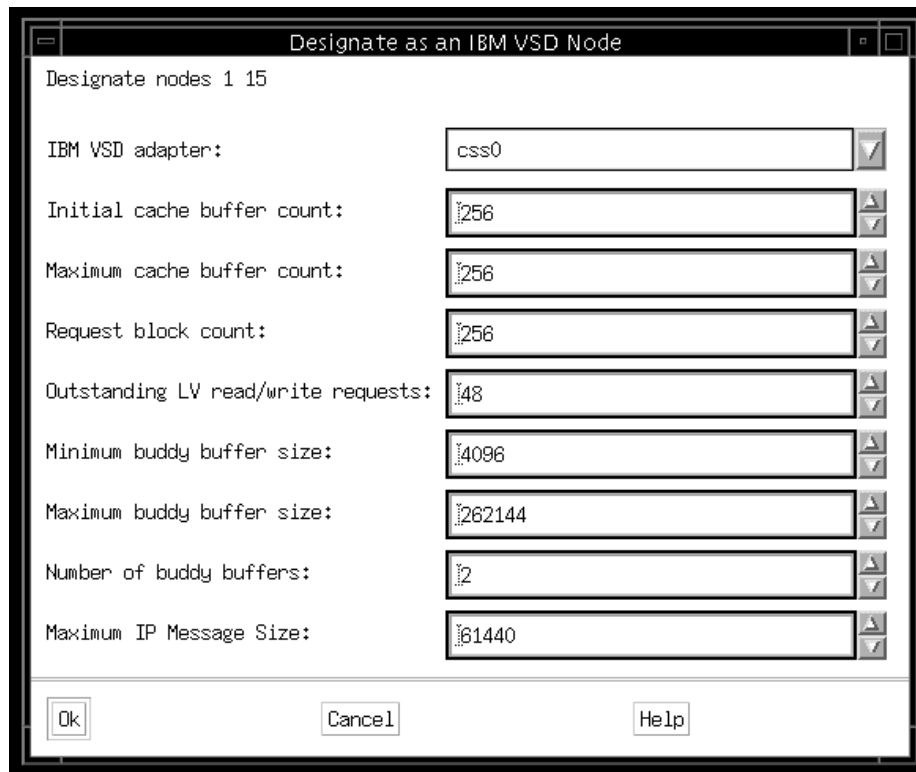


Figure 166. Perspectives Panel for Designating VSD Nodes

3. Create or define the VSDs for each node that is to be a VSD server.

There are two options to set up VSDs on the server nodes: create them or define them.

Creating VSDs means that there are no VGs and LVs established on the server node to be used for VSDs. In this case, one can again go through the Perspectives interface or the command line. In Perspectives, it is necessary to first add a pane that shows the VSDs on the server node (even if none has yet been created). Once the pane is added, use the action **Create** to create both the global VG and the LVs to be used as VSDs. For SMIT, run `smit vsd_data` and then select the **Create a Virtual Shared Disk** option. From the command line, run `createvsd`. Once again, it we recommend that you use the Perspectives interface or the SMIT panels because they automatically bring up all the settings and options available.

Figure 167 is an example of the Perspectives panel used for creating VSDs.

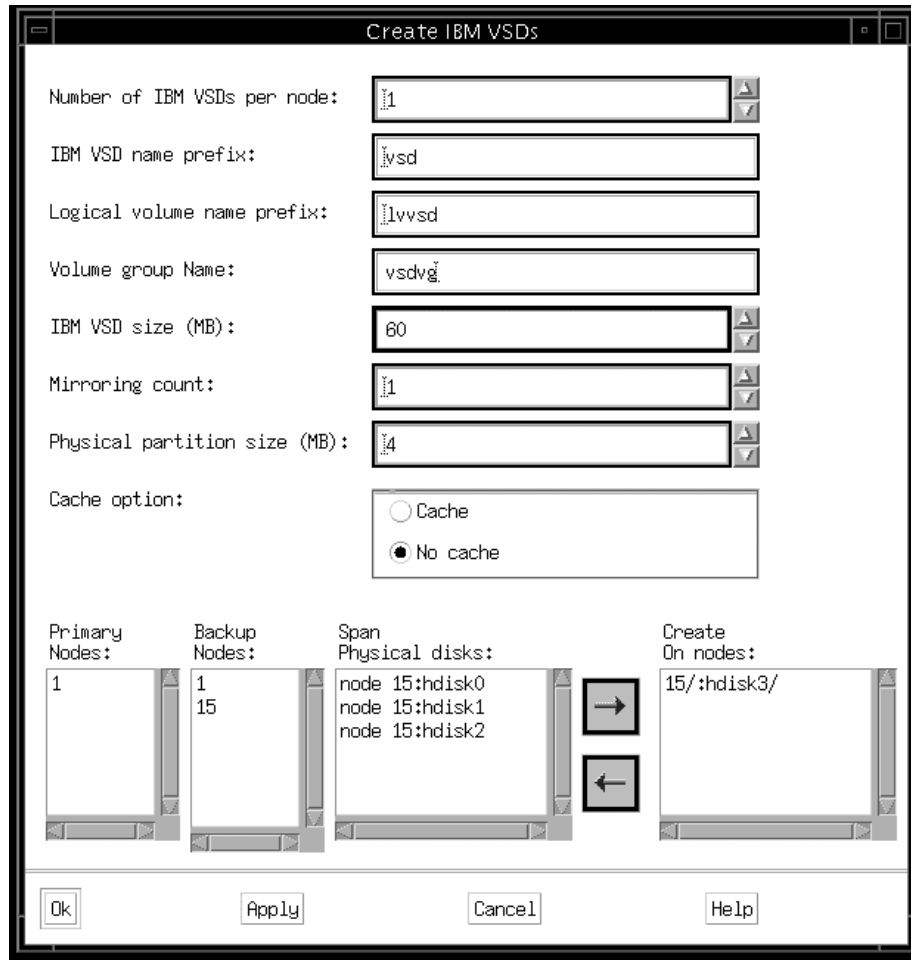


Figure 167. Perspectives Panel for Creating VSDs

Defining VSDs means that VGs and LVs have already been established locally on the node and you want the SP to use them for VSDs, that is, you want to define them in the SDR on the control workstation. In Perspectives, from the VSD pane, use the action **Define** to define a LV as a VSD within a global VG. For SMIT panels, run `smit vsd_data` and then select the **Define a Virtual Shared Disk** option. The command line equivalent is `defvsvd`.

Figure 168 is an example of the Perspectives panel for defining VSDs.

The image shows a window titled "Define an IBM VSD". It contains the following fields and options:

- Logical volume name:
- Global volume group name:
- IBM VSD name:
- Cache option: Cache, No cache

At the bottom of the window are four buttons: Ok, Apply, Cancel, and Help.

Figure 168. Perspectives Panel for Defining VSDs

4. Configure the VSDs.

This step takes the VSDs that have been created and defined on a server and makes them available for use on all the nodes that are going to read from and write to them.

In Perspectives, there are two options to carry out this task. You can select the VSDs you want to configure, then the action **Configure**. Alternatively, you can select the nodes on which you want to configure the VSD, then use the action **Configure IBM VSDs**. For SMIT, run `smit vsd_mgmt` and select the **Configure a VSD** option. The command line equivalent is `cfgvsd`.

Figure 169 on page 401 is an example of the Perspectives panel for configuring VSDs.

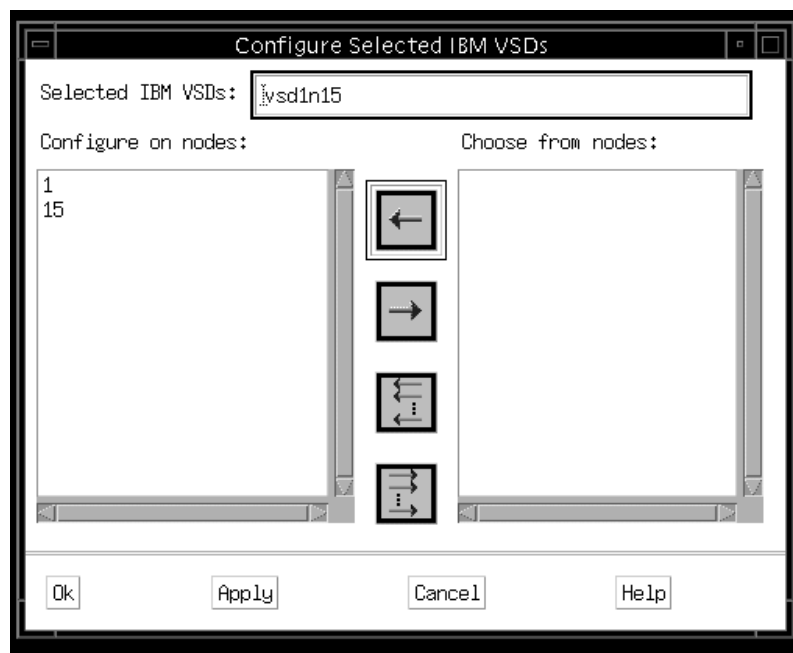


Figure 169. Perspectives Panel for Configuring VSDs

5. Activate the VSDs.

This is the step that puts the defined and configured VSDs into the active, or available state. In order for a VSD to be used, it must be in the active (available) state on both the server and client nodes.

In Perspectives, there are two choices. Select the VSD nodes, then run the action **Change IBM VSDs State**, or select the VSDs and run the action **Change State**. For SMIT, run `smit vsd_mgmt` and select the **Start a VSD** option. For the command line, run `startvsd`.

Figure 170 on page 402 is an example of the Perspectives panel for activating VSDs.

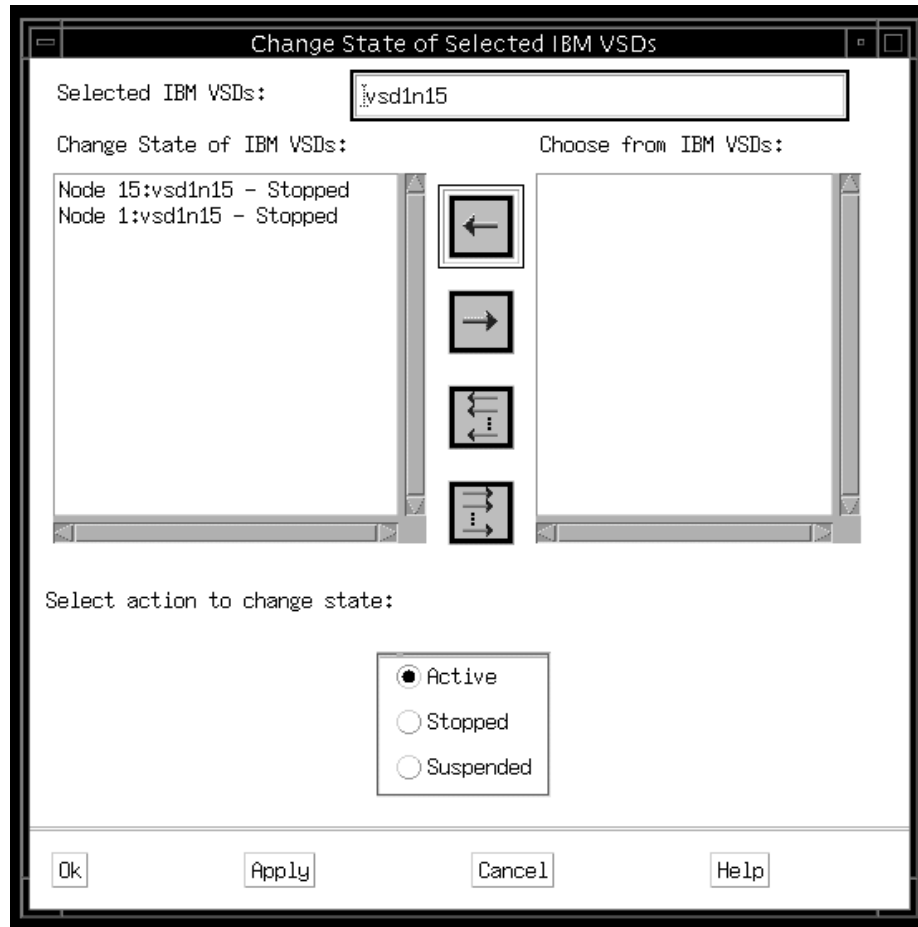


Figure 170. Perspectives Panel for Activating VSDs

Once activated, VSDs have resource variables, which may be monitored to provide operational statistics. Monitoring is set up by administrators in the PSSP Event Perspectives, and it is described in detail in *IBM Parallel System Support Programs for AIX: Managing Shared Disks, SA22-7349*.

There are five different states in which a VSD can be found, depending on the circumstances in the SP system at that time. These circumstances include changes to the configuration (either on a server node or the entire SP system) and problems in the system, application or network. By moving the VSDs into different states, the system is better able to keep track of I/O

operations, for example, and record how far to roll back to properly recover from failures. These states are summarized in Figure 171:

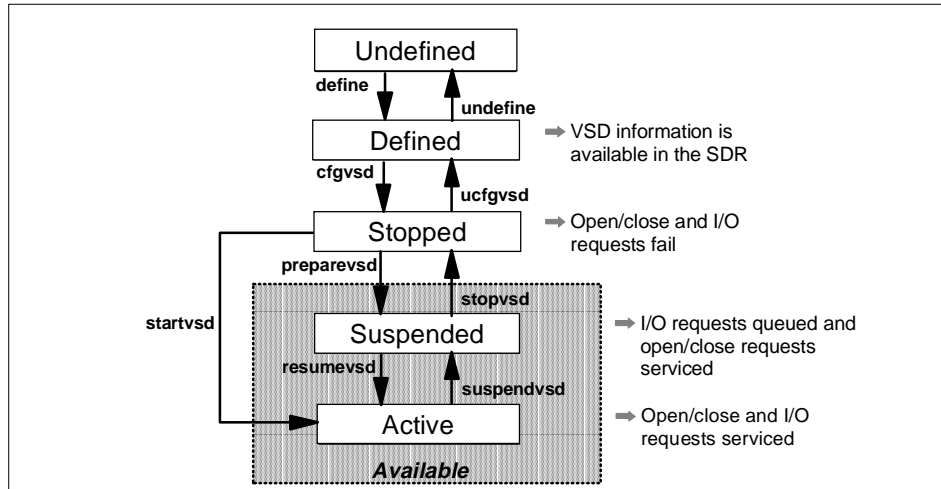


Figure 171. VSD States and Commands

For more information on the commands `vsdnode`, `createvsd`, `defvsd`, `cfgvsd`, `preparevsd`, `stopvsd`, `resumevsd`, `suspendvsd`, and `startvsd`, refer to *IBM Parallel System Support Programs for AIX: Command and Technical Reference*, SA22-7351.

In the past, any changes made to VSDs required the system administrator to first stop the VSDs, unconfigure them from all VSD nodes, make the changes, re-configure the VSDs, and re-start them. This is at best a tedious task. In PSSP 3.1, improvements have been made so that the following functionalities can be dynamically carried out; that is, there is no need to stop and re-start the VSDs:

1. The addition and subtraction of VSD nodes
2. The addition and subtraction of VSDs running on a node
3. Turning on or turning off the cache option
4. Increasing the size of individual VSDs

In addition, a separate filesystem is now used to store all VSD configuration files and logs, pending the availability of disk space. This filesystem is called `/var/adm/csd`. This feature alleviates the space utilization problem in the `/var` file system. If there is insufficient disk space to have a separate filesystem, a directory with the same name is created.

Although VSD provides distributed data access, it does not provide a locking mechanism to preserve data integrity. This task falls upon the application itself.

VSDs can be run over any IP network. However, the SP Switch network is the only communication device available on the RS/6000 SP capable of providing the necessary bandwidth and scalability for VSD to operate at good performance. The SP Switch permits:

- High I/O bandwidth for optimal VSD performance
- Scalable growth for any applications using VSDs

14.3.2 Hashed Shared Disks (HSD)

VSDs running over the SP switch can enable efficient access to raw LVs within an SP system. However, there may be instances where the performance bottleneck lies with the I/O capacity within an SP node itself. To help alleviate this problem, IBM provides Hashed Shared Disk (HSD). HSD is a striped version of VSD. The HSD architecture is shown in Figure 172.

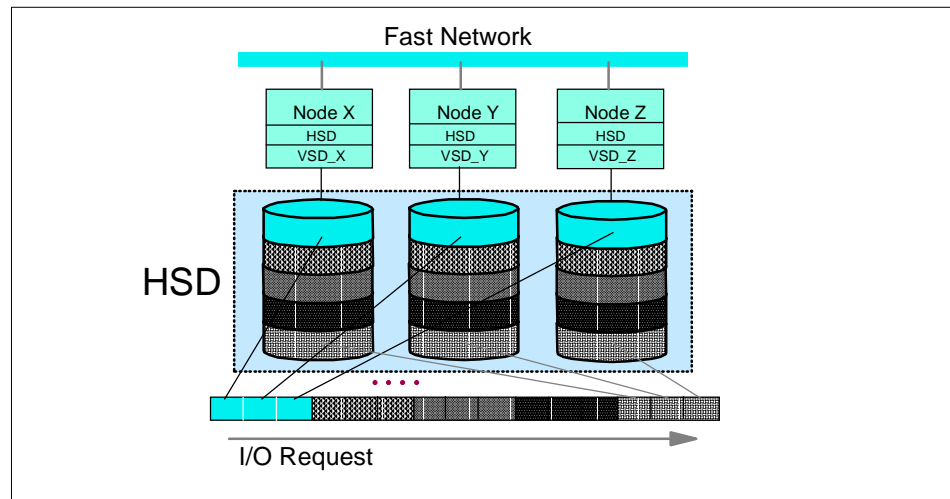


Figure 172. HSD Architecture

HSD adds a device driver above the VSD layer on individual nodes. When an I/O request is made by the application, the HSD device driver breaks it down into smaller blocks, depending upon the strip size specified and the number of VSDs making up the HSD. The strip size defines the amount of data which is read from or written out to one VSD per instruction. The I/O request is then distributed among all the VSDs which make up the HSD for handling.

HSDs are created or defined in a manner similar to the steps outlined in 14.3.1, “Virtual Shared Disks (VSD)” on page 395. Instead of creating, defining, or configuring VSDs, select the same options for HSDs. In this case, the commands are `createhds`, `defhds`, and `cfghsd`. The only significant difference is that an HSD activates automatically upon configuration; that is, there is no need to run step 4 (`startvds`).

HSD helps to spread the data over several VSD and VSD nodes, thereby reducing the chance of I/O performance bottlenecks on individual nodes. It is, however, difficult to manage because any changes, either at the HSD level or at the VSD level, require the deletion and re-creation of the HSD. Other restrictions with HSD are documented in *IBM Parallel System Support Programs For AIX: Managing Shared Disks*, SA22-7349.

14.3.3 Recoverable Virtual Shared Disks (RVSD)

Let us refer again to Figure 165 on page 395. In this figure, there are two VSDs and two VSD nodes. What if Node Y has to be shut down for maintenance? Users and processes on Node X then have no way of accessing the data in `lv_Y`. This problem may be alleviated by twin-tailing the disks holding `lv_X` and `lv_Y`, so that the two VSD nodes are connected to the two VSDs, and using the software Recoverable Virtual Shared Disk (RVSD) to provide a failover mechanism between the two nodes.

RVSD designates nodes which serve VSDs as *primary nodes*. The backup nodes are called *secondary nodes*.

In the example in Figure 165, with RVSD installed and configured, if Node Y has to be shut down for maintenance, then `lv_Y` is going to failover and be controlled by Node X. While Node Y is down, Node X becomes a server to both VSDs `lv_X` and `lv_Y`. When Node Y is back up and operational, RVSD returns back to it the control of `lv_Y`. This failover is transparent to the users and processes in Node X.

There are different versions of the RVSD software for different versions of PSSP. Each version of RVSD can interoperate with each of the supported levels of PSSP as shown in Table 20.

Table 20. RVSD Levels Supported by PSSP Levels

	PSSP 2.2	PSSP 2.3	PSSP 2.4	PSSP 3.1
RVSD 1.2	Y	Y	Y	Y
RVSD 2.1	N	Y	Y	Y
RVSD 2.1.1	N	N	Y	Y

	PSSP 2.2	PSSP 2.3	PSSP 2.4	PSSP 3.1
RVSD 3.1	N	N	N	Y

RVSD is set to operate with the functionality of the lowest PSSP level installed in the SP system, regardless of whether this node is running RVSD or not. To overcome this problem, PSSP 3.1 introduces the command `rvsdrestrict`, which allows system administrators to set the functionality level of RVSD. A detailed description of `rvsdrestrict` in *PSSP 3.1 Announcement*, SG24-5332.

RVSD contains two subsystems: `rvsd` and `hc`. Subsystem `rvsd` controls the recovery for RVSD, while the `hc` subsystem supports the development of recoverable applications.

The `rvsd` subsystem records the details associated with each of the VSD servers: the network adapter status, the number of nodes active, the number of nodes required for quorum, and so on.

Quorum in RVSD is defined as the minimum number of active nodes required to continue serving the VSDs. The default setting is half the number of VSD nodes plus one (a majority). If too many nodes become inactive (for example, shutdown), the quorum is lost and the VSDs are stopped. It is possible to change this setting via the `ha.vsd` command.

The `rvsd` subsystem handles three types of failures: node, disk cable and adapters, and communication adapters. Node failures are described later in this section. Disk cable and adapter failures are also handled as node failures, but only for those VGs which are affected. For example, if there are two global VGs served by one primary node, and one disk within one of the VG fails, only that VG is going to be failed over to the secondary node. The primary node will continue to serve the other VG. When the disk has been replaced, it is necessary to manually run the `vsdchgserver` command to change the control back to the primary node. Note that `rvsd` only handles communication adapter failures for the Ethernet and SP switch networks. It is possible to run RVSD using other types of network adapters, but there is no failover mechanism in those instances. Communication adapter failures are handled in the same manner as node failures due to RVSD's dependence upon an IP network to function.

The `hc` subsystem, also called the Connection Manager, shadows the `rvsd` subsystem, recording the same changes in state and management of VSDs that `rvsd` records. There is, however, one difference: `hc` records these changes *after* `rvsd` has processed them. This ensures that RVSD recovery

events have a chance to begin and complete before hc client application events are carried out. In turn, this serialization helps to preserve data integrity.

Another way to preserve data integrity during application recovery is the concept of fencing, available only with RVSD.

If a recoverable database application is running in a system and one of the running nodes fails, any data locked by the failed application instance is returned to a consistent state. A failed node, however, may still issue I/O requests to the VSDs. It is important that such requests do not get executed, because data may get corrupted. An application's recovery script may use the command `fencevsd` to prevent I/O operations from failed nodes to VSDs in the system. The syntax for this command is `fencedvsd -v [name of vsd] -n [node number or node list]`. For example, to fence node 13 from a vsd named `vsd1n15`, run `fencevsd -v vsd1n15 -n 13`.

When the failed node is active again, the application's recovery script can issue `unfencevsd` to permit it to issue VSD I/Os. The syntax is similar to that for `fencevsd`.

The command `lsfencevsd` may be run to display a map of all fenced nodes and the VSDs from which they are fenced.

Further information about `ha.vsd`, `fencevsd`, `unfencevsd`, `lsfencevsd` and the `rvsd` and `hc` subsystems can be found in *IBM Parallel System Support Programs for AIX: Managing Shared Disks, SA22-7349*.

For problem determination purposes, RVSD keeps a set of logs under `/var/adm/csd`, which may be gathered with the `vsd.snap` command. Two particular files to focus upon for debugging are the `vsd.log` and `vsd.debuglog` files. The `vsd.log` keeps track of the events that are taking place during a failover, while `vsd.debuglog` provides additional details like group membership.

Using `rvsd`, RVSD handles a node failover as a two-event process: `vsd.down1` and `vsd.down2`.

Consider a scenario when a primary node goes down. The first event logged is `vsd.down1`. At `vsd.down1`, the primary node runs `suspendvsd` and `stopvsd` to bring its VSDs into the stopped state. On all non-primary nodes (both secondary servers and clients), `suspendvsd` is run to bring the VSDs into the suspend state. This ensures no further I/O activities to the VSDs, and thereby maintains data integrity. Refer to Figure 171 on page 403 for a pictorial review of VSD states.

At `vsd.down2`, `varyoffvg` is run on the primary node against the VG to which the VSDs belong. After this, the secondary node runs `exportvg`, `importvg` and `varyonvg` against the same VG to break the primary node's reservation, followed by `preparevds` and `resumevds` to bring the VSDs back to the active state. The VSDs are now served by the secondary node. Client nodes then run `resumevds` to bring the VSDs to the active state so they can access them. Finally, `rvsd` decreases its count of up nodes by 1.

Figure 173 summarizes the RVSD node failover events.

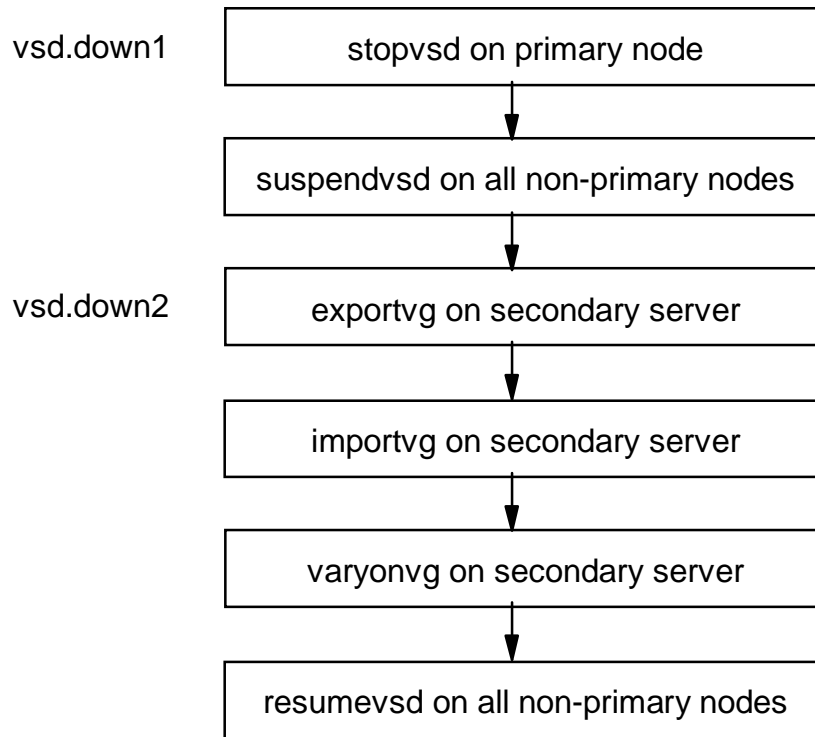


Figure 173. Summary of RVSD Failover Events

When the primary node is back up and running, there is again a two-step process to return control of its VSDs.

Event `vsd.up1` has the clients running `suspendvds` and the secondary node running both `suspendvds` and `stopvds` against the VSDs to stop all I/O activities.

Event vsd.up2 then has the secondary node running `varyoffvg` against the VG to which the VSDs belong, followed by the primary node running `exportvg`, `importvg` and `varyonvg` against the same VG to reestablish reservation. Then `preparevsd` and `resumevsd` on the primary node and `resumevsd` on all other nodes are run to return the VSDs to the active state for normal operations. Finally, `rvsd` increases its count of up nodes by 1.

Figure 174 summarizes the RVSD node recovery events.

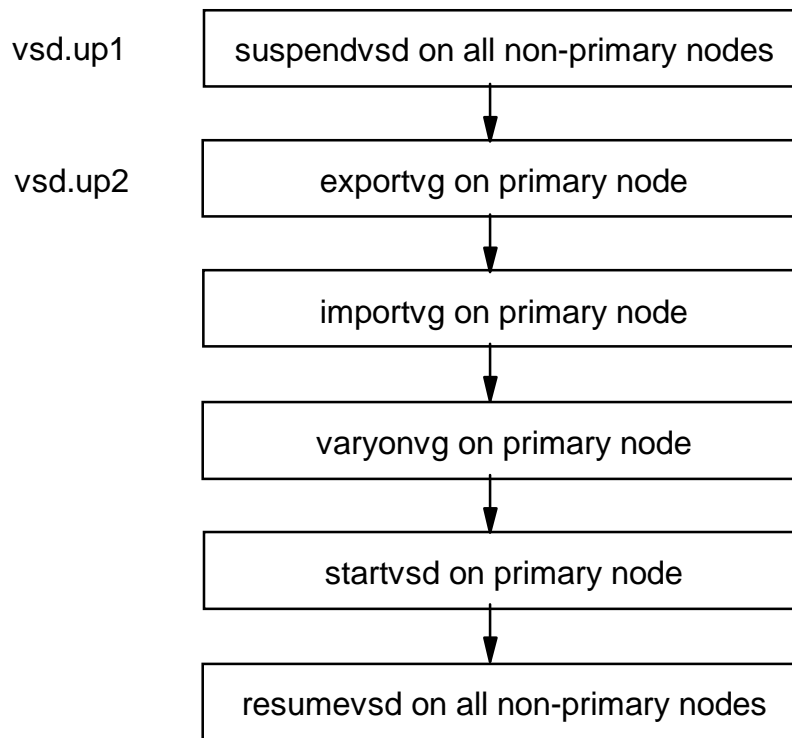


Figure 174. Summary of RVSD Recovery Events

14.4 General Parallel File System (GPFS)

Another method for sharing data globally within an SP system is the IBM-developed General Parallel File System (GPFS). GPFS provides a file system which is accessible by multiple users and by processes running on multiple nodes.

GPFS exploits the VSD technology to build a file system over multiple hard disks on multiple nodes. This file system is then mountable for use by a number of designated nodes. Using a token management system, GPFS enables concurrent read and write access to files. Therefore, both serial and parallel applications can use GPFS.

Although VSDs can be implemented over any IP network, GPFS is only supported over the SP Switch. This is because GPFS is designed for high performance, and the SP Switch offers the highest bandwidth and speed possible within an SP system. Part of GPFS's high performance design involves the striping of data across multiple disks on multiple nodes. This provides multiple servers for one file system, spreading and balancing I/O requests.

Because GPFS spreads a file system across many nodes and disks, it is possible to create very large file systems to hold very large files.

Additional features of GPFS include the capability to add or delete disks, even while the file system is mounted; a dynamic restripe of the file system to improve data block allocations; the use of a log on each GPFS node to track transactions. This log is then replicated on another GPFS node to enhance recovery from errors.

GPFS configurations are stored in the SDR on the control workstation. When a GPFS node boots, it checks the SDR to see if there have been changes. If so, these are applied during the boot. This permits changes to be made to GPFS even if not all the GPFS nodes are up and running.

Figure 175 on page 411 shows the software architecture of GPFS.

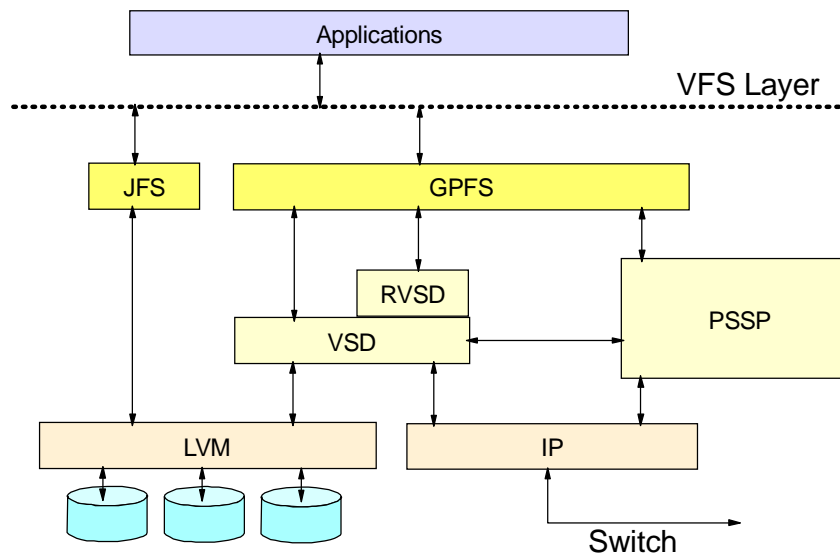


Figure 175. GPFS Software Architecture

The GPFS device driver is a kernel extension which sits between the application and r/vsd layers. It uses standard AIX Virtual File System (VFS) system calls, making it possible for most AIX applications to run over GPFS without modifications. The application views a GPFS file system similar to an AIX Journal File System (JFS). In operations, when GPFS fetches data, it uses the r/vsd device driver to go to the VSD node which has the data. This design makes it possible to separate nodes involved with GPFS into GPFS nodes and VSD server nodes. GPFS nodes are those which mount and use a GPFS while disk server nodes are those which have the underlying VSDs configured on them. The GPFS device driver does not need to know where the data is stored; this is left to the r/vsd device driver. Using the r/vsd device driver makes it possible to separate GPFS nodes into two types: the GPFS nodes, which are those that mount and use GPFS; and the VSD server nodes, which have the underlying hard disks configured as VSDs. The advantage with this is that the VSD servers, unless an application requires it, are not further burdened with running both VSD and GPFS software.

Underneath these layers, GPFS uses the traditional UNIX file system structure of i-nodes, indirect blocks and data blocks to store files. A detailed discussion of these components and their impact on a GPFS file system can be found in *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278.

Nodes that run GPFS are grouped together to form a GPFS domain, which is managed by the mmfsd daemon, described in 14.4.1, “The mmfsd Daemon” on page 412.

A restriction with current version of GPFS is that it does not handle memory-mapped files. It is expected to be resolved in a future version.

The GPFS filesets (LPPs) consists of 6 different filesets, all beginning with mmfs. This prefix refers to GPFS’s development alongside IBM’s Multi-Media LAN Server product. These filesets are:

- mmfs.base.usr.3.1.0.0
- mmfs.gpfs.usr.1.2.0.0
- mmfs.util.usr.3.1.0.0
- mmfs.msg.en_US.usr.3.1.0.0
- mmfs.man.en_US.shr.3.1.0.0
- mmfs.gpfsdocs.shr.3.1.0.0

RVSD software is a prerequisite to GPFS. Although GPFS can operate without the high availability of disks provided by RVSD, there are commands within RVSD that GPFS needs, such as `fencevsd` and `unfencevsd`. These commands are described in 14.3.3, “Recoverable Virtual Shared Disks (RVSD)” on page 405.

The procedure to install the GPFS filesets and the prerequisites are described in *IBM General Parallel File System for AIX: Installation and Administration Guide, SA22-7278*.

14.4.1 The mmfsd Daemon

At the heart of the GPFS software lies the GPFS daemon (mmfsd). This daemon runs on every GPFS node to coordinate, synchronize and manage GPFS-related duties. Such duties include: data and metadata management (disk space allocation, data access, disk I/O operations and so on), security and quota management. To carry out all of these duties, an mmfsd daemon can take on four different roles or personalities, as follows:

Configuration Manager

In this role, the mmfsd daemon is responsible for configuration tasks within a GPFS domain, such as selecting the Stripe Group Manager for each file system and determining whether quorum exists.

The concept of quorum in GPFS is similar to that found in LVM. At least half the number of nodes, plus one, in the GPFS domain must be up and running. If quorum is lost, the Configuration Manager unmounts the file system, suspending further operations to maintain data consistency and integrity.

There is one GPFS Configuration Manager per system partition. It is the first node to join the group Mmfs Group in Group Services (for more information on Group Services, refer to 8.5, “Group Services (GS)” on page 193). If this node goes down, Group Services selects the next oldest node in the Mmfs Group to be the Configuration Manager.

Stripe Group Manager

A stripe group consists of the set of hard disks which makes up a GPFS file system. There is one Stripe Group Manager per GPFS file system to perform the following services:

- Process changes to the state or description of the file system
 1. Add/delete/replace disks
 2. Change disk availability
 3. Repair file system
 4. Restripe file system
- Control disk region allocation

In addition, the Stripe Group Manager handles the token requests for file access, passing each request to the Token Manager Server for processing.

The Configuration Manager is responsible for selecting the Stripe Group Manager. It avoids overloading any one node in a system partition by selecting a different node for each GPFS file system to act as the Stripe Group Manager.

This selection may be influenced by the GPFS administrator. Create the file `/var/mmfs/etc/cluster.preference` and list in it the switch hostnames of the nodes, one per line, that you want to be a Stripe Group Manager. The Configuration Manager looks for this file, and if found, selects a node from this list. There is no order of priority given to the nodes in `cluster.preference`; the only requirement is that the node listed in the file be up and available when the selection is made.

If the node that is the Stripe Group Manager goes down, another node is selected to take over. The selection comes from another node in the `cluster.preference` file, if it is used, or simply another node in the system

partition. In either instance, priority is given to those nodes that are not serving as a Stripe Group Manager.

Metadata Manager

A Metadata Manager maintains the integrity of the metadata for open files within a GPFS file system. There is one Metadata Manager for each open file. The Metadata Manager is the only node that can update the metadata pertaining to the open file, even if the data in the file is updated by another node.

The Metadata Manager is selected to be the first node that opened the file. This node stays as the Metadata Manager for the file until one of following events occur:

- The file is closed everywhere.
- The node fails.
- The node resigns from the GPFS domain.

When the Metadata Manager goes down, the next node to use the metadata service takes over as the Metadata Manager.

Token Manager Server

The use of tokens helps GPFS manage and coordinate file access among the GPFS nodes. In turn, this preserves data integrity. The concept of tokens in GPFS is similar to that of file locks in LVM. There is one Token Manager Server per GPFS file system. It resides in the same node as the Stripe Group Manager, and processes all requests for access to files within the GPFS.

There is a token manager component which runs on every GPFS node. When that node wants to access a file, its token manager requests a token from the Token Manager Server. The Token Manager Server determines if any locking conflicts exist among any previously granted tokens and the current request.

If no conflicts exist, a token is granted.

If conflicts exist, the Token Manager Server sends a list called a *copy set* back to the requesting node's token manager. It is then the token manager's responsibility to process the copy set and negotiate with the token manager(s) on the node(s) in the copy set to obtain a token and access the file.

This type of negotiation among nodes helps to reduce the overhead on the Token Manager Server.

For availability purposes, there are two copies of every token: one at the Token Manager Server and one at the token manager.

If a node goes down, it can recover by having its token manager request a copy of the token from the Token Manager Server.

If the Token Manager Server node goes down, the new Token Manager Server (which is also the node that is the new Stripe Group Manager) then obtains the client copy of all tokens from all the token managers in the GPFS domain.

14.4.2 GPFS Setup

Chapter 2 of *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278 is devoted to a series of steps for planning a GPFS implementation. It is recommended that this section be read and understood prior to installing and using GPFS.

GPFS tasks cannot be done on the control workstation; they must be performed on one of the GPFS nodes.

There are three areas of consideration when GPFS is being set up: the nodes using GPFS, the VSDs to be used, and the file systems to be created. These areas are discussed in detail in the remainder of this section, with reference to a sample file system setup consisting of four nodes. Nodes 12, 13 and 14 are GPFS nodes, while node 15 is the VSD server node.

Important

Do not attempt to start the mmfsd daemon prior to configuring GPFS. Starting the mmfsd daemon without configuring GPFS causes dummy kernel extensions to be loaded and you will be unable to create a file system. If this occurs, configure GPFS and then reboot the node(s).

Carry out the following procedures to configure GPFS, then start the mmfsd daemon to continue creating the file system.

Nodes

The first step in setting up GPFS is to define which nodes are GPFS nodes. The second step is to specify the parameters for each node.

There are three areas where nodes can be specified for GPFS operations: node count, node list, and nodes preferences.

The *Node Count* is an estimate of the maximum number of nodes that will mount the file system, and it is entered into the system only when the GPFS file system is created. It is recommended that you overestimate this number. This number is used in the creation of GPFS data structures that are essential for achieving the maximum degree of parallelism in file system operations. Although a larger estimate consumes a bit more memory, insufficient allocation of GPFS data structures can limit a node's ability to process certain parallel requests efficiently, such as the allotment of disk space to a file. If it is not possible to estimate the number of nodes, apply the default value of 32. A larger number may be specified if more nodes are expected to be added.

However, it is important to avoid wildly overestimating, since this can affect buffer operations. This value cannot be changed later. If you need to change this value, the file system must be destroyed and recreated using the new value.

A *node list* is a file which specifies to GPFS the actual nodes to be included in the GPFS domain. This file may have any filename. However, when GPFS configures the nodes, it copies the file to each GPFS node as `/etc/cluster.nodes`. The GPFS nodes are listed one per line in this file, and the switch interface is specified because this is the interface over which GPFS runs.

Figure 176 on page 417 is an example of a node list file. The filename in this example is `/var/mmfs/etc/nodes.list`.



Figure 177. SMIT Panel for Configuring GPFS

It is possible to configure GPFS to automatically start on all nodes whenever they come up. Simply specify yes to the autoload option in the SMIT panel or the -A flag in the `mmconfig` command. This eliminates the need to manually start GPFS when nodes are rebooted.

The pagepool and malloysize options specify the size of the cache on each node dedicated for GPFS operations. malloysize sets an area dedicated for holding GPFS control structures data while pagepool is the actual size of the cache on each node. In this instance, pagepool is specified to the default size of 4M while malloysize is specified to be the default of 2M, where M stands for megabytes and must be included in the field. The maximum values per node are 512 MB for pagepool and 128 MB for malloysize.

The priority field refers to the scheduling priority for the mmfsd daemon. The concept of priority is beyond the scope of this book: refer to AIX documentation for more information.

Notice the file `/usr/lpp/mmfs/samples/mmfs.cfg.sample`. This file contains the default values used to configure GPFS if none are specified, either through the fields in the SMIT panel or in another file. The use of another file to set GPFS options may appeal to more experienced users, or to those who want to configure multiple GPFS domains with the same parameters. Simply copy this file (`/usr/lpp/mmfs/samples/mmfs.cfg.sample`) to a different file, make the changes according to your specifications, propagate it out to the nodes, and configure using SMIT or the `mmconfig` command.

Further information, including details regarding the values to set for `pagepool` and `mallocsize`, is in *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278.

Once GPFS has been configured, the `mmfsd` daemon has to be started on the GPFS nodes before a file system can be created. The steps to do this are as follows:

1. Set the `WCOLL` environment variable to target all GPFS nodes for the `dsh` command. *IBM Parallel Systems Support Programs: Administration Guide*, SA22-7348, *IBM Parallel Systems Support Programs: Command and Technical Reference*, SA22-7351, and *IBM RS/6000 SP Management, Easy, Lean, and Mean*, GG24-2563, all contain information on the `WCOLL` environment variable.
2. Designate each of the nodes in the GPFS domain as an IBM VSD node.
3. Ensure that the `rvsd` and `hc` daemons are active on the GPFS nodes.
Note: `rvsd` and `hc` do not start unless they detect the presence of one VSD defined for the GPFS nodes. This VSD may or may not be used in the GPFS file system.
4. Start the `mmfsd` daemon by running the following command on one GPFS node:

```
dsh startsrc -s mmfs
```

The `mmfsd` starts on all the nodes specified in the `/etc/cluster.nodes` file. If the startup is successful, the file `/var/adm/ras/mmfs.log*` looks like Figure 178 on page 420.

The image shows a terminal window titled 'aixterm'. The window contains the following text:

```
MMFS: 6027-506 /usr/lpp/mmfs/bin/mmfskxload: /usr/lib/drivers/mmfs is already lo
aded at 83258984.
Thu Feb 18 18:32:01: MMFS: 6027-310 mmfsd initializing ...
Thu Feb 18 18:32:01: MMFS: 6027-300 mmfsd ready for sessions.
# _
```

Figure 178. Sample Output of /var/adm/ras/mmfs.log*

VSDs

Before the file system can be created, the underlying VSDs must be set up. The nodes with the VSDs configured may be strictly VSD server nodes, or they can also be GPFS nodes. Consider the application to decide whether to include VSD server-only nodes in the GPFS domain.

You must also decide the level of redundancy needed to guard against failures. Should the VSDs be mirrored? Should they run with a RAID subsystem on top? Should RVSD be used in case of node failures? Again, this depends on the application, but it also depends on your comfort and preferences for dealing with risk.

In addition to these options, GPFS provides two further recovery strategies at the VSD disk level. GPFS organizes disks into a number of failure groups. A failure group is simply a set of disks that share a common point of failure. A common point of failure is defined as that which, if it goes down, causes the set of disks to become simultaneously unavailable. For example, if a VSD

spans two physical disks within one node, the two disks can be considered a failure group because if the node goes down, both disks become unavailable.

Recall that there are two types of data that GPFS handles: metadata and the data itself. GPFS can decide what is stored on each VSD: metadata only, data only, or data and metadata. It is possible to separate metadata and data, to ensure that data corruption does not affect the metadata, and vice versa.

Further, the separation of data and metadata can even enhance performance. This is best seen if RAID is involved. RAID devices are not suited for handling metadata because metadata is small in size and can be handled using small I/O block sizes. RAID is most effective at handling large I/O block sizes. metadata can therefore be stored in a non-RAID environment, such as mirrored disks, while the data can be stored in a RAID disk. This both protects data and metadata from each other, and maximizes the performance given that RAID is chosen.

Once you adopt the redundancy strategy, there are two ways to create VSDs: have GPFS do it for you, or manually create them. For either method, this is done through the use of a Disk Descriptor file. This file can be set up manually or through the use of SMIT panels. If using SMIT, run `smit gpfs` and then select the **Prepare Disk Descriptor File** option. Figure 179 on page 422 shows the SMIT panel for our example.

In this case, the VSD `vsd1n15` has already been created on node 15 (`sp3n15`). *Do not* specify a name for the server node because the system already has all the information it needs from the configuration files in the SDR. In addition, the VSD(s) must be in the Active state on the VSD server node and on all the GPFS nodes prior to the file system creation.

If the VSDs have not been created, specify the name of the disk (such as `hdisk3`) in the disk name field, instead of `vsd1n15`, and specify the server where this `hdisk` is connected. GPFS then creates the necessary VSDs to create the file system.

The failure group number may be system generated or user specified. In this case, a number of 1 is specified. If no number is specified, the system provides a default number that is equal to the VSD server node number plus 4000.

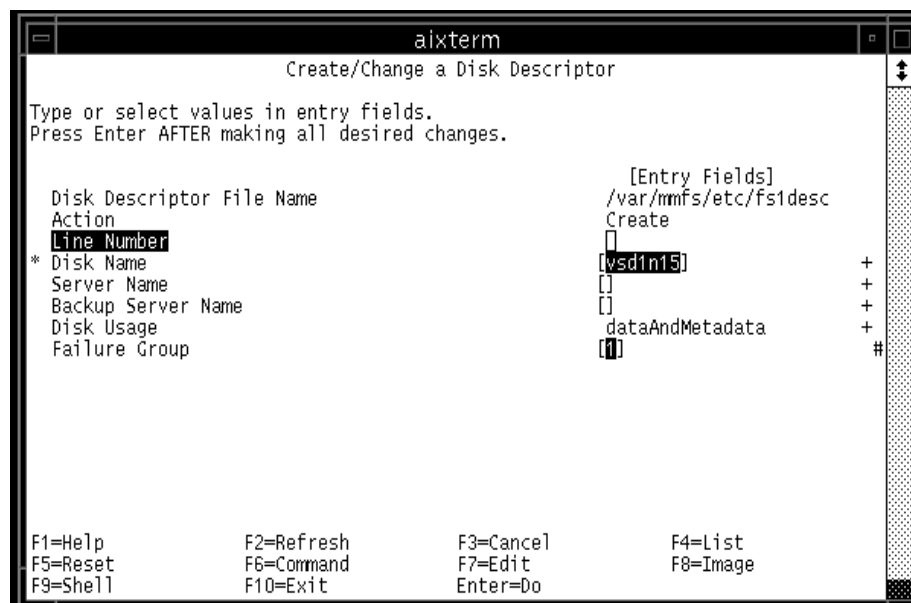


Figure 179. SMIT Panel for Creating Disk Descriptor File

File System

There are two ways to create a GPFS file system: using SMIT panels or the `mmcrfs` command. Figure 180 on page 423 shows the SMIT panel. This is accessed by running `smit gpfs` and then selecting the **Create File System** option. Details on `mmcrfs` can be found in *IBM General Parallel File System for AIX: Installation and Administration Guide, SA22-7278*.

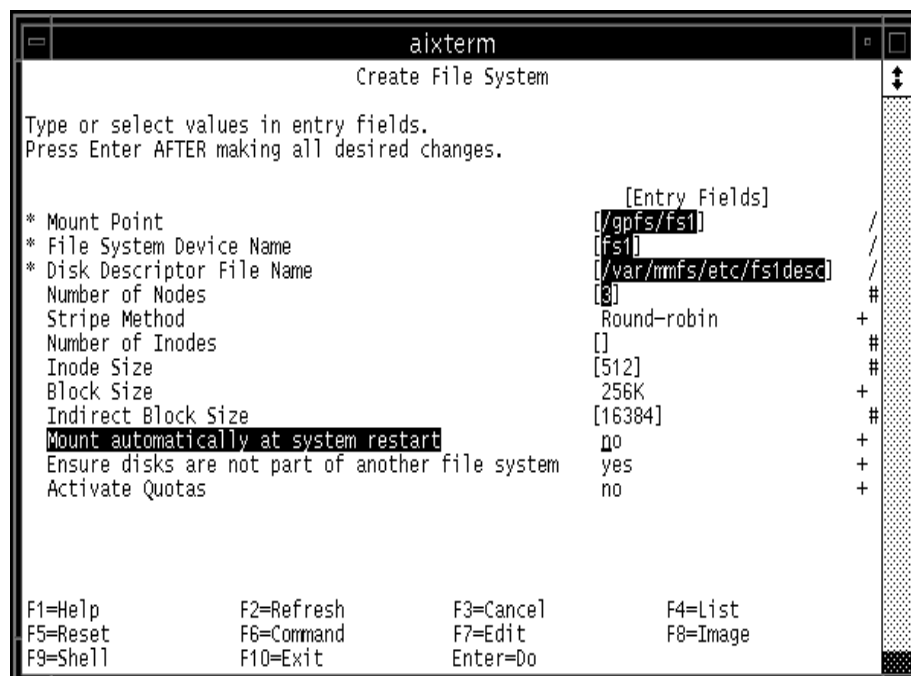


Figure 180. SMIT Panel for Creating a GPFS File System

Four issues must be considered before the file system is created, as follows:

1. How to structure the data in the file system

There are three factors to consider in structuring the data in the file system: block size, i-node size, and indirect block size.

GPFS offers a choice of three block sizes for I/O to and from the file system: 16 KB, 64 KB, or 256 KB. Consider the applications running on your system to determine which block size to use. If the applications handle large amounts of data in a single read/write operation, then a large block size may be best. If the size of the files handled by the applications is small, a smaller block size may be more suitable. The default is 256 KB.

GPFS further divides each block of I/O into 32 sub blocks. If the block size is the largest amount of data that can be accessed in a single I/O operation, the sub block is the smallest unit of disk space that can be allocated to a file. For a block size of 256 KB, GPFS reads as much as 256 KB of data in a single I/O operation, and a small file can occupy as little as 8 KB of disk space.

Files smaller than one block size are stored in fragments, which are made up of one or more sub-blocks. Large files are therefore often stored in a number of full blocks, plus one or more fragments to hold the data at the end of the file.

The i-node is also known as the file index. It is the internal structure that describes an individual file to AIX, holding such information as file size and the time of the last modification to the file. In addition, an i-node points to the location of the file on the hard disk. If the file is small, the i-node stores the addresses of all the disk blocks containing the file data. If the file is large, i-nodes point to indirect blocks which point to the disk blocks storing the file data (indirect blocks are set aside to specifically hold only data block addresses).

The default size of an i-node is 512 bytes. This number can increase to 4 KB, depending on the size of the files the application uses.

An indirect block can be as small as a single sub-block or as large as a full block (up to an absolute maximum of 32 KB). The only additional requirement is that the value of an indirect block is a multiple of the size of a sub-block.

It is also possible to specify the number of i-nodes, which limits the maximum number of files that can be created in the file system. In older versions of GPFS, the maximum number of i-nodes is set at GPFS file system creation time and cannot be changed after. With GPFS 1.2, it is now possible to set a limit at file system creation time, and if it proves necessary, change this upper limit. The upper limit is changed by the `mmchfs` command; the exact syntax can be found in *IBM General Parallel File System for AIX: Installation and Administration Guide, SA22-7278*.

2. Striping method

GPFS automatically stripes data across VSDs to increase performance and balance disk I/O. There are three possible striping algorithms GPFS can implement: round Robin, balanced Random, and Random. A striping algorithm may be set when a GPFS file system is first created, or can be modified as a file system parameter later on.

The three algorithms are defined as follows:

- round Robin

This is the default option the system specifies. Data blocks are written to one VSD at a time until all the VSDs in the file system have received a data block. The next round of writes will then write another block to each VSD *in exactly the same order*.

This method yields the best write performance. There is, however, a penalty when a disk is added or removed from the file system. When a disk is added or removed from the file system, a restriping occurs. The round Robin method takes the longest amount of time among the three algorithms to handle this restriping.

- balanced Random

This method is similar to round Robin. When data blocks are written, one block is written to each VSD. When all the VSDs have received one block of data, the round begins. However, in balanced Random, the order in the second round is not the same as the first round. Subsequent rounds are similarly written to all VSDs, but in an order different from that of the previous round.

- Random

As its name implies, there is no set algorithm for handling writes. Each data block is written to a VSD according to a random function. If data replication is required, GPFS does ensure that both copies of the data are not written to the same disk.

3. Whether or not to use GPFS quotas

GPFS quotas define the amount of space in the file system that a user or a group of users is allowed to use. There are three parameters with which quotas operate: hard limit, soft limit, and grace period.

The hard limit is the maximum disk space and files that a user or group can accumulate. Soft limits are the levels below which a user or group can safely operate. A grace period is only used for soft limits; it defines a period of time in which a user or group can exceed the soft limit.

The usage and limits data are stored in the `quota.user` and `quota.group` files that reside in the root directories of GPFS file systems.

In a quota-enabled configuration, one node is automatically nominated as the quota manager whenever GPFS is started. The quota manager allocates disk blocks to the other nodes writing to the file system, and compares the allocated space to the quota limits at regular intervals. In order to reduce the need for frequent space requests from nodes writing to the file system, the quota manager allocates more disk blocks than requested.

Quotas can be turned on by switching the Activate Quotas entry (shown Figure 180 on page 423) to Yes, or by specifying the `-Q yes` flag for the `mmcrfs` command.

Quotas are further discussed in *IBM General Parallel File System for AIX: Installation and Administration Guide, SA22-7278*.

4. Whether or not to replicate the files

At the file system level, GPFS provides an option to have additional copies of data and metadata stored on the VSDs. This is above and beyond disk mirroring. Therefore, with both replication and mirroring turned on, it is possible to have a maximum of four copies of data being written.

It is possible to replicate metadata, data, or both. The parameters for this are Max Meta Data Replicas and Max Data Replicas, which control the maximum factors of replication of metadata and data respectively, and Default Meta Data Replica and Default Data Replicas, the actual factors of replication. Acceptable values are one or two. One is the default and means no replication (only one copy) and two means replication is turned on (two copies). The Default values must be less than or equal to the Max values. In other words, the Max values grant permission for replication, while the Default values turn the replication on or off.

Replication can be set at file system creation time and *cannot* be set via SMIT panels. The only way to turn on replication is with the command `mncrfs` and the flags `-M` for Max Metadata Replicas, `-m` for Default Metadata Replicas, `-R` for Max Data Replicas and `-r` for Default Data Replicas. Using the same example in Figure 180 on page 423, we can create a file system with both metadata and data replication turned on:

```
mncrfs /gpfs/fs1 fs1 -F /var/mmfs/etc/fs1desc -A yes -B 256K -i 512 -I 16K -M 2 -m 2 -n 3 -R 2 -r 2 -v yes
```

More information on these flags and the `mncrfs` command can be found in *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278.

Once a GPFS file system has been set up, it can be mounted or unmounted on the GPFS nodes using the AIX `mount` and `umount` commands. Or, you can use the SMIT panel by running `smit fs` and then selecting **Mount File System**. Figure 181 on page 427 shows the SMIT panel for mounting a file system.

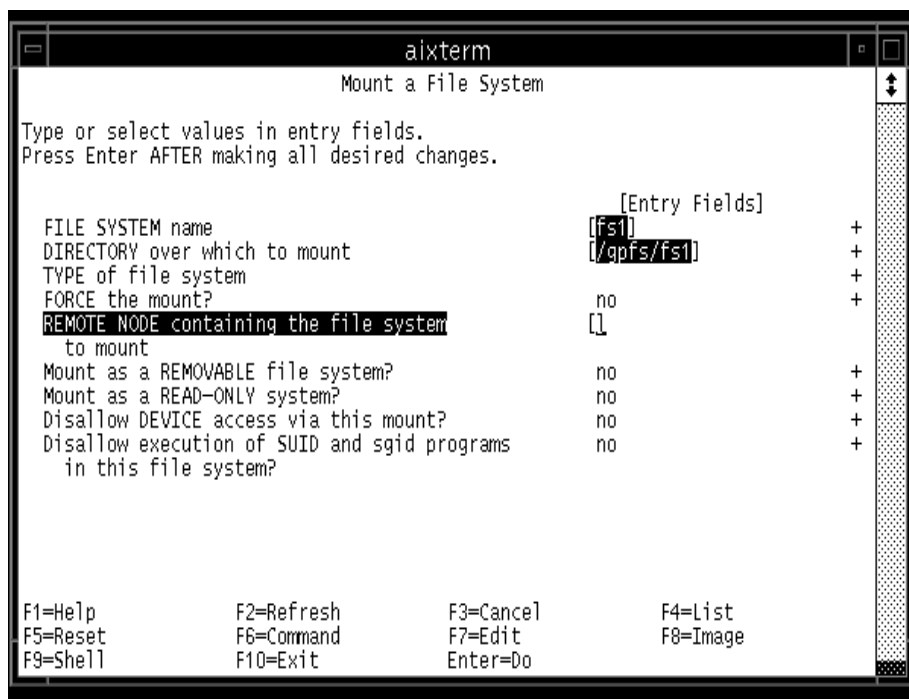


Figure 181. SMIT Panel for Mounting a File System

14.4.3 Managing GPFS

Once a GPFS file system has been set up, a number of tasks can be performed to manage it. Some of the tasks and the commands to execute them are discussed here. Note that SMIT panels are also available to execute the commands. The commands and the SMIT panels are further described in the manual *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278.

Changing the GPFS Configuration

It is possible to change the configuration of GPFS for performance tuning purposes. The command `mmchconfig` and it is capable of changing the following attributes:

- pagepool
- data Structure Dump
- malloysize

- maxFiles To Cache
- priority
- autoload

Changes to pagepool may take effect immediately if the `-i` option is chosen, otherwise the changes will take effect the next time GPFS is started. Changes to data Structure Dump, mallocsize, maxFiles To Cache, and priority require restart of GPFS. Changes to autoload require a reboot of the nodes where this is affected.

For example, to immediately change the size of pagepool to 60 MB, run

```
mmchconfig pagepool=60M -i
```

It is also possible to add and delete nodes from a GPFS configuration. The commands to do so are `mmaddnode` and `mmdelnode`. However, use caution when adding or subtracting nodes from a GPFS configuration, because GPFS uses quorum to determine if a GPFS file system stays mounted or not, and it is easy to break the quorum requirement when adding or deleting nodes.

Note that adding or deleting nodes automatically configures them for GPFS usage. Newly added nodes are considered GPFS nodes in a down state, and are not recognized until a restart of GPFS. By maintaining quorum, you ensure that you can schedule a good time to refresh GPFS on the nodes.

For example, consider a GPFS configuration of four nodes. The quorum is three. With all four nodes running, we can add or delete one node and the quorum requirement is still satisfied. We can add up to three nodes into the GPFS group, as long as all four current nodes stay up. If we try to add four nodes, the GPFS group consists then of eight nodes, with a quorum requirement of five. However, at that point, GPFS can only see four nodes up (configured) and exits on all the current nodes.

Deleting a File System

The command to delete a file system is `mmdelfs`. Before you delete a GPFS file system, however, you must unmount it from all GPFS nodes.

For example, if you want to delete `fs1` (shown in Figure 180 on page 423), you can run `umount fs1` on all GPFS nodes, then run `mmdelfs fs1`.

Checking and Repairing a File System

If a file system cannot be mounted or if messages are received saying that a file cannot be read, it is possible to have GPFS check and repair any

repairable damages to the file system. The file system has to be in the unmounted state for GPFS to check it.

The command `mmfsck` checks for and repairs the following file inconsistencies:

- Blocks marked allocated that do not belong to any file. The blocks are marked free.
- Files for which an i-node is allocated, but no directory entry exists. `mmfsck` either creates a directory entry for the file in the `/lost+found` directory, or it destroys the file.
- Directory entries pointing to an i-node that is not allocated. `mmfsck` removes the entries.
- Ill-formed directory entries. They are removed.
- Incorrect link counts on files and directories. They are updated with the accurate counts.
- Cycles in the directory structure. Any detected cycles are broken. If the cycle is a disconnected one, the new top level directory is moved to the `/lost+found` directory.

File System Attributes

File system attributes can be listed with the `mmfsfs` command. If no flags are specified, all attributes are listed. For example, to list all the attributes of `fs1`, run:

```
mmfsfs fs1
```

To change file system attributes, use the `mmchfs` command. The following eight attributes can be changed:

- Automatic mount of file system at GPFS startup
- Maximum number of files
- Default Metadata Replication
- Quota Enforcement
- Default Data Replication
- Stripe Method
- Mount point
- Migrate file system

For example, to change the file system to permit data replication, run:

```
mmchfs -r 2
```

Querying and Changing File Replication Attributes

The command `mmfsattr` shows the replication factors for one or more files. If it is necessary to change this, use the `mmchattr` command.

For example, to list the replication factors for a file `/gpfs/fs1/test.file`, run

```
mmfsattr /gpfs/fs1/test.file
```

Say the value turns out to be 1 for data replication and you want to change this to 2, run:

```
mmchattr -r 2 /gpfs/fs1/test.file
```

Re striping a GPFS File System

If disks have been added to a GPFS, you may want to restripe the file system data across all the disks to improve system performance. This is particularly useful if the file system is seldom updated, for the data has not had a chance to propagate out to the new disk(s). To do this, run:

```
mmrestripefs
```

There are three options with this command; any one of the three must be chosen. The `-b` flag stands for rebalancing. This is used when you simply want to restripe the files across the disks in the file system. The `-m` flag stands for migration. This option moves all critical data from any suspended disk in the file system. Critical data is all data that would be lost if the currently suspended disk(s) are removed. The `-r` flag stands for replication. This migrates all data from a suspended disk and restores all replicated files in the file system according to their replication factor.

For example, when a disk has been added to `fs1` and you are ready to restripe the data onto this new disk, run:

```
mmrestripefs fs1 -b
```

Query File System Space

The AIX command `df` shows the amount of free space left in a file system. This can also be run on a GPFS file system. However, to obtain information

regarding how balanced the GPFS file system is, the command to use is `mmdf`. This command is run against a specific GPFS file system and shows the VSDs which make up this file system and the amount of free space within each VSD.

For example, to check on the GPFS file system `fs1` and the amount of free space within each VSD which houses it, run:

```
mmdf fs1
```

14.4.4 Migration and Coexistence

The improvements in GPFS 1.2 make it necessary that all nodes in a GPFS domain be at the same level of GPFS code. That is, in a GPFS domain, you cannot run both GPFS 1.1 and 1.2.

It is, however, possible to run multiple levels of GPFS codes, provided that each level is in its own group, within one system partition.

There are two possible scenarios to migrate to GPFS 1.2 from previous versions: full and staged. As its name implies, a full migration means that all the GPFS nodes within a system are installed with GPFS 1.2. A staged migration means that certain nodes are selected to form a GPFS group with GPFS 1.2 installed. Once you are convinced by this test group that it is safe to do, you can migrate the rest of your system.

Migration and coexistence are further described in *PSSP 3.1 Announcement*, SG24-5332, and *IBM General Parallel File System for AIX: Installation and Migration Guide*, SA22-7278.

432 The RS/6000 SP Inside Out

Chapter 15. High Availability

The RS/6000 SP system is a popular choice for businesses because of its scalability, flexibility and manageability. Its potential for high uptime also contributes to its value in the marketplace. Two software products that contribute to high system uptime are available on the RS/6000 SP: High Availability Cluster Multiprocessing Enhanced Scalability (HACMP/ES) and High Availability Cluster Multiprocessing Enhanced Scalability Concurrent Resource Manager (HACMP/ESCRM).

While fault-tolerant systems provide for high availability through expensive hardware redundancy, IBM's HACMP/ES and HACMP/ESCRM software provide a low-cost commercial computing environment that ensures mission-critical applications can recover quickly from hardware and software failure. This is a significant advantage over the fault-tolerant systems where software failures are not taken care of. Moreover, in fault-tolerant systems, the redundant hardware components do no processing in normal circumstances. While fault-tolerant systems surpass high availability systems in terms of service interruptions, high availability stands out for its low-cost, minimal service interruptions and flexibility of configuration. Shared resources, such as shared IP addresses for high network availability and shared volume groups for data availability, provide the foundation of the high availability features of the HACMP family of products.

HACMP/ES supports up to 32 SP nodes or RS/6000 systems. Development is underway to enhance the scalability to support up to 128 nodes. HACMP/ESCRM adds on to HACMP/ES, providing concurrent shared-access management for supported RAID and SSA subsystems. It supports up to 8 SP nodes or RS/6000 systems.

15.1 Components and Relationships

HACMP/ES and HACMP/ESCRM rely on the RS/6000 Cluster Technology (RSCT) for event detection and heartbeat. Initially part of the IBM PSSP for AIX Availability Services, RSCT is now an integral part of the HACMP/ES software. Refer to Chapter 8, "RS/6000 Cluster Technology" on page 185 for details on RSCT.

With this enhancement, HACMP/ES and HACMP/ESCRM can support these configurations:

- A cluster of standalone RS/6000 systems
- A mixed cluster of RS/6000 systems and RS/6000 SP nodes

- A cluster of RS/6000 SP nodes in different system partitions or different RS/6000 SPs

15.2 Modes of Operation

HACMP/ES supports two types of configurations: *cascading* and *rotating*. In these configurations, each shared volume group can belong to any one system, but not to more than one at a time. It comes with a Cluster Single-Point-of-Control (C-SPOC) facility that does dynamic reconfiguration of shared volume groups to ensure consistency of information across the cluster nodes. With this new introduction, common cluster administration tasks can be performed from any node in the cluster.

Another useful feature of HACMP/ES is the Dynamic Automatic Reconfiguration Event (DARE). This utility improves cluster management by allowing the placement alteration of resource groups to specific cluster nodes using the `cl dare` command.

In a cascading configuration, priority is given to each resource group. The node with the highest priority for a given resource group will take precedence over that group. If the node fails, the node with the next highest priority will take over that resource group. When a node with a higher priority rejoins the cluster, it takes over the resource group.

In a rotating configuration, priority is given on a first-come-first-served basis. The node that joins the cluster first will own those resource groups to which it has access. It releases those resource groups only when it fails or leaves the cluster with the takeover option turned on. The next node in line will then take control of the resource group. Upon rejoining, the original node that owned the resource group does not attempt to take back the resource group. So, in a rotating configuration, a resource group does not have a fixed node it attaches to.

HACMP/ESCRM adds to HACMP/ES by providing *concurrent* access to shared volume groups. This is to say more than one system can access a shared volume group at the same time. A distributed locking mechanism must be in place to prevent data contention. One of the restrictions on the use of concurrent volume groups is that it handles only raw logical volumes. With this restriction in place, the type of disk subsystems becomes limited.

Supported IBM devices include:

- 7133 and 7131-405 SSA disk subsystems in non-RAID configurations
- 9333 disk subsystems

- 7135-110 and 7135-210 RAIDiant arrays
- 7137 Disk Arrays

15.3 Planning Considerations

When configuring HACMP/ES or HACMP/ESCRM, allow an adequate amount of time for planning. Proper planning ensures easier installation and administration, higher availability, better performance, and less interruption to the cluster. This section discusses the various areas of consideration and provides suggestions on how to properly configure a HACMP/ES or HACMP/ESCRM environment. The rest of this section will use HACMP/ES as a reference to both HACMP/ES and HACMP/ESCRM unless otherwise stated.

15.3.1 Cluster

With HACMP/ES, you can include up to 32 nodes in a single SP partition. Alternatively, you can define clusters that are not contained within a single SP partition. Consider the following points when planning for cluster nodes:

- Nodes that have entirely separate functions and do not share resources should not be combined in a single cluster. Instead, create several smaller clusters on the SP. Smaller clusters are easier to design, implement, and maintain.
- For performance reasons, it may be desirable to use multiple nodes to support the same application. To provide mutual takeover services, the application must be designed in a manner that allows multiple instances of the application to run on the same node.
- In certain configurations, including additional nodes in the cluster design can increase the level of availability provided by the cluster; it also gives you more flexibility in planning node fallover and reintegration.

15.3.2 Application

Application availability is the key item to look at when considering the purchase of high availability products. Plan carefully in this area before implementing HACMP/ES to take full advantage of its capabilities. Some considerations that must be considered include:

- The application and its data should be laid out such that only the data resides on shared external disks while the application resides on each node that is capable of running it. This arrangement prevents software license violations and simplifies failure recovery. Ensure each node has

the required number of licenses available to run the application. However, if the application is not license-bound (for example, in-house developed applications), then having it on the shared external disks will be a better choice from an administrative point of view because the administrator will then need to contend with only one copy of the application.

- Start and stop scripts should be robust enough to handle the application on all related nodes. They should be able to handle abnormal termination of the application, such as node crash, providing speedy recovery. These scripts must be able to handle unsuccessful starting or stopping of applications. Failure to exit completely from a script constitutes an event error in HACMP/ES.

Certain event errors will cause HACMP/ES to hang in a limbo state and refuse to carry out all other tasks that follow. We recommend that you not include interactive statements in these scripts, as there is no facility for user inputs. HACMP/ES start, stop and takeover should be automatic and should be as transparent to users as possible.

A good practice when implementing such scripts is to test them fully on a node and then on another node without having HACMP/ES software involvement. HACMP/ES software merely kicks off the scripts in the order you tell it to; it does not carry out any recovery procedure on its own. Successful implementation of these scripts depends on how well they are written.

- Modularize the start and stop scripts. In debugging, this helps to determine where a problem lies. Whenever possible, let HACMP/ES call one start and one stop script. Within the start and stop scripts, each will call function statements to start or stop particular applications. Having one start and one stop script simplifies administrative tasks. Having modules, or functions, within the scripts make debugging and changes easier.
- Consider performance when deciding which nodes to use for takeover functions. An overloaded node will not be able to handle additional workload from another failed node. Forcing it to take over more workload will affect performance not only of applications from the failed node but also of its own. Where possible, spread out the load between processors, keeping in mind the amount of load each has to take when it takes over other nodes' responsibilities. Hardware upgrades may be required to boost performance.

15.3.3 Network

In a cluster environment, each node must have network connectivity to other nodes. This is required for communications between nodes to update each

other's status. Client machines are served through these network channels to access the applications. Keep in mind the following considerations when planning for your network layout:

- Eliminate the network as a single point of failure.

Having more than one network available to client machines insures against inaccessibility of nodes due to network failure. This, however, means added cost to network hardware. Supported TCP/IP-based networks include:

- Ethernet (802.3 Ethernet not supported)
- Token-Ring
- Fiber Distributed Data Interchange (FDDI)
- ATM and ATM LAN Emulation

SOCC or SLIP networks are not supported in HACMP/ES cluster.

On the SP, there are 3 SP-specific networks available:

- SP Ethernet

This network is used for software installation and administrative purposes only and should not be included in the HACMP/ES for external access by clients. However, it can be included in HACMP/ES for use as a heartbeat network

- SP Switch

This high speed network connects all SP nodes and is generally used for large data transfers. It is configurable in HACMP/ES for IP Address Takeover (IPAT).

- SP Serial Network

Spanning across all nodes and the CWS is a serial network. It is used for administrative purposes only and is not configurable under HACMP/ES.

- Eliminate the TCP/IP subsystem as a single point of failure.

It is important that nodes within a cluster are aware of each other's status. They keep each other updated through all the network channels. Networks that rely on TCP/IP for communication run a risk of failure on the TCP/IP subsystem. A cluster that is totally reliant on TCP/IP networks for heartbeat transmission will break down once the TCP/IP subsystem fails. To prevent such a failure, a non-IP network should be present in the cluster. Currently, support for non-IP networks includes target mode SCSI (tm SCSI), target mode SSA (tm SSA) and serial (rs232) networks.

- Eliminate Network Adapters as a single point of failure.

Having multiple networks guards against network failures. However, where cost is a concern, there may only be one network available for access by clients. In such a case, we can at least protect ourselves from network adapter failures by having more than just one network adapter. In an IP Take Over (IPAT) environment, for each service adapter, there has to be an equivalent standby adapter on the same node. The SP Switch network is an exception where IPAT is concerned. It cannot have a standby adapter. Failure on an SP Switch adapter can be promoted to a node failure through the use of the AIX error notification facility depending on how important the SP Switch is.

15.3.4 Data Loss Protection

In order to protect against data loss, we have to decide on a mechanism to eliminate data loss as a single point of failure. Two ways to ensure against data loss are as follows:

- Logical Volume Manager Mirroring

The logical volume manager (LVM) in AIX provides a means of mirroring all logical volumes. *Mirroring* means having more than one copy of a physical partition associated with a logical partition. These copies can either reside on the same hard disk or they can reside on different hard disks. It is advisable that mirrored copies reside on different hard disks to prevent loss of data in the event of a hard disk crash. With mirroring in place, should an error occur in one copy of the physical partition, the remaining copy can still function. This means uninterrupted access to the data. Mirroring applies to both logical volumes as well as journal logs.

In a *non-concurrent* configuration, we advise you to have quorum turned off for volume groups with mirroring. This provides continuous access to data even when some copies of data are bad. On restarting HACMP/ES, it also prevents volume groups with missing disks from being activated (varyonvg).

In a *concurrent* configuration, quorum must be enabled. Disabling quorum may result in data corruption. Multiple failures, that result in no common shared disk between cluster nodes, have the potential for data corruption or inconsistency because each node may be accessing different copies of the same data and thus updating them differently.

- Redundant Array of Independent Disks (RAID)

IBM 7135 models 110 and 210 provide supported RAIDiant technology in an HACMP/ES environment. It contains a group of hard disks that work together to provide large storage capacity and high I/O rates.

In a RAID configuration, LVM mirroring must not be used to mirror the same set of data. There are four levels in which it can operate:

- RAID level 0

Data is striped across a bank of disks in the array to improve throughput. It does not provide data redundancy and is not recommended for use in HACMP/ES clusters.

- RAID level 1

Data redundancy is maintained in multiple copies of the data on separate disks. This is similar to LVM mirroring, but is done at the hardware level.

- RAID level 3

This is used with raw disks only and is seldom used.

- RAID level 5

Data redundancy is maintained through the use of parity bits that allow data on a particular failed drive to be reconstructed from existing drives. This is the most commonly used mode. However, with this mode, about 20 to 30 percent storage overhead is expected in maintaining parity information.

To avoid having adapters be a single point of failure, we recommend that original and mirrored hard disks be connected to separate adapters on the same system. This is illustrated in Figure 182 on page 440.

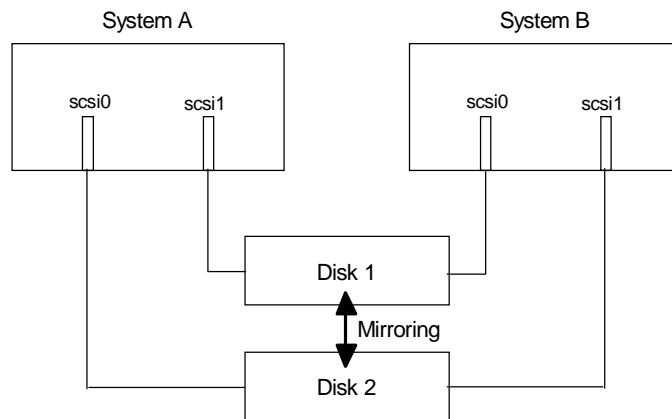


Figure 182. Avoiding Adapters and Disks as Single Point of Failure

15.3.5 Resource Group

HACMP/ES needs to know how to manage each of the resources that is configured. Resources refer to volume groups, service IP addresses, file systems, network file systems, and application servers (including start and stop scripts). Resource groups are created to indicate to HACMP/ES where each resource belongs. While HACMP/ES does not dictate the number of resources that go into a resource group or the number of resource groups in a cluster, it is advisable to keep the design simple. Following are a few general considerations to keep in mind:

- Every cluster resource must be part of one (and only one) resource group.
- You should group resources that cater to the same application in the same resource group.
- A rotating resource group must have a service IP label defined for it.
- A cascading resource group need not necessarily contain a service IP label. If IPAT is required, then the service IP label must be included in one of the resource groups.
- If a node belongs to more than one resource chain, it must have the ability to manage all resource groups simultaneously.
- Cascading or rotating resource groups cannot contain concurrent volume groups.
- Concurrent resource groups contain only application servers and concurrent volume groups.

15.4 Cluster Configuration

With proper planning in place, we can proceed with setting up the HACMP/ES cluster. Users familiar with *HACMP for AIX* will have no problem understanding the steps that follow. Along the way, features associated with HACMP/ES are discussed. The information provided here is not meant to be exhaustive. For details on setup procedure, refer to *HACMP for AIX: Enhanced Scalability Installation and Administration Guide*, SC23-4284.

15.4.1 Server

This section explains some of the terms used in HACMP/ES as well as the general guidelines you have to follow during configuration of the servers.

15.4.1.1 Networks

There are three types of networks to consider when configuring HACMP/ES.

Private Network

A private network is considered a network that is inaccessible by outside hosts. For the SP complex, this refers to the SP Administrative Ethernet and the SP Switch.

- SP Administrative Ethernet

It is advised that the SP Administrative Ethernet be used only as a private network within the SP complex. In HACMP/ES, it is configured as a private network providing a heartbeat channel for keepalive packets.

- SP Switch

The SP Switch is also classified as a private network in HACMP/ES with connections to SP nodes. Unlike the old High Performance Switch (HiPS) where HACMP/ES has to manage the Eprimary takeover, the SP Switch does not allow HACMP/ES to control the Eprimary.

When using SP Switch in HACMP/ES, Address Resolution Protocol (ARP) must be enabled. To find out if a node has ARP turned on, use the following command:

```
# dsh -w <host> "/usr/lpp/ssp/css/ifconfig css0"
```

If NOARP appears on the output, it reflects that ARP is turned off for that node and will require steps to turn it on. There are two methods to turn ARP on.

The first method requires the node to be customized after setting the SDR to enable ARP. Refer to *Parallel System Support Programs for AIX: Administration Guide*, SA22-7348 for the procedure.

The second method requires modifications to the node's ODM to turn it on manually. Do this with the following steps:

Important

The steps listed must be followed exactly and completely. Ensure that the CuAt file is backed up as directed. Without this backup copy, any error made may end up with the re-installation of the node. Mistakes made can be fixed by replacing the CuAt file with the backup copy and then rebooting the node. When in doubt, do not use this method; use the first method to customize the node.

1. Backup the node's CuAt.

```
# dsh -w <host> "cp -p /etc/objrepos/CuAt /etc/objrepos/CuAt.save"
```

2. Save css0 ODM information with ARP value "yes" to temporary file.

```
# dsh -w <host> "odmget -q 'name=css and attribute=arp_enabled' \  
CuAt | sed s/no/yes/ > /tmp/arpon.css"
```

3. Replace ODM information for css0 with the ARP value set to "yes".

```
# dsh -w <host> "odmchange -q 'name=css and attribute=arp_enabled' \  
-o CuAt /tmp/arpon.css"
```

4. Verify that the ODM has the ARP value set to "yes".

```
# dsh -w <host> "odmget -q 'name=css and attribute=arp_enabled' CuAt"
```

5. Remove the temporary file.

```
# dsh -w <host> "rm /tmp/arpon.css"
```

• Asynchronous Transfer Mode (ATM)

ATM is considered a private network since it provides a point-to-point connection between two systems and does not do broadcasting. It uses connection-oriented technology as opposed to IP, which is a datagram-oriented technology. As such, configuring IP addresses as on ATM is complicated. With an ATM switch, only Switched Virtual Circuits (SVC) can be used. Hardware address takeover is not supported in IPAT. An ATM ARP server cannot be an HACMP/ES server.

Public Network

Public networks are used by external hosts to access the cluster nodes. They can be classified under the following categories:

- Ethernet, Token-Ring, and FDDI

These adapters are normally used for external connectivity to client machines. When configured for IPAT, each service adapter will be accompanied by at least one similar standby adapter. Up to seven standby adapters can be configured for each network.

- ATM LAN Emulation

With ATM configured, ATM LAN emulation can be configured to bridge the ATM network with existing Ethernet or Token-Ring networks. Each adapter configured can take the form of either ethernet or token-ring. Like the classic ATM, hardware address takeover is not supported. The network configured is classified as public under HACMP/ES.

Serial Network

Serial networks do not rely on TCP/IP for communication. As such, they provide a reliable means of connection between cluster nodes for heartbeat transmissions.

- RS232

This is the most commonly used non-IP network, as it is cheap and reliable. It is a crossed (null modem) cable connected to a serial port on the node. Earlier model nodes like the 62MHz Thin (7012-370), 66MHz Thin (7012-390), 66MHz Wide (7013-590), 66MHz Thin2 (7012-39H), 77MHz Wide (7013-591) and 66MHz Wide (7013-59H) do not have supported native serial ports and therefore require the use of the 8-port or 16-port asynchronous adapters. Extension nodes S70 and S7A will require PCI multi-port asynchronous adapters to support this feature.

- TMSCSI

Like the RS232 connection, TMSCSI is non-IP reliant. It is configured on the SCSI-2 Differential bus to provide HACMP/ES clusters with a reliable network for heartbeat transmission.

- TMSSA

The TMSSA is a new feature of HACMP/ES, which allows SSA Multi-Initiator RAID adapters to provide another channel for keepalive packets. This feature is supported only when PTF2 or greater is installed.

15.4.1.2 Shared Devices and LVM Components

HACMP/ES requires that shared volume groups be configured on external disks accessible by cluster nodes sharing the same volume group information. Supported disk subsystems include:

- Small Computer System Interface (SCSI)
- Redundant Array of Independent Disks (RAID)

- Serial Storage Architecture (SSA)
- Versatile Storage Server (VSS)

Take note that single-ended SCSI disks cannot be used to configure shared volume groups. When configuring for twin-tail access of SCSI-2 Differential disks, be careful to avoid having a SCSI address clash on the same SCSI bus.

Consider the following when configuring shared volume groups and LVM components for Non-Concurrent Access:

- All shared volume groups must have the auto-varyon feature turned off.
- The major number for a shared volume group must be the same on all cluster nodes using NFS mounted file systems. However, when using NFS, it is also recommended that the major number be kept the same for easier management and control. The `lv1stmajor` command can be used to check for available major numbers on each node.
- All journaled file system logs, logical volume names and file system mount points must be unique throughout the cluster nodes.
- All file systems must have the auto-mount feature turned off.
- For shared volume groups with mirroring, we recommend that quorum is turned off.

LVM components for Concurrent Access cannot contain journaled file systems, since concurrent mode supports only raw logical volumes. In this mode, major number for the shared volume group is not important.

The procedure for making concurrent capable volume groups on serial disk subsystems differs from that on RAID disk subsystems. Refer to *HACMP for AIX: Enhanced Scalability Installation and Administration Guide, SC23-4284* for details on how to configure the different disk subsystems for concurrent access.

15.4.1.3 Preparing AIX

In order for HACMP/ES to be configured properly, the following components in AIX must be set correctly:

- Entries in `/.rhosts` must include all cluster nodes. This is required to allow synchronization of HACMP/ES configuration information between cluster nodes. During synchronization, the `/usr/es/sbin/cluster/godm` daemon is used. HACMP/ES is equipped to take advantage of the Kerberos authentication if configured on the systems. In such a setup, the `/.rhosts` file is not required.

- User and Group IDs must be the same on all nodes. This is to allow users to be able to login on the takeover system in case their main server fails. With the RS/6000 SP, user management can be used to enforce this. In HACMP/ES, groups and users management is made easy through the use of these commands: `cl_mkuser`, `cl_lsuser`, `cl_chuser`, `cl_rmuser`, `cl_mkggroup`, `cl_lsgroup`, `cl_chgroup`, `cl_rmggroup`.

They can be accessed via the SMIT fastpath:

```
# smitty cl_usergroup
```

Ensure that all users' home directories are created on all nodes if they are not located on the shared disks.

- All hostnames must be resolvable. Either the `/etc/hosts` file or the nameserver must contain all IP addresses referenced by HACMP/ES.
- The following network options must be set to 1: *nonlocsrcroute*, *bcastping*, *ipsrcroutesend*, *ipsrcrouterecv*, *ipsrcrouteforward*.
- I/O pacing needs to be turned on for a HACMP/ES cluster to work properly during large disk writes. A high water mark of 33 and a low water mark of 24 is recommended for a start. Failure to turn on I/O pacing may result in nodes crashing during large disk write activities. To do so, use the following command:

```
# chdev -l sys0 -a maxpout=33 -a minpout=24
```

- A `syncd` value of 10 is recommended. `Syncd` has the function of flushing the contents of the RAM and writes to disk what has been updated. In many cluster setups, nodes crash due to `syncd` not releasing the `jfs_sync_lock` during high node activities, thus preventing `clstrmgr` from processing heartbeats. To make the necessary change, search for the following sentence in `/sbin/rc.boot` and modify the default value of 60 to 10.

```
nohup /usr/sbin/syncd 10 > /dev/null 2>&1 &
```

15.4.1.4 Installing HACMP/ES

When installing the software, ensure free space of approximately 52 MB in `/usr` and about 500 KB in `/` (root) file systems. The following AIX filesets are prerequisites for HACMP/ES:

- `bos.adt.lib`
- `bos.adt.libm`
- `bos.adt.syscalls`
- `bos.net.tcp.client`
- `bos.net.tcp.server`

- bos.rte.SRC
- bos.rte.libc
- bos.rte.libcfg
- bos.rte.libcur
- bos.rte.libpthreads
- bos.rte.odm
- bos.rte.lvm.usr 4.3.2 or higher (for CRM)

Install the components that you require for your nodes, bearing in mind the minimum required filesets are:

- rsct.basic.*
- rsct.clients.*
- cluster.es.*
- cluster.cspoc.* (automatically installed when cluster.es.* is being installed)
- cluster.clvm.* (for CRM only)
- cluster.hc.* (for CRM only)

The latest Program Temporary Fixes (PTFs) must be installed together with the base filesets. Without PTFs, certain functionality mentioned in this book may not be applicable. One example is the support of TMSSA.

15.4.1.5 Cluster Information Program Clinfo

The clinfo daemon provides status information about the cluster to cluster nodes and clients. Each node has a copy of `/usr/sbin/cluster/etc/clhosts` file. This file is required by the clinfo daemon to know which nodes or clients can be communicated. It contains resolvable hostnames or IP addresses of all boot and service labels of servers and clients. Take care to not leave this file empty or include standby addresses, as this may affect the way clinfo or clstat work.

When an event occurs in the cluster, the `/usr/sbin/cluster/etc/clinfo.rc` script is executed to update the system's ARP cache. Include all client IP addresses or hostnames to the `PING_CLIENT_LIST` variable. This is especially important where no hardware address swap is configured.

15.4.1.6 Configure Cluster

Configuring HACMP/ES is very similar to configuring HACMP for AIX. The following steps provide a general description of how to configure HACMP/ES.

You have to be a root user to do the configurations. Configuration of a cluster must be done on a single node in the cluster. The information will then be propagated to the rest of the nodes with the synchronization features built into the HACMP/ES software.

Define Cluster Topology

To define the cluster topology, the following steps are taken. All the steps are done on one node.

- Define a cluster by providing a cluster ID and cluster name. Note that in any network where there are multiple clusters, each cluster IDs must be unique. Failure to conform to this rule can cause nodes to crash unexpectedly.

The SMIT fastpath is:

```
# smitty cm_config_cluster
```

- Define all the nodes in the cluster. It is advisable to use hostnames as the node names for easy reference.

The SMIT fastpath is:

```
# smitty cm_config_nodes
```

- Define all network adapters. This includes the serial network, service IP labels, boot IP labels and standby IP labels. Note that the SP Switch is configured under Network Type "hps".

The SMIT fastpath is:

```
# smitty cm_config_adapters
```

- Define a global network. In a cluster where nodes belong to different SP partitions, each SP administrative ethernet will be on its own subnets. In HACMP/ES, you must define these subnets as one global network for heartbeat transmission. For example, in Figure 183 on page 448, we have two frames partitioned into Subnet1 (192.168.3.0) and Subnet2 (192.168.4.0).

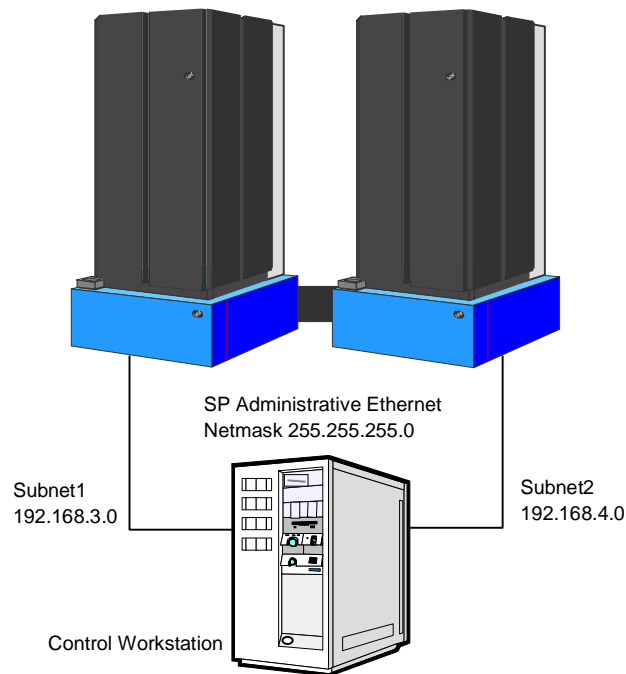


Figure 183. SP Administrative Ethernet on Different Subnets

All routes are defined to ensure that nodes on Subnet1 can communicate to nodes on Subnet2. To ensure heartbeat transmission can flow from nodes on Subnet1 to nodes on Subnet2, we add these 2 subnets to a global network SPGlobal. The following commands show how:

```
# /usr/sbin/cluster/utilities/claddnetwork -u Subnet1:SPGlobal
# /usr/sbin/cluster/utilities/claddnetwork -u Subnet2:SPGlobal
```

The SMIT fastpath is:

```
# smitty cm_config_global_networks.select
```

In case you want to remove Subnet1 from SPGlobal, use this command:

```
# /usr/sbin/cluster/utilities/claddnetwork -u Subnet1
```

- Tune the Network Module. The heartbeat rate can be tuned for all networks defined. If any network is expected to be heavily loaded, the heartbeat rate can be tuned to "slow" to avoid false failure detection over that network. Failure detection is dependent on the *fastest* network in the cluster. For a start, leave the settings as default. Tune only when you experience false failure detections.

- Synchronize the cluster definition to all nodes. You will have to ensure that all hostnames are defined in `/.rhosts` if you are using the AIX authentication method. If you are using Kerberos authentication across the nodes and want to use this instead of `/.rhosts`, follow these steps:
 1. Ensure `/usr/lpp/ssp/kerberos/bin` and `/usr/lpp/ssp/rcmd/bin` are included in the root user path.
 2. Delete all `/.rhosts` files.
 3. Set the cluster security mode to Enhanced


```
# /usr/sbin/cluster/utilities/clchclstr -s Enhanced
```

 The SMIT fastpath is:


```
# smitty cm_config_security.changeshow
```
 4. Run `/usr/sbin/cluster/sbin/cl_setup_kerberos`.
 5. Synchronize cluster definition.
 6. Delete the `/usr/sbin/cluster/sbin/cl_krb_service` file which contains the Kerberos service principals password which you entered earlier when running the `cl_setup_kerberos` command.

Define Resources

In order to associate all resources with each node, resource groups must be configured. Use the following procedures to set up resources for the cluster:

- Define resource groups. When creating resource groups, it is important to note that when specifying Participating Node Names for cascading or rotating, the priority decreases in the order you specify. For concurrent mode, the order does not matter.
- Define application servers. This will tell HACMP/ES which scripts are used for starting the applications and which to stop.
- Define resources for resource groups. Here, you will input all the related resources which you want HACMP/ES to manage for you. Take note that IPAT is required if you are using the NFS mount option. Also, you must set the value of `Filesystems mounted before IP configured` to `true`.
- Configure run time parameters. If you use NIS or nameserver, set the value to `true`. Leave the Debug Level as high to ensure that all events are captured.
- Configure cluster events. You can customize the way HACMP/ES handles events by incorporating pre-event, post-event scripts, notification or even recovery scripts. How you want HACMP/ES to react depends on how you configure these events. The default is generally good enough.

- Synchronize the resource definitions.

15.4.2 Client

In order for an RS/6000 system to receive messages about what is happening on the cluster, it must be configured as a client running the clinfo daemon. Use the following steps to do this:

- Install Base Client Software

Install the following minimum filesets in order for your client system to access any cluster information:

- rsct.clients.*
- cluster.es.client.*

- Edit `/usr/sbin/cluster/etc/clhosts`

Like the server, this file contains resolvable hostnames or IP addresses of all boot and service labels of servers and clients. The clinfo daemon uses names in this file to communicate with the clsmuxpd process on the servers. This file must *not* be left empty, and you must not include standby addresses in it.

- Edit `/usr/sbin/cluster/etc/clinfo.rc`

This file is used the same way as in the servers. Include all client IP addresses or hostnames to the PING_CLIENT_LIST variable. This is especially important where no hardware address swap is configured and where there are clients that do not run the clinfo daemon. Be sure to update the ARP tables on routers and systems that are not running clinfo when IPAT occurs.

- Reboot Clients

Reboot your clients machines to finish the setup.

15.5 Starting and Stopping Cluster Services

With HACMP/ES, starting and stopping cluster services is made easy with the introduction of CSPOC. Instead of having to start cluster services one node at a time, it is now possible to start some or all nodes from a single node. The command that is used to start cluster services is

`/usr/sbin/cluster/sbin/cl_rc.cluster`. This command will call the `/usr/sbin/cluster/etc/rc.cluster` command to start up cluster services on the nodes in a sequential order. The remote execution is basically done through the `rsh` command. The SMIT fastpath is:

```
# smitty cl_clstart.dialog
```

The normal `clstart` command to start up one node at a time is still available and is accessible via the SMIT fastpath:

```
# smitty clstart
```

To stop a cluster using CSPOC, the `/usr/sbin/cluster/sbin/cl_clstop` command is used. The SMIT fastpath is:

```
# smitty cl_clstop.dialog
```

To stop the cluster services one node at a time, use the SMIT fastpath:

```
# smitty clstop
```

Note that the **Forced** option has been removed from the SMIT screen and will not be available till later releases of HACMP/ES. The `clstop` script still retains this forced option, but you are advised not to use it.

In order to monitor cluster activity from a client system using the `clstat` command, the `clinfo` daemon must first be started up. To start it, follow the procedure shown in Figure 184.



Figure 184. Starting Clinfo Daemon on Client System

To start monitoring the cluster status, use the following command:

```
# /usr/sbin/cluster/clstat
```

The output is shown in Figure 185 on page 452.

```
dtterm
Window Edit Options Help

c1stat - HACMP for AIX Cluster Status Monitor

Cluster: SP3N5N6 (56)      Wed Feb 17 14:10:29 EST 1999
      State: UP           Nodes: 2
      SubState: STABLE

      Node: sp3n05      State: UP
      Interface: sp3sn05 (0)      Address: 192.168.13.5
                                   State: UP
      Interface: sp3n05 (1)      Address: 192.168.3.5
                                   State: UP

      Node: sp3n06      State: UP
      Interface: sp3sn06 (0)      Address: 192.168.13.6
                                   State: UP
      Interface: sp3n06 (1)      Address: 192.168.3.6
                                   State: UP

***** f/forward, b/back, r/refresh, q/quit *****
```

Figure 185. C1stat Output Screen

Chapter 16. Parallel Programming

This chapter describes how the SP can be used for parallel programming. Many of today's scientific and engineering problems can only be solved by using the combined computational power of parallel machines, and some also rely on high performance I/O capabilities. Typically, the applications in this area are highly specialized programs, developed and maintained in universities, government labs, or company research departments.

Parallel Environment for AIX (PE), program number 5765-543, provides the application programmer with the infrastructure for developing, running, debugging, and tuning parallel applications on the SP. In 16.1, "Parallel Operating Environment (POE)" on page 453, we describe the operating environment which allows you to run and control programs in parallel from a single node.

As each SP node is an individual RS/6000 machine with its own memory (invisible to other SP nodes), the programs running on different nodes do not share any data. Therefore, the second core component of PE is the Message Passing Interface (MPI) library, through which programs running on different SP nodes can exchange messages. This enables the otherwise independent programs to work collectively on one common job. MPI is discussed in 16.2, "Message Passing Libraries" on page 463, along with some related subjects.

In addition to explicit message passing by the use of message passing libraries, the data-parallel programming model is also supported on the SP through High Performance Fortran (HPF). We give an overview of HPF and some of the products available on the SP in 16.3, "High Performance Fortran" on page 476.

Finally, the development of parallel programs requires parallel debuggers to localize programming errors, and parallel profiling and tracing tools to find, understand and resolve performance problems. 16.4, "Parallel Programming Tools" on page 480 provides information about these tools.

16.1 Parallel Operating Environment (POE)

The heart of the Parallel Operating Environment (POE) is the `poe` command, which allows you to execute programs in parallel from a single node. POE allocates the nodes on which the application should run (as described in 16.1.4, "POE Resource Management" on page 460), starts the programs on these nodes, and redirects `stdin`, `stdout` and `stderr` between the POE *home*

node (on which the `poe` command was invoked) and the *remote nodes* which execute the application.

Before POE can be used, some prerequisites must be met:

- The user must exist with the same (non-root) userid on the home node and all the remote nodes. For security reasons, POE will refuse to run parallel applications as root (UID 0).
- The user must have authorization to run `rsh` to the remote nodes. On an SP system which is managed and used as a whole, listing all the SP nodes which are used for POE jobs in the `/etc/hosts.equiv` file is a natural way to ensure this. Using individual users' `$HOME/.rhosts` files is also possible, of course.
- The desired executable/script must be available on all remote nodes, under the same path name. This can be achieved by starting the application from within a global file system like NFS, DFS, AFS, or even GPFS. Alternatively, the executable can be copied to a local file system on the remote nodes before invoking `poe`.

Note

Starting POE Jobs on Large Systems: For large SP systems, starting an executable on many nodes from within a shared file system like NFS can cause performance problems. DFS has a better client to server ratio than NFS, and so supports a larger number of nodes that simultaneously read the same executable. As a parallel file system, GPFS is also able to sustain a higher number of clients simultaneously accessing the same executable.

Typically, POE is invoked by specifying the executable/script that should run on the remote nodes as the first argument of the `poe` command. Alternatively, POE can prompt interactively for the names of the programs to run, or read this information from a commands file. For example, to run the AIX `hostname` command on four processors and save the resulting output to a file, issue:

```
poe hostname -procs 4 > ./where_was_i
```

The `poe` command allocates the remote nodes, and on each of these nodes, it does the following:

- Initializes the local user environment, including a `chdir()` to the current directory of the home node.
- Runs the command with `stdin`, `stdout` and `stderr` connected to the `poe` process on the home node.

- Passes signals like SIGCONT, SIGINT, SIGQUIT, SIGTERM and SIGHUP to the remote nodes.
- Performs heartbeat checks of the remote nodes (unless the MP_PULSE environment variable is set to zero), and terminates the parallel application if any of the remote nodes are down.
- Gets notifications about abnormal termination of any of the user programs, and in response shuts down the whole parallel application.

The operation of POE can be controlled by a large number of POE environment variables, most of which can also be specified as command line options to `poe` (following the executable/script name). These options can be freely intermixed with command line options which are interpreted by the application. Before we describe some of these configuration options, an overview about the general POE architecture might be useful.

16.1.1 POE Architecture

POE is a client/server application. The `poe` command is the client side running on the home node, and communicates with the *partition manager daemons* (PMDs) running on each of the remote nodes. In PE 2.4, this server component is the `pmdv2` daemon. The protocol which POE uses internally is called *Structured Socket Messages* (SSM).

Each partition manager daemon starts and controls the program which runs on its node, and routes that program's stdin, stdout and stderr traffic to the `poe` command on the home node through SSM. This program may be any executable or (Korn shell) script; it need not be part of a parallel application in which the programs running on different nodes interact with each other.

The POE startup process differs depending on whether the `poe` command is invoked interactively, or from within LoadLeveler. The interactive case is shown in Figure 186 on page 456.

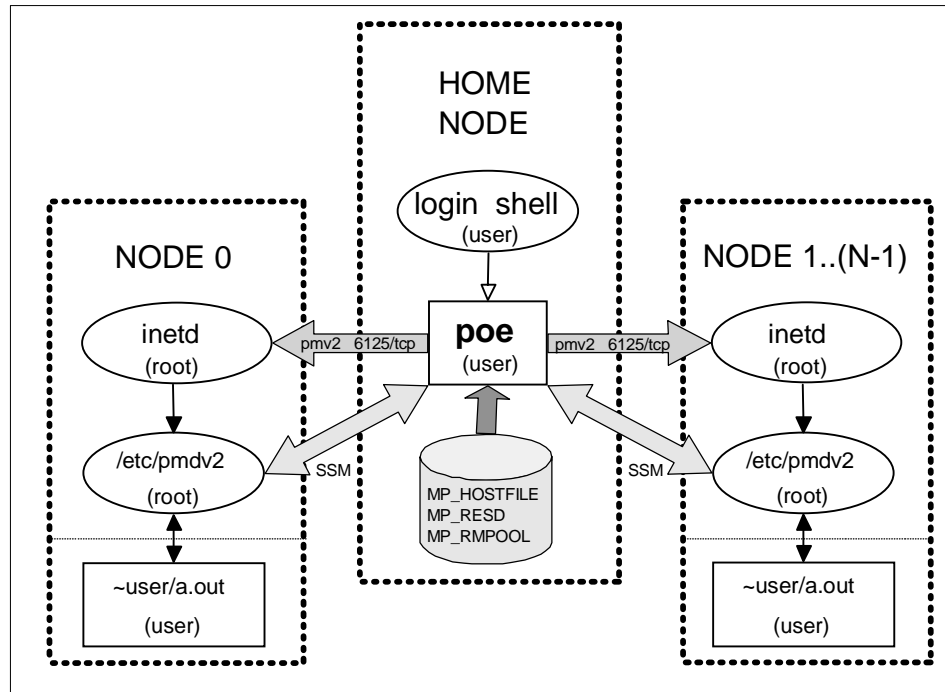


Figure 186. POE Startup for Interactive Use

If `poe` is called interactively, it determines the remote nodes by the resource allocation methods specified in POE environment variables and command line options. This is explained in 16.1.4, “POE Resource Management” on page 460, and might involve LoadLeveler. However, LoadLeveler is only used to do the resource allocation: the POE application is still interactive.

Note

Home Node versus Node 0: In interactive POE, there is no direct relation between the home node and any of the remote nodes. It is a common misconception that the home node is identical with node 0 of the parallel application. In general, this is not true for interactive POE jobs. The resource allocation mechanism chooses the remote nodes, and the selected set of nodes may or may not include the node on which the `poe` command was invoked.

The `poe` command then uses the `pmv2` service (normally port 6125/tcp) to start the `pmdv2` partition manager daemons on the remote nodes, through

the inetd daemon. The pmdv2 daemon runs under the identity of root, since it must check the user's authentication and set the priorities for the user's task. Finally, pmdv2 starts the application under the user's identity.

If POE is run in a LoadLeveler batch job, it is started differently. The resource allocation as well as the start of the POE application is done by LoadLeveler. On as many batch nodes as requested, LoadLeveler starts a LoadL_starter process under the identity of the user. It does so through the LoadL_master and LoadL_startd processes, which run on the batch nodes under the identity of LoadLeveler.

As shown in Figure 187, the LoadL_starter process on each node starts the pmdv2 partition manager daemon, which in this case runs under the identity of the user since all authentication checks and priority settings have been performed already by the LoadLeveler components. Finally, pmdv2 invokes the user's application as in the interactive case.

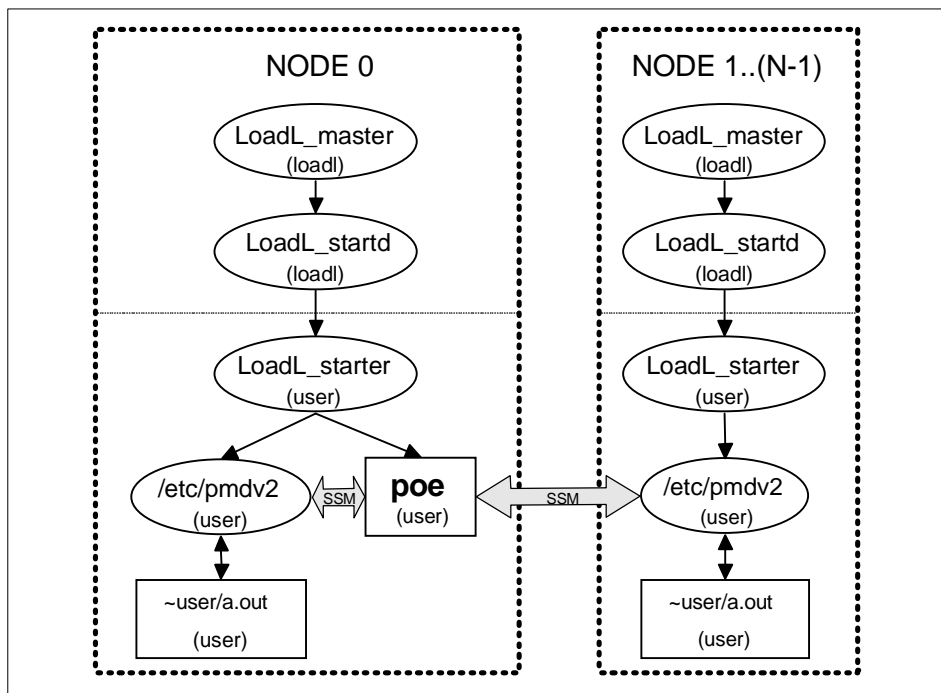


Figure 187. POE Startup under LoadLeveler

When running POE in a batch job, node 0 is a special case. Since there is no poe process already running somewhere, the LoadL_starter process on node 0 spawns two tasks: one which starts the partition manager daemon

and user application, and a second task which represents the home node and runs the `poe` command. The `poe` process running on node 0 communicates with the partition manager daemons (on node 0 as well as on the other nodes) through SSM, just as in the interactive case.

16.1.2 Controlling the POE Runtime Environment

The operation of POE can be influenced by more than 40 different environment variables, which are of the form `MP_POEOPTION`. For many of these options, a corresponding flag for the `poe` command exists. These flags do not have the “MP_” prefix, and are in lower case. For example, the flag corresponding to `MP_POEOPTION` would be `-poeoption`.

POE environment options can be grouped into the following set of functions:

- **Partition Manager Control.** This is the largest set of POE options. Apart from several timing selections, this group of environment variables also determines how resources (nodes, processors in case of SMP nodes, network adapters) are allocated. See 16.1.4, “POE Resource Management” on page 460 for details.
- **Job Specification.** `MP_CMDFILE` specifies a file from which to read node names for allocation. `MP_NEWJOB` can be used to run multiple job steps on the same set of nodes: if set to yes the partition manager will maintain the partition for future job steps. By using `MP_PGMMODEL=mpmd`, applications which use the MPMD programming model can be loaded.
- **I/O Control.** These settings specify how standard I/O which is routed to/from the remote nodes is handled at the home node. The four variables which control this are `MP_LABELIO`, which can be used to prefix each line of output with the task number where it originated, `MP_STDINMODE`, `MP_HOLD_STDIN`, and `MP_STDOUTMODE`.
- **VT Trace Collection.** These settings control VT trace generation, discussed in 16.4.2.1, “IBM Visualization Tool (VT)” on page 488. The level of tracing is set by `MP_TRACELEVEL`, which may take values 0, 1, 2, 3 or 9.
- **Generation of Diagnostic Information.** The most useful option in this category is `MP_INFOLEVEL`. It determines the level of message reporting. A value of 0 specifies no messages, 6 specifies maximum messages.
- **Message Passing Interface.** These environment variables can be used to tune MPI applications. See 16.2.2, “IBM’s Implementation of MPI” on page 466 for details.

See the `poe` page in *PE for AIX: Operation and Use, Volume 1, SC28-1979*, for a full description of all POE environment variables and command-line flags.

Note

Enforcing Global POE Settings: The `/etc/poe.limits` file can be used to enforce system-wide values of three POE options. Users cannot override the values in that file by setting the corresponding environment variables to a different value:

- `MP_BUFFER_MEM` can be used to limit the maximum value to which this variable can be set by users (they can select lower values).
- `MP_AUTH=DFS` can be used to globally enable DCE credential forwarding. See 16.1.5, “POE Security Integration” on page 461 for details.
- `MP_USE_LL=yes` causes POE to reject all POE jobs which are not run under `LoadLeveler`. This can be used to force users to use `LoadLeveler`.

16.1.3 The POE Compilation Scripts

Parallel applications are compiled by the same compilers which are used for serial programs on regular RS/6000 workstations. The following compilers are supported by Parallel Environment for AIX:

- C for AIX Version 4.3 or later, program number 5765-AAR
- C and C++ Compilers for AIX Version 3.6.4 or later, program number 5801-AAR
- XL Fortran Version 5.1.1 or later, program number 5808-AAR
- XL High Performance Fortran Version 1.3.1 or later, program number 5765-613

Note that C Set++ for AIX is withdrawn from marketing, and the VisualAge C++ Professional for AIX Version 4.0 compiler is not supported by Parallel Environment.

POE provides several compilation scripts which contain all the POE include and library paths, and automatically link the user's program with the POE and message passing libraries. Four scripts `mpcc`, `mpCC`, `mpxlf` and `mpxlf90` are available which call the `xlc`, `xlC`, and `xlf` compilers. The Fortran compilation scripts can be configured to use XL HPF instead of XL Fortran (if both are installed) by setting the environment variable `MP_HPF=yes`. These four

scripts also exist in a version which links threaded libraries (suffix `_r`), and in a version which supports checkpointing (suffix `_chkpt`). All these scripts do the following:

- Include `/usr/lpp/ppe.poe/include`
- Link the libraries `libmpci.a`, `libmpi.a`, `libvtd.a` and `libppe.a` (the Fortran scripts also link `libxlf90.a` and `libxlf.a`)
- Link POE's version of `libc.a`, which has its own `exit()` routine that allows synchronized profiling and capture of exit code
- Ensure that the POE runtime initialization routine is called before control is passed to the user program

Note

Initializing the POE Runtime: With AIX 4.2 and later, the POE compiler scripts now link the standard AIX `/lib/crt0.o`, and use the `-binitfni:poe_remote_main` linker option to initialize the POE runtime environment. With earlier releases of AIX which did not support `-binitfni`, POE linked a special `/usr/lpp/ppe.poe/lib/crt0.o`. The call in the executable which actually invokes `-binitfni` initializations is `modinit()`.

16.1.4 POE Resource Management

The allocation of CPUs and network adapters to parallel jobs is controlled by several variables. The hosts which should execute a parallel program can be specified in a host file listing the machines. The name of that file can be defined by the `MP_HOSTFILE` variable. This is straightforward, but normally does not balance the load on the system well. Remote nodes can also be determined by `LoadLeveler`, which should have a better view of the SP's utilization and allocate nodes accordingly. In releases prior to PSSP 3.1, node allocation has been done by the SP Resource Manager (or Job Manager). In PSSP 3.1, these functions have been integrated into `LoadLeveler`. The `MP_RESD` and `MP_RMPOOL` environment variables provide the information if, and from which pool, `LoadLeveler` (or the SP Resource Manager, in previous releases) should allocate nodes. Exclusive use of CPU resources can be requested by setting `MP_CPU_USE=unique`.

Through the environment variable `MP_EUILIB={ip|us}`, you can select if the IP protocol or the User Space (US) protocol should be used. For US communication, the SP Switch adapter (`css0`) will be used. If IP is specified, the `MP_EUIDEVICE` environment variable can be set to request a specific network adapter (`en0`, `css0`, `tr0`, and so on). `MP_ADAPTER_USE` can be set

to shared or dedicated to indicate that the adapter should be exclusively used by the current program. The value of this variable is honored only if LoadLeveler is using to start the job.

The number of tasks can be specified through three environment variables:

- `MP_PROCS` directly specifies the number of tasks in the parallel program. This is the traditional way to allocate CPUs on uni-processors.
- `MP_NODES` specifies the number of SP nodes that should be used. SP nodes could be multi-processor SMP nodes, and this option allows you to specify the number of SP nodes rather than the number of processors.
- `MP_TASKS_PER_NODE` can be used to allocate a fixed number of tasks on all the selected nodes. In conjunction with `MP_PROCS` or `MP_NODES`, this value can be used to describe CPU allocation. It is particularly useful on SMP nodes.

Combinations of these three variables can be specified, as long as they do not result in contradictory requirements.

For a more detailed description of the interrelationships of the POE options, refer to Chapter 2, “Executing Parallel Programs” of *Parallel Environment for AIX V2R4 Operation and Use, Volume 1*, SC28-1979.

16.1.5 POE Security Integration

If the SP is integrated into a DCE/DFS or AFS environment, a parallel job needs to establish appropriate user credentials on the remote nodes to be able to access the user’s data (like a DFS or AFS home directory). If POE is run as a LoadLeveler batch job, LoadLeveler will establish these credentials on the batch nodes. For interactive jobs, this has to be done differently.

Chapter 5, “Installation-Related Procedures” of *Parallel Environment for AIX Installation V2R4*, GC28-1981 contains details on how to enable AFS support in Parallel Environment. In particular, `/usr/lpp/ppe.poe/samples/afs` contains the sample files `gettokens.c` and `settokens.c`, which can be linked to the `poe` command and the `pmdv2` daemon to transparently transfer AFS tokens from the home node to the remote nodes.

Unfortunately, this method does not work with DCE credentials because the buffer space used by POE to transfer the credentials to the remote nodes is too small for DCE tickets.

Future releases of Parallel Environment might include facilities to transparently forward DCE credentials in a way similar to the current AFS

solution. In the meantime, the POE command `poeauth` can be used to explicitly distribute DCE credentials from node 0 (not the home node) to the other remote nodes. To do this, the environment variable `MP_AUTH` has to be set to indicate that DCE credentials should be forwarded. This is achieved by:

```
export MP_AUTH=DFS
```

There is no command line flag for this option. The default value for this variable is `AIX`. Note that even with `MP_AUTH=DFS`, the POE authorization check is still based on the `/etc/hosts.equiv` or `$HOME/.rhosts` file.

The `poeauth` command can then be invoked to copy the DCE credentials, but you have to understand what it does in order to successfully use it:

- Be aware that `poeauth` expects DCE credentials on node 0. This is not always the case, for example when `poe` is run from a login node but the nodes allocated for the actual computation come from another pool of nodes. You have to make sure that DCE credentials exist on node 0, either by a suitable hostfile which contains the home node, or by logging in to node 0 before calling `poeauth`.
- Be aware that `poeauth` is a normal POE application. POE initializes the user's environment on the remote nodes, which includes a `chdir()` to the directory in which it was called on the home node. This means that `poeauth` must not be called when the current directory is in `DFS`: POE would fail when attempting the `chdir()` to `DFS`, before the copying of DCE credentials could take place. You might therefore want to provide a circumvention for this problem, like the following script:

```
#!/bin/ksh
cd /tmp
/usr/lpp/ppe.poe/bin/poeauth $@
cd - >/dev/null
```

- Make sure that the parallel application runs on the same nodes to which `poeauth` has copied the credential files. This can be done by setting the POE environment variable `MP_NEWJOB=yes`.

If these issues are addressed, `poeauth` can be used to run interactive POE jobs which have full DCE user credentials on the remote nodes.

Note

Order of LPP Installation: If you plan to use Parallel Environment in combination with DCE (and DFS), make sure that the DCE Licensed Program Products (LPPs) are installed before you install Parallel Environment! Parallel Environment has DCE-related parts like the `poeauth` command for DCE credential forwarding, but the actual install files only contain an object file `poeauth.o`, no executable. The executable command will be built during installation, but only if the DCE filesets are already installed.

If you install Parallel Environment before DCE, the POE environment will not be completely built for DCE support. The safest solution in such a case is to deinstall POE, and reinstall it after DCE is installed.

16.2 Message Passing Libraries

While POE provides the environment to run programs in parallel, message passing libraries are required if these individual programs need to exchange application data to do their job. The need for message passing libraries is caused by the fact that in a cluster of machines, there is no common address space like in a Symmetric Multiprocessor (SMP), where all processors have shared access to the memory. To access data which resides in the memory of another node, a program needs to call a message passing library procedure which gets that data over the network.

When parallel programming on workstation clusters and parallel computers became popular, many different proprietary message passing systems were developed. The programs written for any one of these systems were normally not portable to other systems. To enhance portability of applications across different systems (and so protect the sometimes huge investments in the development of message passing applications), the Message Passing Interface Forum (MPIF) has (informally) standardized an API for explicit message passing called the Message Passing Interface (MPI). This section describes the MPI industry standard, some details of IBM's implementation of MPI, and a couple of other message passing systems available on the SP.

16.2.1 What is in MPI

The contents of MPI have been defined and published by the Message Passing Interface Forum (MPIF), taking into account the experiences from many different message passing systems. The two relevant documents are:

- *MPI: A Message-Passing Interface Standard (Version 1.1; June 12, 1995)*
- *MPI-2: Extensions to the Message-Passing Interface (July 18, 1997)*

These specifications, and much more information about MPI, can be obtained from the following MPI home pages:

<http://www.erc.msstate.edu/mpi>

<http://www.mpi-forum.org>

MPI Version 1.1 has the following content:

- **Point to Point Communication.** This includes the basic send and receive calls, in blocking and nonblocking mode. MPI datatypes can be used to describe the contents of messages, including user-defined datatypes which may be used to describe sub-arrays or other data structures.
- **Collective Communications.** This part contains procedures for barrier synchronization, broadcasts, gather/scatter operations, and reduction operations. MPI provides standard reductions like SUM or MAX, and user-defined reduction operations can be easily added.
- **Groups, Contexts, and Communicators.** These are used in MPI to describe groups of processes which are in a common communication realm. They provide the infrastructure to develop modular applications (in particular, parallel libraries) which use message passing internally, and guarantee that this communication does not interfere with other messages in the system.
- **Process Topologies.** Views of a set of processors can be created which suit the application needs. For example, if an algorithm works on a 3D grid, the programmer may prefer to reference the processors also as a 3D grid when sending messages between them. These MPI procedures help to map (physical) processors to (virtual) topologies.
- **MPI Environmental Management.** This section of the MPI standard has calls to initialize and finalize MPI, procedures to inquire about the environment (number of tasks, local task number), timer routines, and the MPI error handling infrastructure.
- **Profiling Interface.** MPI defines a simple name-shifting scheme which must be supported by all implementations. This allows the development and use of MPI tools without having access to the library's sources.
- **Language bindings.** Bindings exist for C and for FORTRAN77.

Note

MPI and Fortran: Unfortunately, the MPI FORTRAN77 bindings do not strictly conform to the ANSI/ISO FORTRAN77 standard. Most importantly, MPI uses one procedure to send user data of all possible datatypes (MPI calls the corresponding procedure argument a *choice* argument). This design is supported with C where data is passed as void*, but the Fortran standard does not allow this (most Fortran compilers do, though). MPI also makes some assumptions about compiler implementation details.

These issues might cause problems when using the MPI FORTRAN77 bindings from Fortran 90 or Fortran 95. Details can be found in MPI-2, and in /usr/lpp/ppe.poe/samples/mpif90/README.mpif90.

MPI Version 2 contains the following extensions to MPI Version 1.1:

- Corrections and Clarifications to MPI Version 1.1. This part of MPI-2 defines the MPI Version 1.2 language level.
- Miscellany. Several additions or modifications which do not fit into one of the other chapters of the MPI-2 document.
- Process Creation and Management. MPI Version 1 does not provide facilities for dynamic process management, like spawning additional processes. MPI-2 provides this functionality.
- One-Sided Communication. Traditional message passing requires a matching receive call on a remote process to send data to it. One-sided communication needs only one visible communication partner. The origin task makes an MPI call, which specifies both origin and target parameters. Target parameters are sent to an asynchronous agent or handler at the remote task and the target activities needed to carry it out. The operation occurs without needing user calls to MPI at the target.
- Extended Collective Operations. Extensions of existing collective operations for intracommunicators to support intercommunicators, some new functions, and support for in-place buffers.
- External Interfaces. Defined to facilitate the development of additions to the core MPI library.
- I/O. This is probably the single most important extension of MPI Version 1. The MPI datatype mechanism can be used to define how parallel processes view a file, and calls similar to the send and receive communication calls can be used to access a file in parallel. There are

three different modes of parallel I/O defined in MPI-2: explicit offset, shared file pointer, and local file pointers.

- Language Bindings. C++ bindings are provided, and some Fortran 90 issues are discussed. As previously mentioned, the MPI Fortran bindings are not completely (Fortran) standard-conforming, so it is difficult to provide a clean binding for Fortran 90 or Fortran 95.

Although the MPI-2 document was published about two years before the time of writing this redbook, there are very few (if any) full implementations of that standard available. This is quite different to MPI Version 1, for which a number of full implementations exist (in the public domain as well as commercial products).

16.2.2 IBM's Implementation of MPI

Parallel Environment 2.4 fully supports MPI Version 1.2. The threads-based version of the MPI library also supports a subset of the MPI-IO chapter of MPI-2. See 16.2.2.3, "Compatibility Issues" on page 470 for details on the signal-based and threads-based MPI libraries, and 16.2.3, "MPI-IO Subset" on page 474 for the MPI-IO subset supported by PE 2.4. IBM has committed to provide full MPI-IO support and expects to continue development of the remainder of the MPI-2 standard in future releases of Parallel Environment for AIX.

In most cases, the details of the IBM implementation of MPI should be transparent. Here we discuss some of the areas which are directly visible to the user.

16.2.2.1 Signal-based and Threads-based MPI

There is one important fact which influences the implementation of MPI: The MPI runtime system has to do a lot of processing in the background (that is, work which is not coupled to a specific MPI procedure invocation). Examples include the handling of nonblocking communication requests, as well as reacting to operating system notifications that messages have arrived through the network. However, the MPI library executes in user mode, not in kernel mode. This means that the MPI runtime system has to share CPU resources with the actual user application. Consequently, there has to be a means to switch between the application (doing calculations) and the MPI runtime (processing communication operations to and from the SP Switch or UDP/IP layers).

Using Signals

Early PE implementations of MPI relied on signals to switch between the application and the POE/MPI runtime. Specifically, there are three groups of signals which are used for different purposes:

- Asynchronous control of the individual parallel processes, especially their termination, is facilitated by POE through the use of the signals SIGQUIT, SIGHUP, SIGPWR, SIGDANGER, SIGTSTP, SIGTERM, SIGHUP and SIGINT.
- The SIGALRM and SIGIO signals are used to manage message traffic. The MPI library has to process communication requests of the application, or deal with data that arrives through the network. These two signals are used to pass control between the application code and the MPI runtime system.
- If User Space (US) communication over the SP Switch is performed, the SIGPIPE signal is used during processing of SP Switch faults. Its main purpose is to defer all MPI processing until the SP Switch has processed the fault and the switch is available again.

Note that with the signal-based MPI library, these signals are not available to the user application; they are used internally by POE/MPI.

This signal-based version of MPI is still provided with PE 2.4 for backward compatibility. It is contained in the libraries libppe.a and libmpi.a. The compiler scripts mpcc, mpCC, mpXlf and mpXlf90 described in 16.1.3, “The POE Compilation Scripts” on page 459 use these signal-based libraries.

Threads-based MPI

The more recent PE implementation of MPI uses threads instead of signals. Threads are normally more efficient than the interrupts which are caused by signals, and they also offer the potential of parallelism if the MPI task runs on an SMP node. The threads-based implementation resides in libppe_r.a and libmpi_r.a. The compiler scripts mpcc_r, mpCC_r, mpXlf_r and mpXlf90_r described in 16.1.3, “The POE Compilation Scripts” on page 459 use these threads-based libraries.

Figure 188 on page 468 shows the control flow in a task which uses the threads-based MPI implementation:

- The Partition Manager Daemon (PMD) uses fork() and exec() to start the POE task. This is the same as for the signal-based implementation.
- Before calling the user's main() program, the POE initialization routine will create a thread called the “POE Asynchronous Control Thread”. This

thread will handle all the asynchronous signals to terminate the process, for example SIGINT.

- The user application itself may create any number of “compute threads”, for example if it exploits thread parallelism by specifying the `-qsmp` compiler option.
- In any of the application's threads, `MPI_Init()` is called. Note that exactly one thread must call `MPI_Init()`, not all of them. This will spawn two “service threads”: the Timer Thread (which replaces the SIGALRM signal), and the Asynchronous I/O Thread (which replaces the SIGIO signal).
- All user threads in the process can then use MPI procedures.
- The `MPI_Finalize()` call must be made from the same thread which called `MPI_Init()`. This call will terminate all MPI service threads.
- No thread is allowed to call any MPI procedure after this time.
- Any threads created by the application must also be terminated by the application; this is not managed by `MPI_Finalize()`.

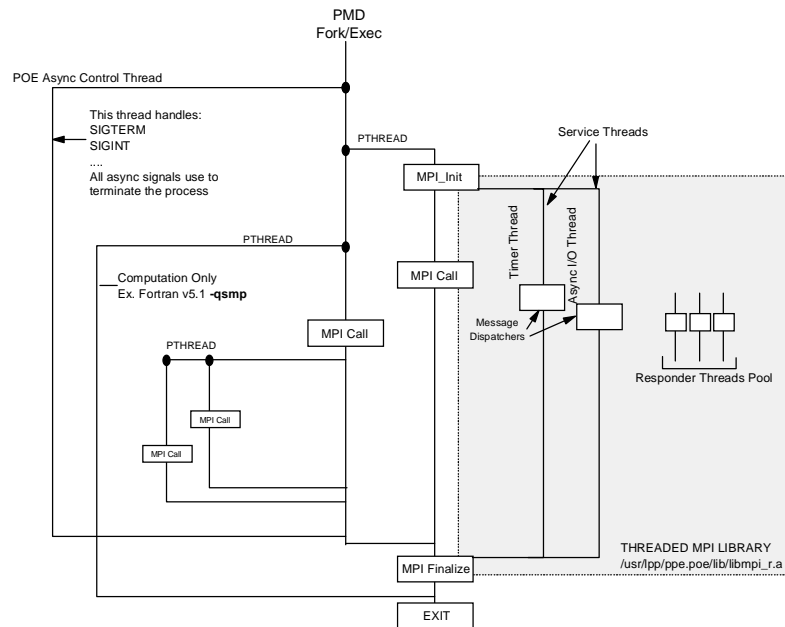


Figure 188. Thread Structure of a MPI-POE Task

The MPI library also creates “responder threads” that handle asynchronous MPI-IO and non-blocking communications. These threads are dynamically

created when needed by an MPI asynchronous operation, but they are not destroyed after the operation ends. This allows a fast reuse of responder threads by future operations. The `MPI_Finalize()` call will eventually deallocate these resources.

In contrast to the service threads, which are merely a more efficient implementation of the functionality which was already implemented with signals, the responder threads offer significant new capabilities: they can exploit parallelism to process asynchronous communication calls.

The name responder thread derives from the fact that a nonblocking MPCI `Irecv` can register a handler function. When a message arrives that matches that MPCI `Irecv`, MPCI responds by putting the registered handler onto the responder queue so it can be executed on a responder thread. Thus, the MPI task “responds” to the arriving message without any application-level participation.

16.2.2.2 The Underlying Communication Subsystem

The IBM implementation of MPI does not directly implement the MPI procedures for a specific networking device. To isolate the MPI library from lower level details, a layered model as shown in Figure 189 on page 470 is used:

- The message-oriented calls of MPI are mapped to an intermediate layer called the Message Passing Client Interface (MPCI). Note that, whereas MPI supports both point-to-point and collective operations, MPCI only supports point-to-point messages. So the mapping of collective communications to point-to-point messages takes place completely within the MPI library.
- MPCI transforms the message-oriented MPI calls to stream-oriented calls, and finally communicates with the device-dependent layer, which is packet oriented.
- On the device-dependent level, two different communication models are supported. Communication can be done over UDP/IP using UNIX sockets, or alternatively the User Space (US) communication protocol of the SP Switch can be used.

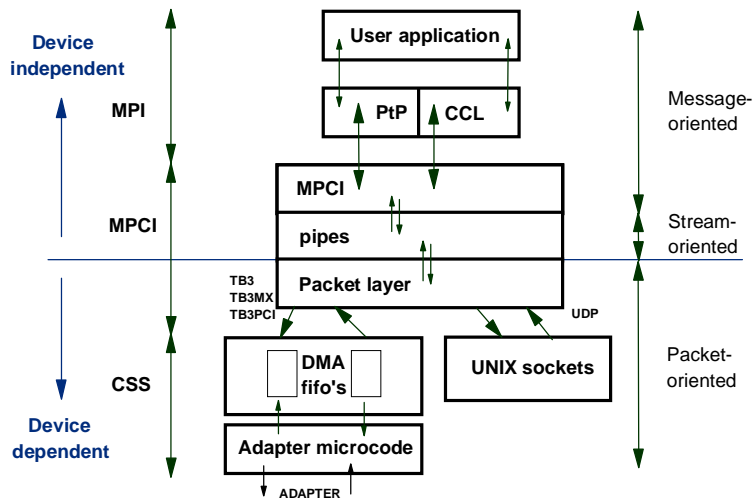


Figure 189. MPI Structure

The MPI library is not aware of the underlying communication subsystem; there is one MPI library regardless of the networking infrastructure. However, different MPCl libraries are available for each communication model, as follows:

```
# ls1pp -w "*libmpci*"
File                               Fileset                             Type
-----
/usr/lpp/ssp/css/libip/libmpci_r.a  ssp.css                             File
/usr/lpp/ssp/css/libtb3/libmpci_r.a ssp.css                             File
/usr/lpp/ssp/css/libip/libmpci.a    ssp.css                             File
/usr/lpp/ssp/css/libtb3/libmpci.a   ssp.css                             File
/usr/lpp/ppe.poe/lib/ip/libmpci.a   ppe.poe                             File
/usr/lpp/ppe.poe/lib/ip/libmpci_r.a ppe.poe                             File
```

The ppe.poe filesets deliver a UDP/IP version of MPCl so Parallel Environment can be used on non-SP workstations. The ssp.css filesets deliver versions for UDP/IP and for US over the SP Switch. The libraries which have the _r suffix are used with the threads-based implementation of MPI.

16.2.2.3 Compatibility Issues

Internals of both AIX and PSSP/PE regularly change from release to release. Normally, binary compatibility is kept between such changes and old applications should run with new software levels without modification.

Nevertheless, there are always some exceptions, and we recommend that you recompile and relink your applications whenever a significant system

upgrade has been performed (for example, moving from AIX 4.2 to AIX 4.3 and from PE 2.3 to PE 2.4).

Since recompilation of the entire application might not be possible or desirable in all cases, this section summarizes two important compatibility issues to which you should pay attention if you use the threaded MPI library.

POSIX Threads Version

Threads for AIX 4.3.0 and earlier are based on Draft 7 of the IEEE POSIX Thread Standard. AIX 4.3.1 and later is based on the IEEE POSIX 1003.1-1996 Thread Standard, which is the final standard, and different from Draft 7.

This means that the AIX threads library (libpthreads.a) has changed in AIX 4.3.1 and later, but AIX maintains binary compatibility across versions. AIX does that by having multiple shared objects in the library itself. Executables built in AIX 4.3.0 or earlier (which link the threads library) reference the shr.o shared objects. Executables compiled in AIX 4.3.1 and later reference shr_xpg5.o shared objects. However, AIX 4.3.1 and later maintains both shared objects in the library, so “old” applications referencing shr.o objects will run without problems. This is shown in Figure 190.

The threaded MPI library in Parallel Environment 2.4 is compiled in AIX 4.3.1, which means it uses the shr_xpg5.o shared objects. Earlier versions of the threaded MPI library reference shr.o objects, since they have been built with AIX 4.2.1 or earlier. Applications compiled with previous versions of AIX and Parallel Environment will run, as long as mutexes (locks) and thread condition structures (signaling structures) are not shared.

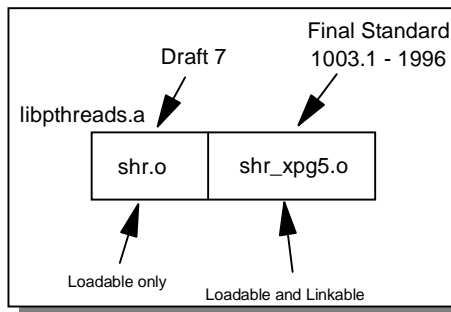


Figure 190. Structure of AIX Thread Library

The interesting case comes when part of an old application has to be recompiled under AIX 4.3.1 and later. In that case, part of the application will

use shr.o shared objects, and part will use shr_xpg5.o shared objects. This is not guaranteed to run, mainly because it is likely that these recompiled parts of the application will share locks or signaling structures with “old” parts; this is not supported across these shared objects (shr.o and shr_xpg5.o).

In order to safely recompile part of an “old” application, Parallel Environment provides a flag (-d7) in the mpcc_r compiler script to link the libthreads_compat.a library, which contains only the shr.o shared objects.

16.2.2.4 AIX Thread Structure

AIX 4.3.1 and later has changed the default thread structure. In AIX versions prior to AIX 4.3.1, the mapping between user threads (pthreads) and kernel threads (kthreads) is 1:1 (also called *System Contention Scope*). This means that each user thread has a kernel thread allocated to it. The AIX Kernel Dispatcher (sometime called Scheduler) allocates a kernel thread each time a user or process thread is created.

In AIX 4.3.1 and later, this thread structure has changed. The default is now M:N or *Process Contention Scope*, which means that the Kernel Dispatcher has a “pool” of kernel threads which are dynamically allocated to user or process threads when a user scheduling thread switches from one user thread to another, or when a user thread makes blocking system calls.

This new default in AIX brings some problems to multi-threaded parallel applications, because the kernel threads are allocated to process threads only when they are available, which does not guarantee that your application will be able to execute all the threads you (or the MPI runtime library) may want. However, in System Contention Scope, your threads have each allocated a kernel thread no matter what they are doing.

You can change this default to System Contention Scope by setting an environment variable called AIXTHREAD_SCOPE to S. For applications compiled with Parallel Environment 2.4, this variable is not required because the MPI library changes each process scope thread which makes MPI calls into system scope. However, it will not change the scope of a thread which does not call MPI. Applications built with previous versions of Parallel Environment under AIX 4.3.1 or later need to set this variable to S (System Contention Scope), since the previous versions of the MPI library do not change this.

16.2.2.5 Performance Tuning

Similar to POE, the behavior of the MPI runtime and its cooperation with the communication subsystem can be controlled through environment variables. These environment variables are described in Table 13 of *Parallel*

Environment for AIX 2.4: Operation and Use, Vol. 1, GC23-3894. The following options have proved to be particularly useful for a broad range of applications:

MP_EAGER_LIMIT — This is a threshold for MPI message sizes. It has a default value of 4 KB and a maximum of 64 KB. **MP_EAGER_LIMIT** specifies the maximum message size for which the MPI runtime can use a faster internal protocol, because it is expected that the partner of the communication has enough buffer space to receive the messages. Messages above that threshold have to be transmitted using a slower protocol. Consult the Parallel Environment documentation for the interrelation of the number of MPI tasks, **MP_EAGER_LIMIT** and **MP_BUFFER_MEM**.

MP_BUFFER_MEM — This can be used to set the size of MPI internal working space. For US communication, this value defaults to its maximum (64 MB), but for IP communication, the default is slightly less than 3 MB. You might want to increase that value if you are sending large messages over IP, or are developing an application in IP mode which should later run with US.

MP_CSS_INTERRUPT — Setting this variable to yes (and maybe adjusting **MP_INTRDELAY**) may improve the performance for applications which perform asynchronous I/O and have the potential of overlapping communication and computation. With the default setting of **MP_CSS_INTERRUPT=no**, no SIGIO interrupt will be generated by the runtime if new data has arrived asynchronously and needs to be processed. This implies that the actual processing may be done as a side effect of other MPI calls, or it may be done on the timer threads, or it may not be done until the matching **MPI_Wait()** call, which causes unnecessary delays. Enabling **MP_CSS_INTERRUPT** causes additional overhead to process the SIGIO interrupts, but may improve the overall performance because it can significantly reduce the time spent in **MPI_Wait()** calls.

MP_SINGLE_THREAD — Set this variable to yes to avoid some overhead in a program which is compiled with the threaded MPI library, but is known to be single threaded.

Note: that programs using MPI-IO may not set this to yes.

In any case, you need to experiment with these settings to find out if and how your application will benefit from changing the default values.

Apart from these environment variables, the behavior controlled by **MP_INTRDELAY** and **MP_CSS_INTERRUPT** can also be queried and set from the application by using a set of IBM proprietary procedure calls. This might help you to improve performance in cases where the communication

patterns change during the execution of the program, so that a global setting is not suitable. However, we recommend that you do not use these non-standard procedure calls unless significant performance improvements can be achieved.

16.2.3 MPI-IO Subset

The Message Passing Interface (MPI) standard provides an efficient way to communicate and synchronize multiple tasks. However, MPI Version 1 does not provide support for parallel file I/O. Although applications may use a parallel file system to achieve this, the portability of these applications to other platforms and other file systems is not guaranteed.

Chapter 9 of the MPI-2 standard defines the set of MPI calls that allow parallel file I/O. This set of calls is called MPI-IO and a portion is being implemented as part of the threaded MPI library within Parallel Environment 2.4.

The MPI-IO implementation of the MPI-2 standard in Parallel Environment has been divided in two parts. Parallel Environment 2.4 includes most of the MPI-IO functions although not all. The decision as to which part of the subset needed to be implemented first was heavily based on customer requirements. For detailed information about the current implementation of the MPI-IO subset in Parallel Environment 2.4, refer to *IBM Parallel Environment for AIX: MPI Programming and Subroutine Reference*, GC23-3894.

The MPI-IO subset provides great flexibility for applications to define how they will do their I/O. Tasks within an application can use MPI predefined and derived datatypes to partition the single file in multiple views. This allows applications to partition the data and create their own access patterns based on these basic blocks or datatypes.

PE Version 2.4 supports:

- Basic file manipulation (open, close, delete, sync)
- Get/set of file attributes (MPI view, size, group, mode, info)
- Blocking data access with explicit offsets, both independent and collective
- Nonblocking data access with explicit offsets (independent only)
- Derived datatypes for memory and file mapping

This is an initial release of only a subset of MPI-IO. Emphasis in this release is not on performance, but instead focuses on robustness and correctness. This release has no support for file hints which might improve performance. It

only uses standard POSIX calls. Support for MPI-IO will be enhanced in future PE releases.

In PE 2.4, MPI-IO is fully supported only for GPFS. Other file systems may be used only when all tasks of the MPI job are on a single node or workstation. (JFS cannot make a single file visible to multiple nodes. NFS, DFS and AFS can make the same file visible to multiple nodes, but do not provide sufficient file consistency guarantees when multiple nodes access a single file.)

16.2.4 Parallel Virtual Machine

Parallel Virtual Machine (PVM) is a popular message passing system which was developed by Oak Ridge National Laboratory (ORNL) to use workstation clusters for parallel computing. It predates the MPI standard, but is still actively used.

Since the release of Parallel Environment Version 2.3, IBM no longer provides its own implementation of PVM. We recommend that you use MPI as the industry standard API for explicit message passing. If PVM has to be used on the SP (for example to port existing PVM applications), the public domain implementation of PVM can be used. PVM Version 3.3 is widely used, and Version 3.4 is the current release as of this writing. There is additional information about this topic at:

http://www.epm.ornl.gov/pvm/pvm_home.html

There are two implementations which run on the SP: an RS6K version and an SP2MPI version. Only the SP2MPI implementation supports User Space communication over the SP Switch, as it is layered on top of IBM's MPI. The disadvantage of this PVM implementation is that it requires one extra processor for each invocation of `pvm_spawn/pvmfspawn` and tasks created by different spawn calls do not intercommunicate via User Space protocol. Therefore, care must be taken when this PVM implementation is used for applications which call `pvm_spawn/pvmfspawn`. The RS6K implementation does not have this drawback, but can only use IP communication.

16.2.5 Low-Level API (LAPI)

The LAPI library provides uni-lateral communication on the SP Switch in User Space mode. One process initiates a LAPI operation and no complementary action on any other process is required. LAPI provides the `LAPI_Put()` and `LAPI_Get()` functions, as well as a general *active message* function `LAPI_Amsend()`, which allows programmers to register their own handlers.

LAPI is an IBM proprietary calling interface, and only available for the SP switch in User Space mode. Although broadly similar to the one-sided communications of the MPI-2 standard, the semantics and implementation of LAPI are very different from MPI-2 one-sided communication. In addition, LAPI does not support the versatile data layout of MPI datatypes.

Note

MP_MSG_API: If you want to use LAPI, the POE environment variable MP_MSG_API must be set to include LAPI, since LAPI requires a separate DMA window on the SP Switch adapter, different than the adapter window used by MPI.

Additional information can be found in *PSSP for AIX: Administration Guide*, SA22-7348, and in the Chapter 10, "Overview of LAPI", of the redbook *Technical Presentation for PSSP Version 2.3*, SG24-2080.

16.2.6 Message Passing Library (MPL)

Message Passing Library (MPL) is IBM's proprietary application programming interface for explicit message passing, which was developed before the MPI standard. Its calling interface is still provided for backwards compatibility, but we strongly recommend that you use MPI for all new application developments. You should also be aware that MPL is not thread-safe, so the MPL calls are only available in the signal-based version of the MPI library.

MPL and MPI could be used in the same program, but both partners of a communication must always be handled by calls to the same library. For example, an MPI_Send() call cannot be received by an MPL mpc_rcv() call.

A concise mapping of MPL calls to corresponding MPI calls can be found in Chapter 5, "Babel Fish" of the *Parallel Environment for AIX: Hitchhiker's Guide*, GC23-3895.

16.3 High Performance Fortran

Although explicit message passing is the most flexible tool to develop parallel applications on a MIMD parallel computer, this programming model is a long step away from the conventional serial programming languages. It requires the whole serial program be recoded into a message passing program to exploit parallelism. For a limited class of problems, the data-parallel programming model offers an alternative: parallelism can be exploited by performing the same (or similar) operations in parallel on large, regular data

sets. Data-parallel programming languages are normally based on a sequential programming language, and add directives and new constructs to exploit this parallelism. Several dialects exist for both Fortran and C/C++, but only High Performance Fortran (HPF) is widely implemented and has become a de facto standard. HPF compilers are available for the SP as a separate IBM Licensed Program Product, and also from several independent software vendors.

16.3.1 What is in HPF

High Performance Fortran Version 1 is an extension of Fortran 90 to support the data-parallel programming model on MIMD and SIMD parallel computers using Fortran. HPF retains the usual programming paradigm: a parallel HPF program like a sequential program sees a global address space, and the control flow through the program is (conceptually) identical to that of a normal Fortran program. The most important extensions are:

- Data distribution directives: The distribution of data on the available processors can be explicitly specified, in order to take the different access speeds for data in local and remote memory into account. For example, arrays are aligned to each other, and distributed to a set of (logical) processors, by the ALIGN, DISTRIBUTE and PROCESSORS directives. These directives are comments to a serial Fortran compiler, and also do not change the semantics of the program when interpreted by an HPF compiler.
- New parallel constructs: The FORALL construct, the INDEPENDENT directive on DO loops and FORALLs, new INTRINSICs and an HPF_LIBRARY module extend the possibilities of Fortran 90 to express parallelism. Most of these features are language extensions which will not be understood by a serial Fortran compiler.
- An interface to other programming models: The EXTRINSIC mechanism can be used to define an interface for routines not written in HPF (not in Fortran and/or not data-parallel), which can then be used like ordinary HPF procedures. For example, computational kernels could be written using MPI and then integrated into an HPF main program. To mention one important application of this mechanism: IBM provides HPF interfaces to the IBM Parallel ESSL subroutine library, which is originally coded as a message passing library.

With this additional information, low-level work like the transformation of global to local indices, managing the communication among processors (by message passing, shared memory mechanisms, and so forth) and optimization of communication patterns in the program can be left to the HPF

compiler. Compared to explicit message passing, this significantly speeds up program development: parallelism is both expressed at a higher level, and can be introduced gradually into existing, serial programs. However, achieving good performance depends heavily on the available compiler technology, which is still evolving.

HPF Version 1 defines a subset HPF language, which requires neither full Fortran 90 support nor all the HPF features. Although many HPF compilers now support the full Fortran 90 language, very few support all of the HPF Version 1 features.

The HPF Version 2 language specification is meant to replace the Version 1 specifications, not to add to them (like in MPI, where Version 2 is an addition to Version 1). HPF Version 2 differs from HPF Version 1.1 in the following ways:

- The base language is Fortran 95, not Fortran 90. This means that the FORALL construct has been removed, as it is included in Fortran 95.
- There is no longer a subset HPF. The standard has been reorganized into a core language (similar contents as the previous subset), and a number of approved extensions. These contain features which are not widely used, difficult to implement, or otherwise unlikely to appear in many HPF compilers.
- The extensions which are defined within HPF Version 2.0 include features for more flexible data mapping, data and task parallelism, asynchronous I/O, as well as enhanced functionality of some intrinsic and library procedures and the HPF EXTRINSIC features.

More details can be found in the HPF Language Specifications:

- *HPF Language Specification Version 2.0* (January 31, 1997)
- *HPF Language Specification Version 1.1* (November 10, 1994)

These documents, as well as more information on the HPF language, compilers, applications and tools can be found on the home page of the High Performance Fortran Forum (HPFF) at:

<http://www.crpc.rice.edu/HPFF/>

16.3.2 IBM XL High Performance Fortran

IBM's product to support HPF is the XL HPF Compiler for AIX, program number 5765-613. XL HPF Version 1.4 supports the HPF Version 1.1 subset, full Fortran 90, and most of the full HPF Version 1.1 language features.

Since XL HPF Version 1.4 is based on the XL Fortran Version 6.1 compiler, program number 5765-D78, it also contains the functionality of that compiler. This includes full Fortran 95 and partial OpenMP support, exploitation of the IBM POWER, POWER2, POWER3, and PowerPC architectures (including SMP nodes), and support for 64-bit applications. Be aware that in XL HPF, some of these features may only be available in non-HPF mode, and that neither PSSP 3.1 nor Parallel Environment 2.4 support 64-bit applications.

For more information about XL HPF, check out the following sources:

- <http://www.software.ibm.com/ad/fortran/xlhpf/>
- *IBM XL High Performance Fortran Language Reference and User's Guide*, SC09-2631
- <http://www.software.ibm.com/ad/fortran/xlfortran/>
- *IBM XL Fortran Language Reference*, SC09-2718
- *IBM XL Fortran User's Guide*, SC09-2719

XL HPF requires PSSP and Parallel Environment for AIX as prerequisites, because it generates intermediate serial Fortran code with embedded MPI message passing calls. This is then linked to an executable which contains the HPF, MPI and POE runtimes. The POE infrastructure is used to actually run the HPF program. To POE, there is no difference between an explicit message passing program which has been built using the Parallel Environment's message passing libraries and compilation scripts, and a data-parallel HPF program which has been built using the `xlhpf` command.

XL HPF is integrated with the Visualization Tool (VT) described in 16.4.2.1, "IBM Visualization Tool (VT)" on page 488. This enables source level tracing and profiling of the original HPF program. There is no fully functional HPF source level debugger available for XL HPF.

16.3.3 Portland Group pgHPF

An important independent software vendor's HPF product is pgHPF from the Portland Group, Inc. (PGI). PGI's home page is:

<http://www.pgroup.com>

The European distributor for pgHPF is Pallas GmbH.

<http://www.pallas.de>

The pgHPF compiler is available for many different platforms, including a version for RS/6000 workstation clusters which uses IP communication and a version for the SP which can also exploit User Space (US) communication

over the SP Switch. Using US communication requires Parallel Environment for AIX, including IBM's implementation of MPI.

The `pghpfc` compiler actually uses the machine's native Fortran compiler (XL Fortran on AIX platforms), and supplements it with PGI's own libraries and runtime environment. The compiler produces intermediate serial Fortran code with embedded message passing calls. These calls can be layered on top of various message passing libraries. PGI provides RPM by default, which is a stripped-down version of PVM. On the SP, we recommend that you use IBM's implementation of MPI to achieve the highest performance. After building the executable, running it on the SP is comparable to running a plain MPI program.

In addition to the `pghpfc` compiler, PGI provides an HPF profiler `pgprof` which supports line-level profiling at the HPF source level. For debugging, the TotalView parallel debugger can be used. TotalView is described in 16.4.1.2, "The TotalView Debugger" on page 484.

16.4 Parallel Programming Tools

Compilers, message passing libraries, and runtime support are the core components which are required to build and run parallel application programs. However, for the development of parallel applications the programmer also needs other tools. Most importantly, parallel debuggers for all the available languages are necessary to allow the programmer to localize programming errors (including message passing errors). While debuggers help to make parallel programs correct, tuning the applications so that they achieve adequate performance requires tools to analyze the runtime characteristics. Serial profiling tools can (and should) be used to tune the code which runs on a single node. But to understand and tune the parallel performance of an application, parallel profiling and tracing must be done. This section summarizes the parallel tools which are available on the SP, either as IBM products or from independent software vendors.

More information about ongoing efforts in the parallel computing community to build parallel programming tools can be found on the home page of the Parallel Tools Consortium:

<http://www.ptools.org/>

16.4.1 Parallel Debuggers

In addition to the serial debuggers available in AIX, IBM provides two parallel debuggers as part of the Parallel Environment for AIX: the text-based `pdbx`

debugger and a debugger with a graphical user interface called `pedb` (formerly known as `xpdbx`). One of their main limitations is the fact that they do not support all the C++ and Fortran 90/95 language features which are supported by the IBM compilers. On the other hand, they offer the possibility of attaching to an already running program, which can be very convenient if problems appear in late stages of a long-running program.

As an alternative to these IBM parallel debuggers, the TotalView Multiprocess Debugger from Etnus, Inc. can be used. Its usage is more intuitive, it has much more complete language support, provides superior functionality, and is available on several platforms. For sites with significant parallel programming activities, we recommend that you install and use the TotalView debugger, which is also the debugger selected by the US Department of Energy (DOE) for the ASCI blue project.

16.4.1.1 IBM Parallel Debuggers `pdbx` and `pedb`

The `pdbx` debugger is a text-based POE application, which runs on the POE home node and controls the execution of the standard AIX debugger `dbx` on each of the remote nodes. It allows grouping of tasks, making it easier to send the normal `dbx` commands to all or a user-selected set of tasks of the application.

Since `pdbx` is a source-level debugger, applications should be compiled with the `-g` option so that sufficient debugging information is available in the executable. In addition, the program's source files must be available on all nodes, not only the executable. If the source files and executables reside in different directories, the `-qfullpath` option should also be used.

The `pdbx` debugger can operate in two modes: *normal mode*, which starts the application under control of the debugger; and *attach mode*, which permits the attachment of the debugger to an already running POE application. Attach mode is particularly useful for long-running applications which are suspected of hanging in a late stage. If such applications had to be run completely under control of a debugger, it would probably take too much time to reach the interesting state in the calculation. With `pdbx`, it is possible to attach to the running program at any time.

To debug a parallel program in normal mode, you basically replace the `poe` command in your program invocation with the `pdbx` command. The debugger accepts most of the POE environment variables and command-line arguments. For example:

```
pdbx my_prog [prog_options] [poe_options] -l src_dir1 -l src_dir2
```

The `-l` option (lowercase L) specifies a directory where program source files reside. If multiple directories are needed, multiple `-l` options must be specified. After the compute nodes have been selected (by the same mechanisms as when calling `poe`), `pdbx` starts the `dbx` debugger on each of the compute nodes. On all the nodes, the application's executable is then run under control of the local `dbx` debugger, which is controlled by `pdbx` on the home node.

To attach to an already running POE application, `pdbx` must be invoked with the `-a` option, on the same home node as the POE application to which it should be attached. The application is specified to `pdbx` by the AIX process ID of the `poe` command running on the home node. For example:

```
$ poe my_prog [prog_options] [poe_options] & # start application
[1]      31186
$ sleep 300 # run it for a while
$ pdbx -a 31186 -l src_dir1 -l src_dir2 # attach to application
```

Since the POE environment is already set up when the application has been started, it is normally not necessary to specify any program or POE options to `pdbx` when starting it in attach mode.

The `pedb` debugger, formerly known as `xpdbx`, provides a Motif-based graphical user interface. Like `pdbx`, it executes on the home node and can be run in normal mode, or in attach mode by specifying the `-a` option. In addition to the options accepted by `pdbx`, the `pedb` debugger also accepts X windows options to customize its GUI. As for `pdbx`, the application should be compiled and linked with the `-g` option, and the source files must be available on all the nodes. An example of a source file view in `pedb` is shown in Figure 191 on page 483.

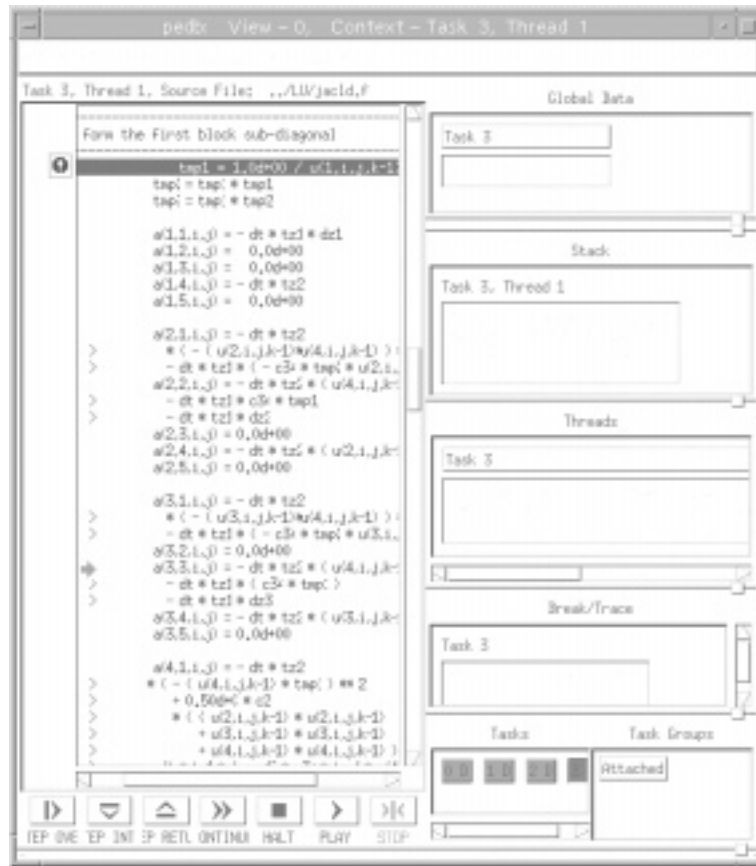


Figure 191. Source File View with pedb

The graphical user interface of `pedb` makes it more convenient to display data structures in the application. However, there are some limitations concerning the size of the data sets that can be displayed. The `pedb` debugger also provides a message queue debugger to inspect the MPI messages queued on the nodes.

Note that `pedb` requires the selection of a fixed set of tasks during startup. This cannot be changed later without detaching from the application and attaching again with a new selection of tasks to be debugged. The tasks that are not selected will not run under debugger control. The maximum number of tasks supported within `pedb` is 32. For larger applications, only a subset of at most 32 processes can be debugged.

Note

Initial breakpoint: Both `pdbx` and `pedb` allow you to set a user-defined initial breakpoint (the place in the program where control is passed to the user for the first time): The POE environment variable `MP_DEBUG_INITIAL_STOP` can be used to set an alternative file name and source line. By default, the debuggers stop at the first executable statement in the main program.

More details on `pdbx` and `pedb` can be found in *IBM Parallel Environment for AIX, Operation and Use, Volume 2, Part 1: Debugging and Visualizing*, SC28-1980.

16.4.1.2 The TotalView Debugger

TotalView is a multiprocess, multithread debugger for applications written in C, C++, FORTRAN 77, Fortran 90, and PGI HPF. It supports multiple parallel programming paradigms including MPI, PVM, and OpenMP. On the SP, TotalView Version 3.8 provides message queue debugging for IBM's implementation of MPI.

The TotalView debugger was formerly available from Dolphin Interconnect Solutions (see <http://www.dolphinics.com/>). It is now marketed by Etnus, Inc. (see <http://www.etnus.com/>). The European distributor for TotalView is Pallas GmbH (see <http://www.pallas.de/>). The Etnus and Pallas home pages also include an on-line presentation of many of the debugger's features.

To use TotalView with an MPI application, the application must be compiled with the `-g` and `-qfullpath` options, and must be linked with the `-bnoobjreorder` option. For example:

```
mpxlf -g -qfullpath -c myprog.f mysub.f
mpxlf -g -bnoobjreorder myprog.o mysub.o -o myprog
```

The program is then started under TotalView control, for example:

```
totalview -no_stop_all poe -a myprog -procs 4
```

All TotalView options must be specified after the `totalview` and before the `poe` command. The `-no_stop_all` option specifies that only the task which reaches a breakpoint stops at that breakpoint, by default all other tasks would stop, too. Also note that the `-a` option must be specified between `poe` and all its arguments.

TotalView then displays its root window, and a *process window* as shown in Figure 192 on page 485. Initially, both windows only show the `poe` process on the home node.



Figure 192. TotalView Process Window: poe

Type `g` (lower case g for go) in the poe process window to start the individual tasks of the parallel application. The screen displays progress information for the parallel tasks, and a pop-up which allows you to stop all the tasks before they enter the application's main program. Then a separate process window for the main application's tasks opens, as shown in Figure 193 on page 486. Note that the window title shows the application program's name (heatd2) and the task number (one). The process window for poe can then be closed.



Figure 193. TotalView Process Window: Application Program

The three main areas of the process window are:

- Stack Trace: Displays the current calling hierarchy
- Stack Frame: Displays names and values of (local and global) variables
- Source Code Area: Displays the actual program source

After setting breakpoints, the program can be continued by typing **G** (uppercase G for go group). The root window then displays the status of all tasks, as shown in Figure 194 on page 487. Here **R** means running, **T** means stopped by TotalView, and **B** with a number indicates the breakpoint at which the task is stopped.

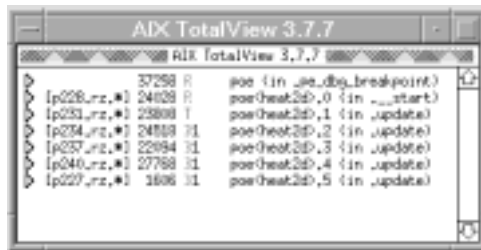


Figure 194. TotalView Root Window: Process Status

Normally, TotalView is controlled using the mouse: the left mouse button selects objects, the middle button displays the currently available functions/commands, and the right button "dives" into the selected object. The right mouse button can be used to display a subroutine's source text, values of variables, or attributes of breakpoints.

To navigate between different tasks, select them in the root window, or use the navigation buttons in the upper right corner of the process window. In general, commands in lower case only apply to one task, whereas upper case commands apply to all tasks of a parallel application.

Note

MP_PULSE: To avoid timeouts during debugging, we recommend that you set the MP_PULSE environment variable to zero before starting TotalView. Otherwise, POE might cancel the application if a task is stopped and does not respond to POE's heartbeat messages.

To use TotalView to debug a serial program, call `totalview` with the name of the executable as argument. For PVM programs, the directory where the TotalView executables reside must be included in the `ep=` clause in the hostfile, and `totalview` must be invoked with the `-pvm` option after the `pvm` daemon has been started. Debugging PGI HPF programs requires that the program is compiled and linked with the `-g -qfullpath -Mtv -Mmpi` options, and run with the `-tv` runtime option which starts off TotalView.

16.4.2 Profiling and Tracing

For performance analysis and optimization, the programmer needs graphical tools that display the performance information obtained by sample runs of a parallel application. The amount of tracing information (as well as the trace files used to store it) can be overwhelmingly large. The successful

deployment of performance analysis tools depends on presenting this tracing information in an efficient way, and allowing the programmer to quickly navigate and focus on the relevant parts.

Among the tools to do this are IBM's VT and Xprofiler, which are part of the Parallel Environment for AIX. In addition, the Vampir tool from Pallas GmbH is introduced. Vampir has a focus similar to the trace visualization part of VT, but is easier to use and often presents information in a more comprehensive way than VT.

16.4.2.1 IBM Visualization Tool (VT)

Visualization Tool (VT) is IBM's graphical tool for trace visualization of application program execution. Set the POE environment variable `MP_TRACELEVEL` to a value between 0 (no tracing) and 9 (maximum tracing), to instruct Parallel Environment to generate trace records of a POE application while it is running. This trace data can be analyzed by the `vt` command afterwards. A large number of display options are available, including computation and MPI communication analysis and general system utilization information.

In a second mode of operation, VT is also used for performance monitoring of a set of SP nodes without accessing a particular application's trace files.

More details on VT can be found in *IBM Parallel Environment for AIX, Operation and Use, Volume 2, Part 1: Debugging and Visualizing*, SC28-1980.

16.4.2.2 IBM Xprofiler

AIX supports profiling of CPU usage for serial programs through the standard UNIX `prof` and `gprof` profiling commands. Both `prof` and `gprof` support subprogram level profiling and subprogram call counts. They require that the application be compiled and linked with the `-g` and `-pg` option, respectively. The `gprof` command also provides call graph information.

Parallel Environment takes steps to ensure that each task of a parallel job writes its profile data to a separate file. More details can be found in *IBM Parallel Environment for AIX: Operation and Use, Volume 2*, SC28-1980.

Note

tprof: AIX also provides a `tprof` profiling command. The `tprof` command supports subprogram level profiling without the need to recompile/relink with special options, and line-level profiling for applications which have been compiled and linked with the `-g` option. However, `tprof` provides neither subprogram call counts nor call graph information. There are also some security concerns about using `tprof`: it can be used to profile the AIX kernel, and to do so requires root privileges. If `tprof` is made available to normal users by using SUID root, some security exposures might occur.

The Parallel Environment's Xprofiler is a serial application with a graphical user interface, which provides user-friendly access to `gprof` profiling information for both serial and parallel applications. Like `gprof`, Xprofiler provides only CPU usage information and requires that the application is compiled and linked with the `-pg` option. It is recommended that the `-g` option is also used, since some of the Xprofiler functions also require this flag. Xprofiler is started as follows:

```
xprofiler a.out gmon.out1 gmon.out2 ... gmon.outN [xprof_options]
```

Note that Xprofiler accepts wildcards, so the previous line could be rewritten as follows:

```
xprofiler a.out gmon.out* [xprof_options]
```

As can be seen in this example, for a parallel application all profiling output files must be specified, in addition to the (SPMD) executable.

Note

POE profiled libraries: The AIX operating system and compilers provide a version of their libraries which has been built with the `-pg` compiler options. Typically these are located in `/usr/lib/profiled/`. When Parallel Environment for AIX is installed, it rebuilds `libc.a` and stores it in `/usr/lpp/ppe.poe/lib/`. To support profiling of this modified `libc.a`, a copy of that library is also built with the `-pg` profiling option, and stored in `/usr/lpp/ppe.poe/lib/profiled/`. Be sure to include this library in your applications if you want to profile system routines. This can be achieved by setting `MP_EUILIBPATH`, for example:

```
export MP_EUILIBPATH=/usr/lpp/ppe.poe/lib/profiled:/usr/lib/profiled:/lib/profiled  
:/usr/lpp/ppe.poe/lib
```

The Xprofiler can provide the same table reports as `gprof`. In addition, it visualizes the subprogram call tree of the application. Figure 195 on page 490 shows an example of this. Subprograms are represented by boxes whose width and height provide details on the procedure. For example, in Xprofiler's *summary mode*, the height of a subprogram box represents the CPU time spent in itself, while the width also includes the CPU time of its descendants. In *average mode*, the height represents the CPU time in that subprogram averaged over all processes (for which `gmon.out` files are loaded), while the width gives the standard deviation of that average.

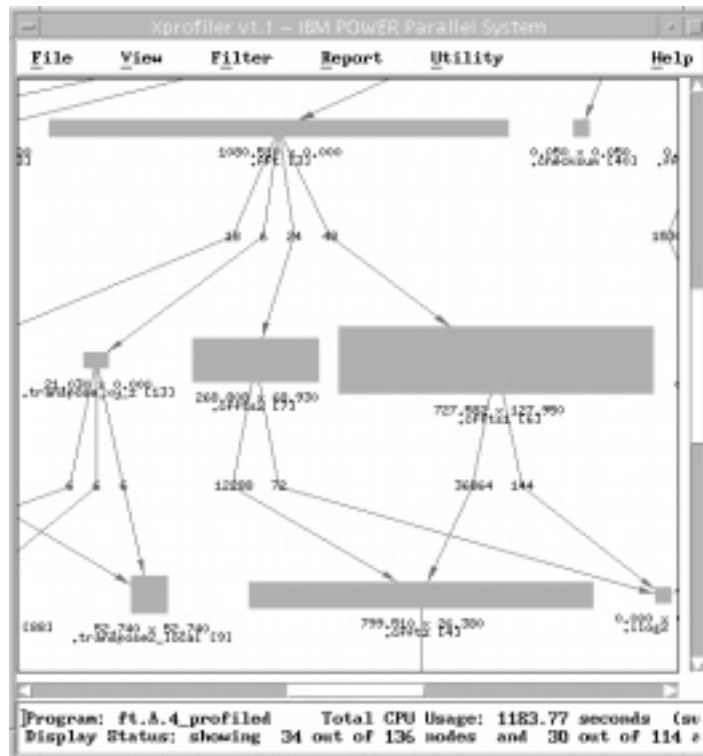


Figure 195. Xprofiler Main Window with Subprogram Call Tree

With Xprofiler, you can easily zoom in and out of interesting parts of the code.

Another important feature is the concept of clustering, which for example allows you to bundle all the subprograms of a library into one box. By default, all the subprograms from a shared library are clustered (or collapsed) into one box. They can be unclustered (or expanded) if details about individual subprograms within that library are required.

More details on Xprofiler can be found in *IBM Parallel Environment for AIX, Operation and Use, Volume 2, Part 2: Profiling*, SC28-1980.

16.4.2.3 Pallas Vampir and VampirTrace

Vampir (Visualization and Analysis of MPI pRograms) is an MPI performance analysis tool. It was initially developed in the German research center Forschungszentrum Juelich (FZJ) and is available as a commercial product from Pallas GmbH:

<http://www.pallas.de>

The tool consists of two parts:

- VampirTrace is a library that collects runtime traces of MPI applications.
- Vampir itself is a graphical visualization tool which can be used to analyze these runtime traces after the program execution.

To instrument an MPI program, it is necessary to link the application with the VampirTrace library. For example, if the library libVT.a resides in /usr/tools/lib/, use the command:

```
mpxlf myprog.f mysub.f -L/usr/tools/lib -lVT -o myprog
```

After setting some environment variables (license manager, configuration file location, and so on) the application can be executed in the usual way, for example by calling:

```
poe myprog -procs 8
```

The VampirTrace library uses the MPI profiling interface to attach to the application, and writes a trace file when the application calls MPI_Finalize. Details can be controlled through a configuration file, or by inserting subroutine calls into the application. For the above example, the default name of the trace file would be myprog.bpv.

The visualization tool Vampir is used to analyze the resulting trace files. It is started by the `vampir` command. Vampir is very easy to use, and has a fast and versatile graphical user interface which allows you to zoom into arbitrary parts of a trace. A variety of graphical displays presents important aspects of the application runtime behavior. Three important classes of displays are:

- Timeline views display application activities and message-passing along the time axis. Figure 196 on page 492 shows an example, zoomed to a very detailed level.

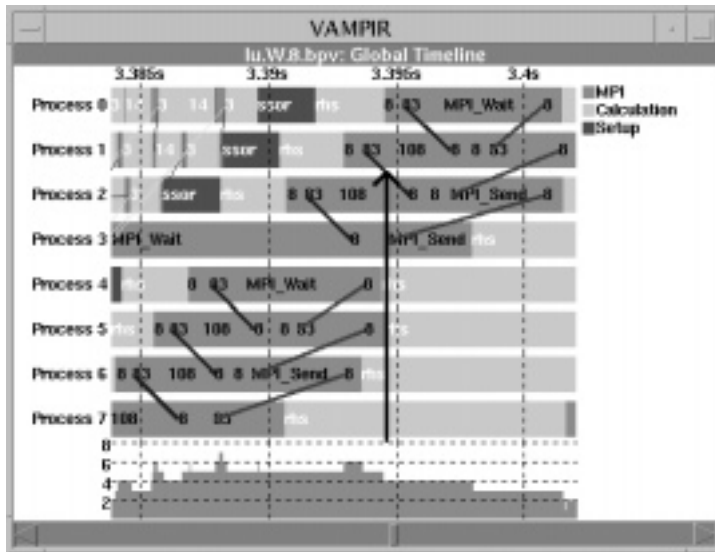


Figure 196. Vampir Global Timeline View

Individual messages can be identified by clicking on them, Figure 197 shows the pop-up window which identifies the message marked by an arrow in Figure 196.

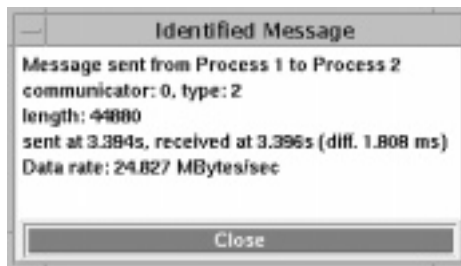


Figure 197. Vampir Message Identification

- Communication statistics present communication metrics for an arbitrary time interval. Figure 198 on page 493 shows a matrix with average message rates between all processes. The striped pattern is typical for nearest-neighbor communication, here for the NAS LU benchmark.

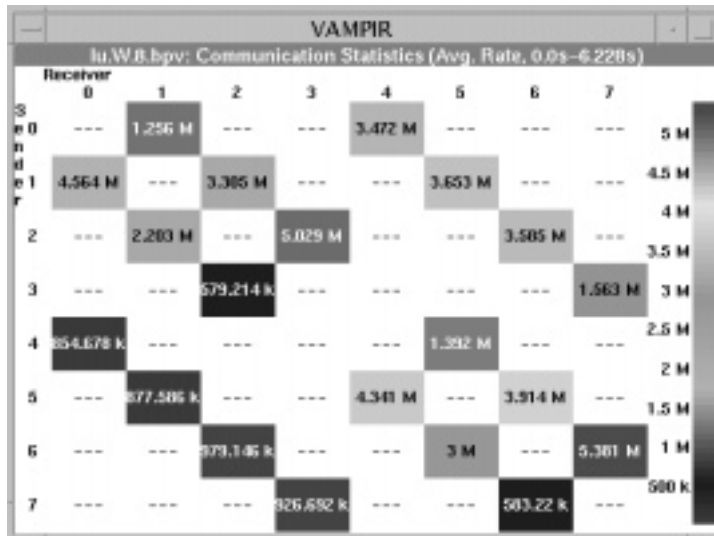


Figure 198. Vampir Average Message Rate Display

Message length distribution histograms are also available, as shown in Figure 199.

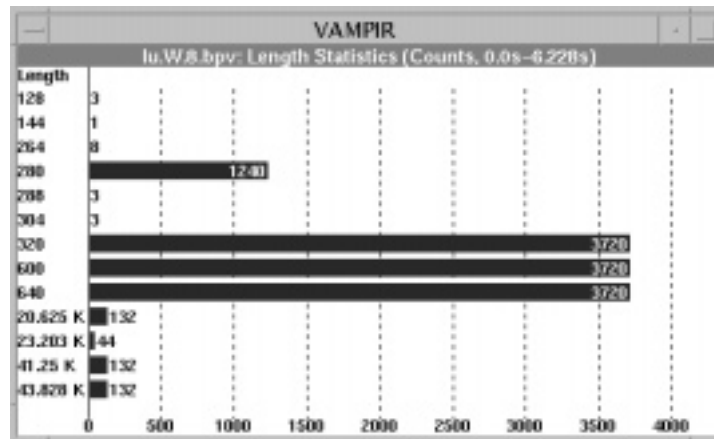


Figure 199. Vampir Message Length Distribution Histogram

- Execution statistics display global (all processes) or local (one process) subroutine execution metrics for an arbitrary time interval. Typical examples are pie charts that present the fraction of time spent in the

application, in message passing, and in profiling as shown in Figure 200. The user can also select specific processes, procedures, and time intervals and display the corresponding activity.

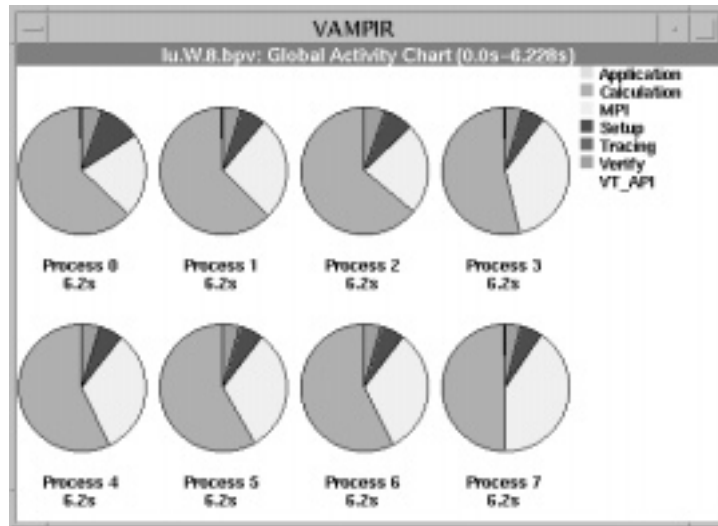


Figure 200. Vampir Global Execution Statistics

Figure 201 on page 495 shows the time spent in selected MPI calls for one of the processes. Call-tree comparison between different program runs is also possible to evaluate the effect of optimizations.

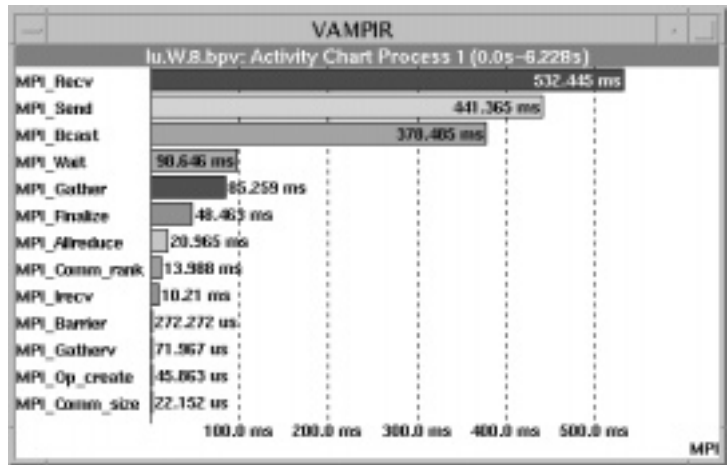


Figure 201. Vampir Local Execution Statistics (User-Selected Routines)

496 The RS/6000 SP Inside Out

Chapter 17. LoadLeveler

LoadLeveler is a job management system originally developed at the University of Wisconsin. It is offered by IBM to provide the facility for building, submitting and processing batch jobs within a network of machines.

This network of machines may include an SP, other IBM RS/6000s and other types of workstations. The network environment may include Distributed Computing Environment (DCE), AFS, NFS and NQS. All machines in this network that can run LoadLeveler jobs are grouped together and called a cluster.

A batch job consists of a set of job steps. Each job step is an executable that is run on a node or a group of nodes. A job is defined by a Job Command File, which stores the name of the job, the job step(s) involved and other LoadLeveler statements. It is this file that is submitted to LoadLeveler for execution.

A job can either run on one machine (serial) or multiple machines (parallel).

When batch jobs are submitted to LoadLeveler by users, LoadLeveler schedules their execution by matching their requirements with the best available machine resources.

LoadLeveler jobs can be submitted and monitored via commands or the GUI called xloadl.

17.1 Architecture

Once a machine becomes part of a LoadLeveler cluster, it can take on one or more of four roles:

- **Central Manager Machine or the Negotiator.** A central manager is a node dedicated to examining a submitted job's requirements and finding one or more nodes in the cluster to run the job. There is only one central manager per LoadLeveler cluster; however, it is possible to identify an alternate central manager which becomes the central manager if the primary central manager becomes unavailable.
- **Scheduling Machine.** When jobs are submitted to LoadLeveler, they get stored in a queue on the scheduling machine. It is this machine's responsibility to contact the central manager and ask it to find an appropriate machine to run the job.
- **Executing Machine.** This is a node that runs the jobs submitted by users.

- **Submit-only Machine.** This type of machine can only submit, query and cancel jobs. A submit-only machine does not carry out any scheduling or execution of jobs. This feature allows machines outside of a LoadLeveler cluster to submit jobs. Submit-only machines have their own GUI which contains a subset of LoadLeveler functions.

Depending on job requirements, a machine in a cluster may take on multiple roles. A machine outside of a cluster can only take part as a submit-only machine.

When a node is part of a LoadLeveler cluster, it may be necessary for the node to stop processing jobs for tasks such as a routine backup. The administrator of a LoadLeveler node therefore can put the node into one of four states:

- Available
- Available only during certain hours
- Available when the keyboard and mouse are not being used interactively
- Not available

There are six daemons that are used by LoadLeveler to process jobs:

- **master** — This daemon runs on every node in a LoadLeveler cluster. It does not, however, run on any submit-only machines. It manages all other LoadLeveler daemons running on the node.
- **schedd** — This daemon receives the submitted jobs and schedules them for execution. The scheduling is based on the machines selected by the negotiator daemon (discussed later in this section) on the central manager. The schedd is started, restarted, signalled and stopped by the master daemon.
- **startd** — This daemon monitors jobs and machine resources on all executing machines in the cluster. It communicates the machine availability to the central manager, and receives a job to be executed from the schedd.
- **starter** — This is spawned by the startd daemon when startd receives the job from schedd. The starter daemon is responsible for running the jobs and reporting the status back to startd.
- **negotiator** — This daemon runs on the central manager and records information regarding the availability of all executing machines to perform jobs. It receives job information from schedd and decides which nodes are to perform the job. Once a decision has been made, it sends this information to schedd and lets schedd contact the appropriate executing machines.

- **kbdd** — This daemon monitors the keyboard and mouse activity on a node. It does not work by using interrupts. Rather, it sleeps for a defined period of time and then determines if there have been any keyboard or mouse activities during the time it slept. If so, it sends this information to `startd`.

Figure 202 on page 500 shows the steps LoadLeveler uses to handle a job. They are summarized as follows:

1. A job is first submitted by a user using either the GUI or the `llsubmit` command to the `schedd` daemon on the scheduling machine. The submitting machine can be a submit-only machine or any other machine in the cluster.
2. The `schedd` daemon on the scheduling machine receives the job and places it into the job queue.
3. The `schedd` daemon contacts the negotiator daemon on the central manager to inform it that a job has been placed in the queue. The `schedd` daemon also sends the job description information to the negotiator daemon.
4. The negotiator daemon, which receives updates from the `startd` daemon on all executing machines, checks for an available executing machine with resources which match the job's requirements. Once found, the negotiator daemon sends a "permit to run" signal to the `schedd` daemon.
5. The `schedd` daemon dispatches the job to the `startd` daemon on the executing machine.
6. The `startd` daemon spawns the starter daemon which handles the job. When the job is finished, the starter daemon sends a signal back to the `startd` daemon.
7. The `startd` daemon sends a signal back to the `schedd` daemon informing it that the job has been completed.
8. The `schedd` daemon updates the negotiator daemon that the job has finished.

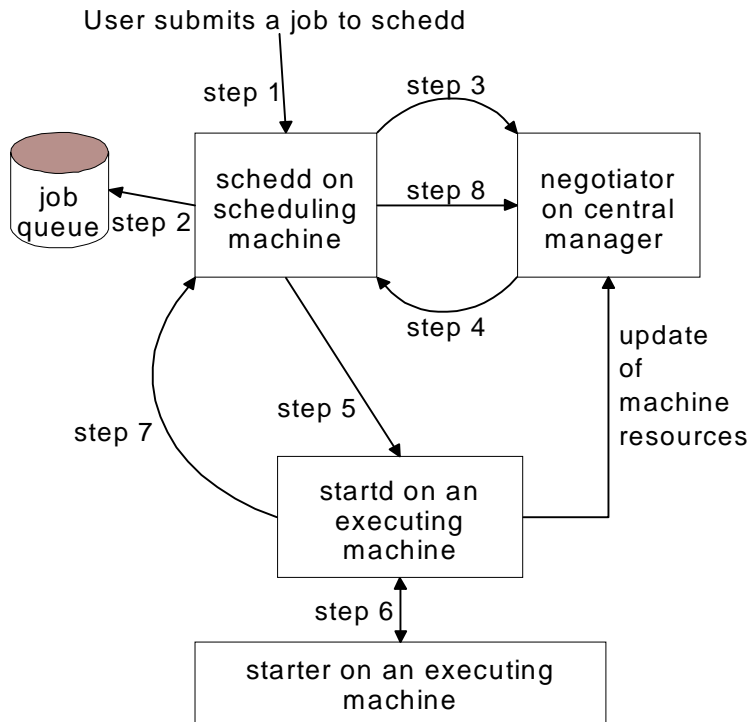


Figure 202. Summary of How LoadLeveler Executes a Job

Note that jobs are not necessarily handled in a first-come, first-served manner. In the job requirements, there are two factors which can influence the scheduling of a job: its class and priority.

A job's class is the user-defined category to which a job belongs, and it is possible to define a specific period when a job may be run. For example, you may want to clean up the contents of a particular file system on a regular basis. This job, however, can only run for a short period of time to minimize outage. You can therefore define a class called `small_job`, with the restriction that any job belonging to this class can only run for 15 minutes on a particular CPU. Classes are defined in the `/home/load/LoadL_admin` file. An example of a class stanza in this file is as follows:

```

small_job:      type = class          # class for small jobs
                priority = 100       # ClassSysprio
                cpu_limit = 15:00    # Limit of 15 minutes
  
```

The second factor that influences job scheduling is job priority. Jobs' priorities determine which job runs ahead of other jobs. LoadLeveler looks at two priorities: the value the user assigns to the job (user priority), and the value LoadLeveler calculates (SYSPRIO).

The user priority is specified in the job command file. It is a number between 0 and 100, inclusive. The higher the number assigned, the greater the priority given to the job.

LoadLeveler calculates its priority for a job by using a formula defined in its configuration file to come up with the SYSPRIO. The system administrator sets up the formula, which can be based on a number of factors. Details on how to set up the SYSPRIO formula can be found in *IBM LoadLeveler for AIX Using and Administering Version 2 Release 1*, SA22-7311.

LoadLeveler schedules a job based on an adjusted system priority, a number that factors in both the user priority and the SYSPRIO. When jobs are submitted, LoadLeveler does the following:

1. Assigns a SYSPRIO
2. Orders the jobs by SYSPRIO
3. Assigns an adjusted system priority to jobs belonging to the same user and the same class, by taking all jobs with the same SYSPRIO and ordering them according to the user priority

17.2 Configuration of LoadLeveler

The installation and configuration of LoadLeveler is summarized in 10.3, "Installing and Configuring LoadLeveler", in *PSSP 3.1 Announcement*, SG24-5332.

The installation procedure is described in detail in *Installation Memo for IBM LoadLeveler Version 2 Release 1*, G110-0642.

After installing LoadLeveler, you need to configure it for your system. Global configuration information includes the following:

- LoadLeveler user ID and group ID (default user ID is loadl)
- The configuration directory (default is /home/loadl)
- The global configuration file (default is LoadL_config)

There are three major steps to configure LoadLeveler:

1. Set up the administration file, which lists and defines the machines in a LoadLeveler cluster. The default name of this file is

/home/loadl/LoadL_admin, which assumes that you have given the user id loadl for LoadLeveler.

2. Set up the global configuration file that contains the parameters controlling how LoadLeveler operates in the cluster.
3. Set up local configuration files on machines to define settings for specific nodes. The default file is
/home/loadl/<hostname_of_node>/LoadL_config.local.

Configuration of LoadLeveler can be complex. The procedure is discussed in depth in *IBM LoadLeveler for AIX Using and Administering Version 2 Release 1*, SA22-7311.

Once LoadLeveler has been installed and configured, the command to start it is `/usr/lpp/LoadL/full/bin/llctl -g start`. This starts LoadLeveler in all the machines defined to be part of the LoadLeveler cluster.

To check on the status of all the machines in a LoadLeveler cluster, run `/usr/lpp/LoadL/full/bin/llstatus`.

17.3 Serial Jobs

Once LoadLeveler is configured and started, you can begin to build and submit jobs to be run. There are two ways to build the jobs: write your own job command file or use the xloadl GUI. By default, the name of a job command file ends in `.cmd`.

The following is an example of a job command file; it is called `errpt.cmd`:

```
#!/bin/ksh
#@ job_type = serial
#@ executable = /usr/bin/errpt
#@ arguments = -a
#@ output = $(Executable).$(Cluster).$(Process).out
#@ error = $(Executable).$(Cluster).$(Process).err
#@ initialdir = /usr/lpp/LoadL/full/bin
#@ notify_user = loadl@sp4en0.msc.itso.ibm.com
#@ notification = always
#@ checkpoint = no
#@ restart = no
#@ requirements = (Arch == "R6000") && (OpSys == "AIX43")
#@ queue
```

All lines beginning with `#` are treated as comments. LoadLeveler commands begin with `#@`. The word following the `#@` is the LoadLeveler keyword which describes the job to be done. The LoadLeveler keyword is case insensitive.

A job command file consists of a number of job steps, or commands to be executed. LoadLeveler stores a job by placing each job step into the queue managed by the schedd daemon to await execution. The keyword that lets LoadLeveler know when a job step has been defined is queue. Each job command file must contain at least one job step (or one queue keyword).

In this example, we ask each node in the cluster to run the command `errpt -a` and store it in a unique filename as defined by the output keyword.

The job_type keyword is optional and defines the type of job, serial or parallel, that we are running.

The executable keyword describes the command that you want to run. This can be a system command or a script you wrote. If the executable keyword is not used, then the job command file itself is treated as the executable. You need to then specify the commands as though the job command file, that is, without a # or a #@.

The arguments keyword specifies the flags that the executable requires.

The output and error keywords define the files to which LoadLeveler will write the respective output and error messages.

The initialdir keyword specifies the path name of the directory to be used as the initial directory during the execution of the job step. The directory specified must exist on both the submitting machine and the machine where the job runs.

The notification and notify_user keywords specify whether LoadLeveler is to send any messages and to which user the messages are to be sent.

The checkpoint keyword specifies whether you want to use checkpointing or not. Checkpointing means that the state of a job is periodically saved for recovery purposes should the node go down. It can be specified via the application program (user initiated) or taken at intervals specified in the LoadLeveler configuration file. There are a number of issues to consider when checkpointing is used. Refer to 10.1.3, "Checkpointing", in *PSSP 3.1 Announcement*, SG24-5332 for a summary and *IBM LoadLeveler for AIX Using and Administering Version 2 Release 1*, SA22-7311 for details.

The restart keyword specifies whether the central manager is to requeue the job should the node go down and come back up. This is different than a restart using checkpoint because this restarts the whole job, whereas a checkpoint specifies a particular point in the program.

The requirements keyword specifies one or a number of requirements which a machine in the LoadLeveler cluster must meet in order to execute the job step.

Once the job command file is built, it must be submitted for LoadLeveler to place on the queue. The command to execute this is `llsubmit`. For our previous example, run `llsubmit errpt.cmd` to submit the job to LoadLeveler. If the command is successful, it returns the message:

```
llsubmit: The job "sp4n05.msc.itso.ibm.com.1" has been submitted.
```

The job name given is the name of the node where the job is submitted, and a job ID that LoadLeveler assigns. In this case, the job ID is 1.

The preceding is just a simple example of a job command file for a serial job. There are many other ways to construct a job command file. For a detailed explanation of the possible structures (including examples) and all the keywords that can be specified in a job command file, refer to *IBM LoadLeveler for AIX Using and Administering Version 2 Release 1, SA22-7311*.

To use the GUI `xloadl` to handle the chores of building and submitting jobs, run `xloadl` from the command line. This starts the GUI shown in Figure 203 on page 505.

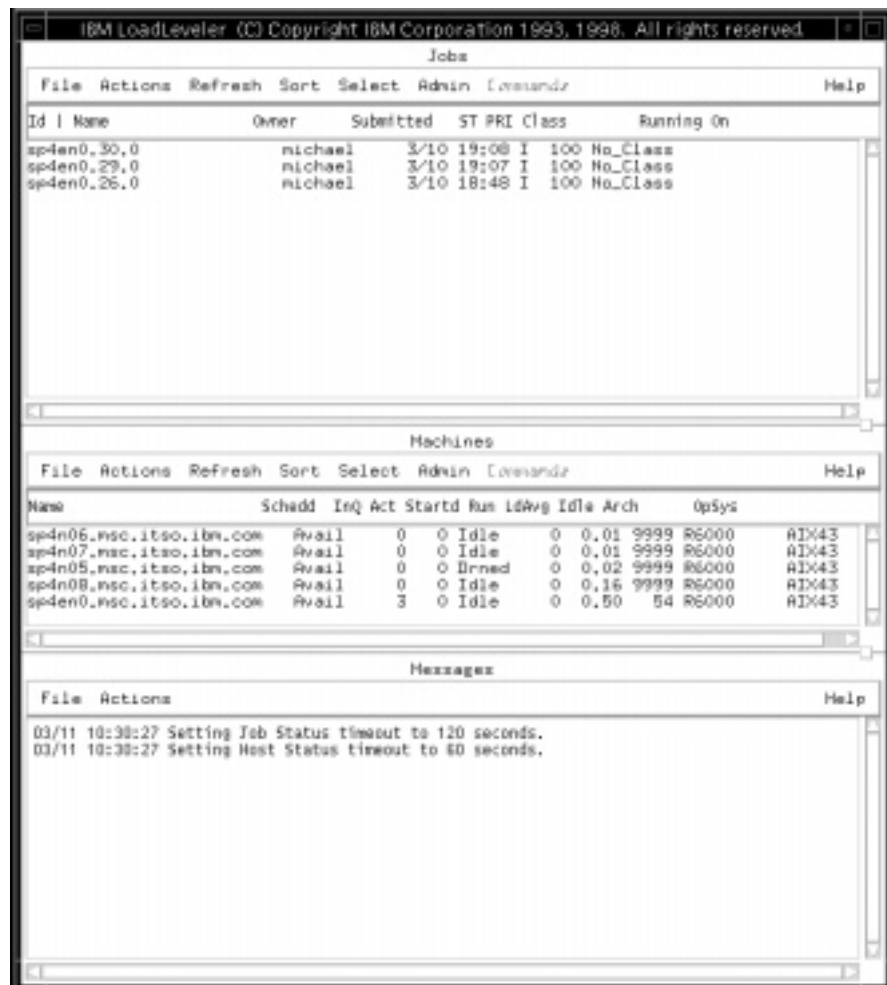


Figure 203. The xloadl GUI

To build a serial job, select **File**, then **Build_File_Job** and finally **File_Build_Job_Serial**. This brings up the dialog box where you can fill in the values for all keywords available in a job command file for serial jobs. Figure 205 on page 507 shows this dialog box.



Figure 204. Dialog Box for Writing a Job Command File

Once the file has been built, you have the option to submit the file right away or save it first and submit it later. To submit it immediately, click on the **Submit** button. To save the file first, click on the **Save** button and specify the file name in the dialog box that pops up.

To submit a job command file that has already been built, choose from the main xload GUI **File, File_Submit_Job** and select the job command file you want to submit in the associated dialog box. This dialog box is shown in Figure 205 on page 507.

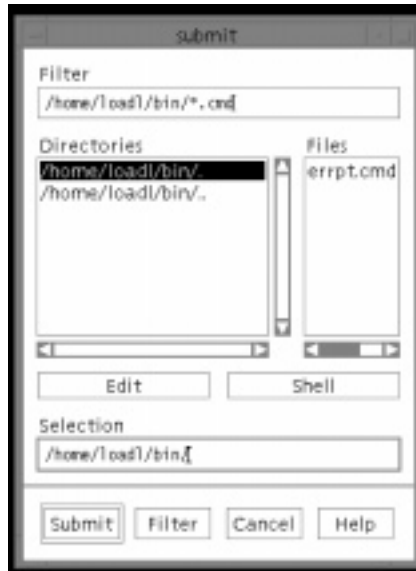


Figure 205. Dialog Box for Submitting a Job

After a job command file has been submitted for execution, you can run the command `/usr/lpp/LoadL/full/bin/llq` to check on its status within a queue. The following is a sample output from running `llq`:

```

$ llq
Id                Owner      Submitted  ST PRI Class      Running On
-----
sp4en0.21.0      spuser1   3/10 13:02 ST 100 No_Class      sp4n06
1 job steps in queue, 0 waiting, 1 pending, 0 running, 0 held

```

17.4 Parallel Jobs

LoadLeveler allows you to submit parallel jobs that have been written using the following:

- IBM Parallel Environment Library (POE/MPI/LAPI) 2.4.0
- Parallel Virtual Machine (PVM) 3.3 (RS6K architecture)
- Parallel Virtual Machine (PVM) 3.3.11+ (SP2MPI architecture)

After choosing the method to submit parallel jobs, install the appropriate software. Information on parallel programming is found in Chapter 16, “Parallel Programming” on page 453.

You have to make some changes to the LoadLeveler configuration to enable it to handle parallel jobs.

For POE jobs, ensure that all the adapters available to be used on each node have been identified to LoadLeveler. To do this, run `llextSDR` to get the node and adapter information out of the SDR. The following example shows what these entries look like for a node with the hostname `sp4n05`, one Ethernet adapter and one SP switch adapter:

```
sp4n05:
type = machine
adapter_stanzas = sp4sw05.msc.itso.ibm.com sp4n05.msc.itso.ibm.com
spacct_exclude_enable = false
alias = sp4sw05.msc.itso.ibm.com sp4n05.msc.itso.ibm.com

sp4sw05.msc.itso.ibm.com:
type = adapter
adapter_name = css0
network_type = switch
interface_address = 192.168.14.5
interface_name = sp4sw05.msc.itso.ibm.com
switch_node_number = 4

sp4n05.msc.itso.ibm.com:
type = adapter
adapter_name = en0
network_type = ethernet
interface_address = 192.168.4.5
interface_name = sp4n05.msc.itso.ibm.com
```

Create two adapter stanzas in the `LoadL_admin` file, one for each adapter, and then add an `adapter_stanzas` line to the machine stanza. The following example shows these entries:

```

sp4n05: type = machine
        central_manager = false
        adapter_stanzas = sp4n05_en0 sp4n05_css0
        spacct_exclude_enable = false

sp4n05_en0:    type = adapter
               adapter_name = en0
               interface_address = 192.168.4.5
               interface_name = sp4n05.msc.itso.ibm.com
               network_type = ethernet

sp4n05_css0:  type = adapter
               adapter_name = css0
               interface_address = 192.168.14.5
               interface_name = sp4sw05.msc.itso.ibm.com
               network_type = switch
               switch_node_number = 4

```

After adding these entries, ensure that you have selected an appropriate scheduler for your job (for information on scheduling, refer to 17.5, “Scheduling” on page 511).

At this point, you can consider whether you want to set up specific classes in the LoadL_admin file to describe POE job characteristics. Other optional configurable functions include grouping nodes into pools, restricting nodes to handle particular types of jobs (batch, interactive or both), and turning on SP exclusive use accounting.

Once these have been considered and added to the LoadL_admin file, you are ready to start LoadLeveler.

For PVM 3.3 RS6K architecture jobs, you need to set up a path for LoadLeveler to find the PVM software. LoadLeveler by default expects that PVM is installed in ~loadl/pvm3. If the software is installed elsewhere, include this in the machine stanza in the LoadL_admin file. For example, if you have PVM installed in /home/loadl/aix43/pvm3, include the following line in the machine stanza:

```
pvm_root = /home/loadl/aix43/pvm3
```

This entry sets the environment variable \$PVM_ROOT, required by PVM.

If you are running PVM 3.3.11+, LoadLeveler does not expect the software to be installed in ~loadl/pvm3. Rather, LoadLeveler expects PVM to be installed in a directory that is accessible to and executable by all nodes in the LoadLeveler cluster.

Both versions of PVM have a restriction that limits each user to only run one instance of PVM on each node. You can ensure that LoadLeveler does not start more than one PVM job per node by setting up a class for PVM jobs. For more information on this topic, refer to Chapter 6, "Administration Tasks for Parallel Jobs", in *IBM LoadLeveler for AIX Using and Administering Version 2 Release 1*, SA22-7311.

Once LoadLeveler is properly set up, it is possible to build and submit jobs either by the command line or the `xloadl` GUI.

The GUI for building and submitting parallel jobs is the same as the one for serial jobs. The difference is the selection in the **Job Type** field under the **Build a Job** dialog box. You have your choice of either **Serial**, **Parallel** or **PVM**.

Here is an example of a job command file for a parallel job (using POE):

```
# @ job_type = parallel
# @ output = poe_job.out
# @ error = poe_job.err
# @ node = 4, 10
# @ tasks_per_node = 2
# @ network.LAPI = switch,shared,US
# @ network.MPI = switch,shared,US
# @ wall_clock_limit = 1:30, 1:10
# @ executable = /home/spuser1/poe_job
# @ arguments = /poe_parameters -euilib "share"
# @ class = POE
# @ queue
```

We can take a closer look at this job command file.

The `job_type` obviously must be set to parallel.

The output and error entries remain the same as in the case of a serial job.

The `node` keyword can be used to specify a minimum and maximum number of nodes that this job can use. The format is `node = min, max` where `min` is the minimum number of nodes this job requires and `max` is the maximum number of nodes required. In this case, the values of 4 and 10 indicate that the job requires at least 4 nodes but no more than 10.

Since the backfill scheduler available with LoadLeveler 2.1 supports multiple tasks being scheduled on a node, you can specify how many jobs are to be scheduled per node. This is done using the `tasks_per_node` keyword, which is used in conjunction with the `node` keyword.

The network keyword specifies the communication protocols, adapters and their characteristics to be used for the parallel job. In this instance, we define two protocols, *LAPI* and *MPI*. For each, we are to use the network type of *switch*. The *shared* option means that the adapters can be shared with tasks from other job steps, and *US* refer to the mode of the communication subsystem. (US means User Space; the other option is Internet Protocol, specified by IP.)

The wall_clock_limit sets the time limit for running a job. It is required, and is set either in the job command file by the user or in the class configuration in the LoadL_admin file by the administration when a backfill scheduling algorithm is used. The two numbers for the wall_clock_limit in this example specify a hard limit of one hour and thirty minutes and a soft limit of one hour and ten minutes. The hard limit is the absolute maximum while the soft limit is an upper bound that may be crossed for a short period of time.

The executable and arguments keywords are used just like in a serial file to specify the job to run.

The class is specified to be POE in this example.

The queue keyword is once again required to let the system know when a job step's definitions have been completed and that the schedd can place this job step on the queue to be dispatched by LoadLeveler.

The llsubmit command is used to submit the job command file to LoadLeveler. Once submitted, you can view its status either in the GUI or through the use of the llq command.

One major change that has been implemented in LoadLeveler 2.1 is the incorporation of certain Resource Manager functions into the program to run parallel batch jobs, since Resource Manager is no longer offered in PSSP 3.1. LoadLeveler 2.1, for example, is able to load and unload the Job Switch Resource Table (JRST) itself using a switch table API. LoadLeveler can also provide the JRST to other programs, such as POE. The JRST is discussed in greater detail in Chapter 11, "Parallel Environment 2.4" in *PSSP 3.1 Announcement*, SG24-5332, while the switch table API is detailed in *IBM Parallel System Support Program for AIX Command and Technical Reference, Version 3 Release 1*, SA22-7351.

17.5 Scheduling

Once jobs have been defined and submitted to LoadLeveler, they are scheduled for execution. LoadLeveler can schedule jobs using one of three possible algorithms: the default LoadLeveler scheduler, the backfill scheduler

and the job control API. The scheduling method is set by manipulating two keywords within the global configuration file: SCHEDULER_API and SCHEDULER_TYPE. Recall from section 17.2, “Configuration of LoadLeveler” on page 501, that the system default name for the global configuration file is /home/loadl/LoadL_config.

The default LoadLeveler scheduler is meant primarily for serial jobs, although it can handle parallel jobs. This algorithm schedules jobs according to a machine’s MACHPRIO. The higher the MACHPRIO, the more available a machine is to handle a job.

The MACHPRIO keyword can set to an expression defined by the system administrator and used by the central manager to measure executing machines’ ability to handle jobs. It is based on a combination of eight factors:

- LoadAvg - the one-minute load average of the machine
- Cpus - the number of processors in the machine
- Speed - the relative speed of the machine as defined in the LoadL_admin file
- Memory - the size of real memory (in MB) on the machine
- VirtualMemory - the size of the available swap space on the machine in KB
- Disk - the amount of free space left in the file system where the executables reside in KB
- CustomMetric - allows you to set a relative priority number for one or more machines
- MasterMachPriority - a value of 1 for this keyword specifies a machine as a master node for a parallel job

The MACHPRIO can only be set in the global configuration file or the configuration file local on the central manager.

The default LoadLeveler scheduler continues to monitor a machine after it has been assigned a job to ensure that the workload is not too heavy. If the workload is deemed to be too heavy for the node, the job may be suspended and resumed at a later time. If this method is used to handle parallel jobs, it uses a reservation method to accumulate the required number of nodes before dispatching the job. The problem with this method is that if a machine is reserved, it cannot handle any other jobs. If a job requires a large number of nodes, it is possible to waste valuable resources because nodes that otherwise can be used to process small jobs are held in reserve for the large job. It is also possible that a large job with a low priority will never get

dispatched because LoadLeveler is unable to accumulate the sufficient number of nodes.

The default LoadLeveler scheduler is set by including this expression in the global configuration file:

```
SCHEDULER_API = NO
```

Do not include the SCHEDULER_TYPE keyword because it overwrites the SCHEDULER_API expression. Simply set SCHEDULER_API to NO.

The backfill scheduler can be used for both serial and parallel jobs, although it is designed to handle primarily parallel jobs. The backfill scheduler requires that every job sent to schedd has the wall_clock_limit set in its job command file. This defines the maximum amount of time in which the job will complete. Using this information, the backfill algorithm works to ensure that the highest priority jobs are not delayed.

The backfill scheduler supports the scheduling of multiple tasks per machine and the scheduling of multiple user spaces per adapter. This means that if a node is reserved for running a large job, and a small job with high priority is received (one that can be completed before the large job is started), the backfill scheduler algorithm permits LoadLeveler to run this small job on the node.

The backfill scheduler is defined in the configuration file by this line:

```
SCHEDULER_TYPE = BACKFILL
```

Setting SCHEDULER_TYPE overrides any entry you may have in the SCHEDULER_API keyword.

Job control API can be specified if you want to use an external scheduler. This API is intended for those users who want to create a scheduling algorithm for parallel jobs based on specific on-site requirements. It provides a time-based interface, instead of an event-based interface. Further information on the Job Control API is in *IBM LoadLeveler for AIX Using and Administering Version 2 Release 1, SA22-7311*.

To enable the job control API, include this line in the configuration file:

```
SCHEDULER_API = YES
```

Do not include a SCHEDULER_TYPE entry.

17.6 SMP Features

At Version 2 Release 1, LoadLeveler has added the support of multiple tasks being dispatched to nodes and the creation of multiple user spaces per adapter. This feature can improve performance when combined with the backfill scheduling algorithm, as described in section 17.5, "Scheduling" on page 511.

Unfortunately, this does not mean that LoadLeveler is able to schedule jobs on individual CPUs within SMP nodes. LoadLeveler treats all nodes, regardless of the number of CPUs within the node, as one machine and dispatches jobs to the machine to be run. You can, however, specify a job to be run on a particular node. If you have a job that needs to run on an SMP node, you can define that particular job in a class called SMP. Then, you specify in the nodes' configuration files which node can run the SMP class job.

For example, you can add the following stanza in the administration file:

```
smp: type = class
class_comment = "This is a class that defines a job to require a SMP node"
priority = 90
```

And then add in the local configuration file on the SMP node:

```
Class = SMP
```

You can also add in other jobs that you want the node to handle. Recall that the parameters specified in the local configuration file overwrites the settings in the global configuration file. Therefore, any job classes that you defined as being allowed to run on the SMP node will not run unless you specify them again in the local configuration file.

17.7 DCE Security Integration

For those running the Distributed Computing Environment (DCE) and LoadLeveler, there is one additional factor of which you need to be aware.

When users log in using DCE, they are issued a DCE ticket granting ticket. If they then submit a job using LoadLeveler, LoadLeveler keeps a copy of the DCE ticket granting ticket. If this job has to wait in the queue, there is a chance that the DCE ticket granting ticket may expire when the job is finally dispatched by LoadLeveler. If this is the case, the job will not run because there is insufficient authentication.

We therefore recommend that you exercise caution and set a reasonable expiration date for DCE ticket-granting tickets.

516 The RS/6000 SP Inside Out

Appendix A. Currently Supported Adapters

These lists are for reference only. A complete list and further documentation can be found at the official Web site for RS/6000 products at the following URL:

<http://www.rs6000.ibm.com>

Table 21. Supported Network Interface Cards for MCA Nodes

Feature Code	Card Type	Description
2972	[8-S]	Auto Token-Ring LAN Streamer Adapter
2980	[2-1]	Ethernet Adapter
2992	[8-U]	Ethernet 10 Mbps AUI/RJ-45 Adapter
2993	[8-V]	Ethernet 10 Mbps BNC Adapter
2994	[9-K]	Ethernet 10/100 Mbps Adapter
2724	[2-R]	FDDI-Fiber Single Ring Adapter
2723	[2-S]	FDDI-Fiber Dual Ring Upgrade Adapter
2989	[9-9]	TURBOWAYS 155 Mbps ATM Adapter
2960	[2-4]	X.25 Interface Co-processor/2
2700	[2-3]	4-Port Communications Controller (SDLC)

Table 22. Supported SCSI and SSA Adapters for MCA Nodes

Feature Code	Card Type	Description
2412	[4-C]	Enhanced SCSI-2 Differential Fast/Wide Adapter
2415	[4-7]	SCSI-2 Fast/Wide Adapter
6216	[4-G]	Enhanced SSA 4-Port Adapter
6219	[4-M]	SSA Multi Initiator/RAID EL Adapter

Table 23. Other Supported Adapters for MCA Nodes

Feature Code	Card Type	Description
2754	[5-3]	S/390 ESCON Channel Emulator

Feature Code	Card Type	Description
2755	[5-2]	Block Multiplexer Channel Adapter
2756	[5-3]	ESCON Control Unit Adapter
2930	[3-1]	8-Port Asynchronous Adapter (EIA-232)
8128	[3-7]	128-Port Asynchronous Controller
7006		RICP 1 MB Portmaster Adapter
6305	[6-6]	Digital Trunk Dual Adapter

Table 24. Supported Network Interface Cards for PCI Nodes

Feature Code	Card Type	Description
2920	[9-O]	Auto LAN Streamer Token Ring Adapter
2985	[8-Y]	Ethernet 10 Mbps BNC/RJ-45 Adapter
2987	[8-Z]	Ethernet 10 Mbps AUI/RJ-45 Adapter
2968	[9-P]	10/100 Mbps Ethernet Adapter
2969	[9-U]	Gigabit Ethernet adapter
2741		SysKonnnect SK-NET FDDI-LP SAS
2742		SysKonnnect SK-NET FDDI-LP DAS
2743		SysKonnnect SK-NET FDDI-UP SAS
2963	[9-J]	Turboways 155 Mbps UTP ATM Adapter
2988	[9-F]	Turboways 155 Mbps MMF ATM Adapter
2962	[8-L]	2-Port Multiprotocol Adapter

Table 25. Supported SCSI and SSA Adapters for PCI Nodes

Feature Code	Card Type	Description
6206	[4-K]	UltraSCSI Single-Ended Adapter
6207	[4-L]	UltraSCSI Differential Adapter
6208	[4-E]	SCSI-2 Fast/Wide Adapter
6209	[4-F]	SCSI-2 Differential Fast/Wide Adapter

Feature Code	Card Type	Description
6215	[4-N]	SSA Multi-Initiator/RAID EL Adapter

Table 26. Other Supported Adapters for PCI Nodes

Feature Code	Card Type	Description
2751	[5-5]	S/390 ESCON Channel Adapter
2943	[3-B]	8-Port Asynchronous Adapter (EIA-232/RS 422)
2934	[3-C]	128-Port Asynchronous Controller
2947	[9-R]	ARTIC960Hx 4-Port Selectable Adapter
6310	[4-E]	ARTIC960RxD Quad Digital Trunk adapter (T1/E1)

520 The RS/6000 SP Inside Out

Appendix B. 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, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this

information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM ®	AIX
BookManager	Global Network
ESCON	HACMP/6000
LoadLeveler	OS/390
POWERparallel	RS/6000
S/390	SP

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered

trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries. (For a complete list of Intel trademarks see www.intel.com/dradmarx.htm)

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

524 The RS/6000 SP Inside Out

Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see "How to Get ITSO Redbooks" on page 529.

- *PSSP 3.1 Announcement*, SG24-5332
- *RS/6000 SP PSSP 2.4 Technical Presentation*, SG24-5173
- *RS/6000 SP PSSP 2.3 Technical Presentation*, SG24-2080
- *RS/6000 SP PSSP 2.2 Technical Presentation*, SG24-4868
- *Inside the RS/6000 SP*, SG24-5145
- *SP Perspectives: A New View of Your SP System*, SG24-5180
- *RS/6000 SP Monitoring: Keeping it Alive*, SG24-4873
- *RS/6000 SP: PSSP 2.2 Survival Guide*, SG24-4928
- *Understanding and Using the SP Switch*, SG24-5161
- *RS/6000 SP Performance Tuning*, SG24-5340 (Available in June 1999)
- *RS/6000 Performance Tools in Focus*, SG24-4989
- *Elements of Security: AIX 4.1*, GG24-4433
- *RS/6000 SP High Availability Infrastructure*, SG24-4838
- *GPFS: A Parallel File System*, SG24-5165
- *IBM 9077 SP Switch Router: Get Connected to the SP Switch*, SG24-5157
- *RS/6000 SP: Problem Determination Guide*, SG24-4778
- *SP System Management: Easy, Lean and Mean*, GG24-2563
- *AIX Storage Management*, GG24-4484

C.2 Redbooks on CD-ROMs

Redbooks are also available on the following CD-ROMs:

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177

CD-ROM Title	Collection Kit Number
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbook	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037

C.3 Other Publications

These publications are also relevant as further information sources:

- *IBM Parallel System Support Programs for AIX: Installation and Migration Guide, GA22-7347*
- *IBM Parallel System Support Programs for AIX: Administration Guide, SA22-7348*
- *IBM Parallel System Support Programs for AIX: Diagnosis Guide, GA22-7350*
- *IBM Parallel System Support Programs for AIX: Messages Reference, GA22-7352*
- *IBM Parallel System Support Programs for AIX: Command and Technical Reference, Volume 1 and Volume 2, SA22-7351*
- *IBM Parallel System Support Programs for AIX: Managing Shared Disks, SA22-7349*
- *IBM RS/6000 SP Planning Volume 1, Hardware and Physical Environment, GA22-7280*
- *IBM RS/6000 SP Planning Volume 2, Control Workstation and Software Environment, GA22-7281*
- *RS/6000 Cluster Technology: Event Management Programming Guide and Reference, SA22-7354*
- *RS/6000 Cluster Technology: Group Services Programming Guide and Reference, SA22-7355*
- *IBM Parallel System Support Programs for AIX Performance Monitoring Guide and Reference, SA22-7353*

- *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278
- *HACMP for AIX: Enhanced Scalability Installation and Administration Guide*, SC23-4284
- *Parallel Environment for AIX: Operation and Use, Volume 1*, SC28-1979
- *Parallel Environment for AIX: Installation Version 2 Release 4*, GC28-1981
- *Parallel Environment for AIX: Operation and Use, Volume 2, Part 1 and 2*, SC28-1980
- *Parallel Environment for AIX: Hitchhiker's Guide*, GC23-3895
- *IBM XL High Performance Fortran Language Reference and User's Guide*, SC09-2631
- *IBM XL Fortran Language Reference*, SC09-2718
- *IBM XL Fortran User's Guide*, SC09-2719
- *LoadLeveler for AIX: Using and Administering Version 2 Release 1*, SA22-7311
- *AIX Version 4.3 System User's Guide: Communications and Networks*, SC23-4127
- *Distributed Computing Environment for AIX, Version 2.2: Quick Beginnings*, SC23-4188
- *AIX Version 4.3 Commands Reference*, SC23-4119
- *AIX Version 3.2 and 4 Performance Monitoring and Tuning Guide*, SC23-2365
- *AIX Version 4.3 System Management Guide: Operating System and Devices*, SC23-4126
- *AIX Version 4.3 Network Installation Management Guide and Reference*, SC23-2627

C.4 External Publications

These are non-IBM publications relevant as further information sources:

- *Managing NIS and NFS*, O'Reilly and Associates
- *The Kerberos Network Authentication Service (V5)*, RFC1510
- *Telnet Authentication Option*, RFC1416
- *Generic Security Service API*, RFC1508

528 The RS/6000 SP Inside Out

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO 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 via e-mail including information from the 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.ibm.link.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.ibm.link.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.ibm.link.ibm.com/pbl/pbl/

This information was current at the time of publication, but is continually subject to change. The latest information for customer may be found at <http://www.redbooks.ibm.com/> and for IBM employees at <http://w3.itso.ibm.com/>.

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 also view redbook, residency, and workshop announcements at <http://inews.ibm.com/>.

List of Abbreviations

AIX	Advanced Interactive Executive	GS	Group Services
AMG	Adapter Membership Group	GSAPI	Group Services Application Programming Interface
ANS	Abstract Notation Syntax	GVG	Global Volume Group
API	Application Programming Interface	HACMP	High Availability Cluster Multiprocessing
BIS	boot/install server	HACMP/ES	High Availability Cluster Multiprocessing Enhanced Scalability
BSD	Berkeley Software Distribution	hb	heart beat
BUMP	Bring-Up Microprocessor	HiPS	High Performance Switch
CP	Crown Prince	hrd	host respond daemon
CPU	central processing unit	HSD	Hashed Shared Disk
CSS	communication subsystem	IBM	International Business Machines Corporation
CWS	control workstation	IP	Internet Protocol
DB	database	ISB	Intermediate Switch Board
EM	Event Management	ISC	Intermediate Switch Chip
EMAPI	Event Management Application Programming Interface	ITSO	International Technical Support Organization
EMCDB	Event Management Configuration Database	JFS	Journaled File System
EMD	Event Manager Daemon	LAN	Local Area Network
EPROM	Erasable Programmable Read-Only Memory	LCD	liquid crystal display
FIFO	first-in first-out	LED	light emitter diode
FS	file system	LP	logical partition
GB	gigabytes	LRU	last recently used
GL	Group Leader	LSC	Link Switch Chip
GPFS	General Parallel File System	LV	logical volume
		LVM	Logical Volume Manager
		MB	megabytes

MIB	Management Information Base	RPQ	Request For Product Quotation
MPI	Message Passing Interface	RSCT	RS/6000 Cluster Technology
MPL	Message Passing Library	RSI	Remote Statistics Interface
MPP	Massive Parallel Processors	R/VSD	Recoverable/Virtual Shared Disk
NFS	Network File System	RVSD	Recoverable Virtual Shared Disk
NIM	Network Installation Management	SBS	structured byte string
NSB	Node Switch Board	SCSI	Small Computer System Interface
NSC	Node Switch Chip	SDR	System Data Repository
OID	object ID	SMIT	System Management Interface Tool
ODM	Object Data Manager	SSA	Serial Storage Architecture
PAIDE	Performance Aide for AIX	VG	volume group
PE	Parallel Environment	VSD	Virtual Shared Disk
PID	process ID		
POE	Parallel Operating Environment		
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		

Index

Symbols

/etc/ntp.conf 134
/etc/rc.ntp 134
/etc/security/user 147
/etc/sysctl.pman.acl 297
/usr/lpp/ssp/install/bin/spfbcheck 289

Numerics

100BASE-TX 48, 56, 58
10BASE-2 47
10BASE-T 48
8274 57

A

abbreviations 531
Accelerated Strategic Computing Initiative 3
Access Control Lists 164, 171, 180, 182
ACL callback 181
acronyms 531
Administrative Ethernet 47
ADSM 7
Adstar Distributed Storage Manager 7
Agora 5
AIX Error Notification Facility 331
ALIGN 477
alternate rootvg 291
AMD 365
amd_config 369
Andrew File System 461
ARP cache 56
ASCII 3, 6, 481
asymmetric encryption 146, 150
asynchronous I/O 478
Athena project 141, 148
auditing 139
AUTH callback 181
auth_install 143
auth_methods 143, 166
auth_root_rcmd 143, 165
auth1 147
Authentication 140
 methods 143
 Service 149
 System 149
authenticator 153

Authorization 140, 172
authorization callbacks 181
AutoFS 365, 366
automounter 364, 365, 366
autosensing 58

B

Babel Fish 476
backfill scheduler 511
backup 375
barrier synchronization 464
bcastping 445
BNC 47
boot device 280, 376
boot image 376
boot sector 376
boot/install server 42, 51, 52, 258, 279, 285, 289
bootable image 376
bootlist 292
bootp 273, 288
broadcast storm 55
broadcasts 464
BUMP processor 288

C

C2 level of trust 141
cascading 434
CDS server 168
Central Manager 497
Cerberus 148
chauthpar 170
choice arguments 465
client/server model 140
clinfo 446
clstrmgr 445
cluster 463
Cluster Single-Point-of-Control 434
Collective Communications 464
Collective Operations 465
Commands
 3dmon 319
 add_principal 162
 backup 381
 bffcreate 265
 cfgvsd 400
 chauthent 143

chauthpar 143, 166
 chgcss 308
 chuser 340
 cl_chgroup 445
 cl_chuser 445
 cl_clstop 451
 cl_lsgroup 445
 cl_lsuser 445
 cl_mkgroup 445
 cl_mkuser 445
 cl_rc.cluster 450
 cl_rmgroup 445
 cl_rmuser 445
 cl_setup_kerberos 449
 clchclstr 449
 clhare 434
 clstat 451
 clstop 451
 cpio 381
 create_krb_files 167
 createvsd 398
 css.snap 241, 245, 336
 CSS_test 290
 cw_restrict_login 357
 dbx 481
 dce_login 169
 dcecp 169
 dceunixd 169
 defvsd 399
 dsh 180
 Eannotator 222, 286
 Eclock 238, 239, 287
 Efence 231, 237
 Eprimary 286
 Equiesce 238
 Estart 217, 225, 238
 fencevsd 407
 ftp 141
 hmadm 116, 173
 hmcmds 111, 119, 171
 hmmon 111, 116, 171
 install_cw 267
 k4destroy 159
 k4init 159, 171
 k4list 159, 171
 k5dcelogin 169, 178
 kadmin 162
 kinit 150
 kpasswd 159
 ksrvtgt 162
 ksrvutil 161
 kstash 163
 llctl 502
 llxtSDR 508
 llq 507, 511
 llstatus 502
 llsubmit 511
 llsummary 325
 lsauth 143, 177
 lsauthpar 143
 lsfencevsd 407
 lskp 158
 lsuser 340
 lvlstmajor 444
 mkautomap 372
 mknimres 285
 mksysb 375
 mkuser 340
 mmconfig 418
 mmmkvds 183
 mmremote 183
 mpxlf 484, 491
 nodecond 111, 119, 171, 287
 pdbx 480, 482, 484
 pedb 482, 483, 484
 perspectives 119
 pghpf 480
 pgprof 480
 poe 481, 484, 491
 poeauth 462
 post_process 267
 pssp_script 260, 289
 psspfb_script 289
 ptpectrl 318
 ptpehier 316, 318
 rcmdtgt 179
 rcp 173
 rdump 381
 rexec 142
 rmuser 340
 rsh 143, 173, 175
 rvsdrestrict 406
 s1term 111, 114, 119, 171, 288
 savevg 376
 scp 146
 script.cust 289
 SDR_test 105, 267
 SDRArchive 103, 107, 379

- SDRChangeAttrValues 104, 106
- SDRClearLock 102, 106
- SDRGetObjects 105
- SDRListClasses 106
- SDRRestore 103, 107, 380
- SDRWhoHasLock 102, 106
- setup_authent 164, 168, 267
- setup_server 167, 284
- setup_ssp_server 164
- slogin 146
- smit config_mgmt 103
- snap 334
- spacctnd 321
- spacs_cntrl 346, 347
- spbootins 291
- spbootlist 291
- spchuser 341, 344
- spevent 297
- sphardware 119, 171
- splstdata 103, 282
- splsuser 341, 345
- spmuser 341, 343
- spmon 111, 117, 171
- spmon_itest 267
- spruser 341, 344
- spsetauth 165, 166
- spsitenv 133, 321, 341
- spvsd 304
- ssh 146
- startvsd 401
- supper 356, 360
- sysctl 180
- sysctl svcrestart 304
- SYSMAN_test 286
- syspar_ctrl 279
- tar 381
- telnet 141
- totalview 484, 487
- tprof 489
- unfencevsd 407
- vampir 491
- vsd.snap 337
- vsdnode 397
- vt 488
- xauth 144, 145
- xhost 144
- xlhp 479
- xloadl 504
- xmperf 319

- xntpd 134
- xpdbx 481
- xprofiler 489
- Communicators 464
- concurrent 433, 434
- concurrent access 444
- Configuration Manager 412
- consensus 133
- console 114, 119
- control 109
- control workstation 42
- cookie 144, 145
- CSMA/CD 47
- customization 283, 289
- customize 258
- CustomMetric 512
- CWS
 - See control workstation

D

- Daemons
 - automountd 366
 - clinfo 446
 - css.summlog 240, 241
 - cssadm 227, 229, 235, 237
 - godm 444
 - haem 279
 - hags 279
 - hardmon 110, 267, 270
 - hats 279
 - hc 406
 - hmrmd 173
 - hr 279
 - inetd 457
 - kadmind 162, 164
 - kbdd 499
 - kerberos 162, 164
 - krshd 176, 178
 - master 498
 - negotiator 498
 - nfd 113
 - pmand 330
 - pmanrmd 330
 - pmdv2 456
 - pvmd 487
 - rshd 176, 177
 - rvsd 406
 - s70d 113

schedd 498
 sdrd 91, 98, 99
 setup_logd 267
 sp_configd 330
 spdmapi 315
 spdmcold 315
 spdmspld 315
 splogd 111, 173
 startd 498
 starter 498
 supman 356
 sysctld 180
 Worm 218, 235
 xmservd 312
 ypbind 349
 yppasswd 349
 ypserv 349
 ypupdated 349

DARE
 See Dynamic Automatic Reconfiguration Event

Data distribution directives 477
 Data Encryption Standard 149
 dataless 258
 data-parallel programming 476
 DCE ticket granting ticket 515
 dce_export 147
 dce_login -f 177
 dceunix -t 177
 Department of Energy 481
 design 257
 device database 236
 diagnostics mode 118
 discretionary security control 141
 diskless 258
 DISTRIBUTE 477
 dog coffins 5
 Dolphin Interconnect Solutions 484
 Dynamic Automatic Reconfiguration Event 434
 dynamic port allocation 394
 dynamic reconfiguration 434

E

eavesdropping 148
 EMAPI 203
 endpoint map 394
 Environment Variables
 KRBTKFILE 160, 162
 MP_AUTH 459, 462
 MP_BUFFER_MEM 459
 MP_DEBUG_INITIAL_STOP 484
 MP_MSG_API 476
 MP_NEWJOB 462
 MP_PULSE 487
 MP_TRACELEVEL 488
 MP_USE_LL 459
 SP_NAME 98, 100
 Envoy 5
 Ethernet switch 48, 54
 Etnus, Inc. 481, 484
 Event Management 200
 aixos 208
 client 202
 daemon 187
 EMAPI 187, 204
 EMCDB 204, 205
 event registration 203
 expression 203
 ha_em_peers 204
 haemd 201, 204
 rearm expression 203
 resource class 203, 206
 resource variable 203
 RMAPI 202, 204
 Event Perspective 296
 /etc/sysctl.pman.acl 297
 Condition Pane 302
 Conditions Pane 301
 Create Condition notebook window 301
 Event Condition 301, 302
 Event Definition 301, 303
 event definition 296
 Event Definitions 298
 Event Definition 302
 event icon 300
 Event Management 296
 Event Notification Log 303
 event notification log 300
 icon colors for event definitions 299
 pre-defined events 298
 Rearm Expression 302
 rearm expression 300
 registering events 296, 299
 resource elements ID 303
 resource variable 302
 resource variable class 302
 spevent 297
 unregistering events 296

Executing Machine 497
EXTRINSIC 477

F

Fast Ethernet 56, 57

fault-tolerant 433

File Collections 339, 355

Files

\$HOME/.k5login 178
\$HOME/.netrc 141, 142
\$HOME/.rhosts 142, 177
\$HOME/.Xauthority 145
.k 163, 164
.k5login 169
.klogin 165
/etc/auto.master 366
/etc/auto/maps/auto.u 369, 370
/etc/auto_master 366
/etc/bootptab 260
/etc/bootptab.info 274
/etc/firstboot 289
/etc/group 147, 339, 340
/etc/hosts 101
/etc/hosts.equiv 142, 177
/etc/inetd.conf 177
/etc/inittab 100, 164, 235, 241, 289
/etc/krb.conf 157, 164, 167
/etc/krb.realms 157, 164, 167
/etc/krb5.conf 169
/etc/krb-srvtab 161, 162, 165, 167, 171, 172, 179, 181
/etc/ntp.config 134
/etc/passwd 147, 339, 340
/etc/poe.limits 459
/etc/rc.net 263
/etc/rc.ntp 134
/etc/rc.sp 98, 100
/etc/SDR_dest_info 98, 99, 100, 101, 289
/etc/security/ 147
/etc/security/group 340
/etc/security/login.cfg 147
/etc/security/passwd 339, 340
/etc/security/user 147, 346
/etc/services 98, 158
/etc/SP/expected.top 286
/etc/sysctl.acl 182, 304
/etc/sysctl.conf 180
/etc/sysctl.mmcmd.acl 183

/etc/sysctl.pman.acl 183, 297
/etc/sysctl.vsd.acl 304
/etc/sysctl/logmgt.acl 183
/spdata/sys1/install/images 259, 284
/spdata/sys1/sdr/defs/Frame 113
/spdata/sys1/sdr/system/classes/Frame 113
/spdata/sys1/spmon/hmacls 116, 172
/spdata/sys1/spmon/hmthresholds 111
/spdata/sys1/spmon/hwevents 111
/tftpboot/firstboot.cust 284
/tftpboot/-new-srvtab 167
/tftpboot/script.cust 284, 289
/tftpboot/tuning.cust 267, 284, 289
/tmp/tkt 160
/tmp/tkt_hmrmd 173
/tmp/tkt_splogd 173
/usr/lib/security/DCE 147
/usr/lpp/mmfs/bin/mmcmdsystcl 183
/usr/lpp/ssp/bin/nodecond_chrp 287
/usr/lpp/ssp/bin/nodecond_mca 287
/usr/lpp/ssp/install/bin/psspsfb_script 289
/usr/lpp/ssp/install/config/tuning.default 267
/usr/lpp/ssp/samples/block_usr_sample 347
/usr/sbin/cluster/etc/clhosts 446
/usr/sbin/cluster/etc/clinfo.rc 446
/usr/sbin/cluster/sbin/cl_krb_service 449
/var/adm/SPlogs/css/out.top 248
/var/adm/SPlogs/css/summllog 240
admin.add 164
admin.get 164
admin.mod 164
cssadm.cfg 227
cssadm.debug 230
LoadL_admin 502
LoadL_config 502
logevnt.out 245
logmgt.cmds 183
passwd_overwrite 148
perfagent.tools 262
pman.cmds 183
principal.dir 163
principal.pag 163
firstboot.cust 167
FORALL 477
Fortran 95 479
frame 20, 111
 model frame 21
 short expansion frame 21
 short model frame 21

- SP Switch frame 22
- supervisor interface 114
- tall expansion frame 21
- tall model frame 21

FZJ 491

G

- gather/scatter operations 464
- General Parallel File System (GPFS) 409
- get_auth_method 143, 175, 177, 178
- gettokens.c 461
- Global file systems 387
- global network 447
- Global ODM 188
- GODM
 - See Global ODM
- GRF 39
- Group Services 187
 - barrier synchronization 199
 - clients 193
 - external or user groups 194
 - group 193
 - Group State Value 194
 - Membership List 194
 - Name 194
 - Group Leader 197, 198, 199
 - Group Services Application Programming Interface 195
 - hagsd 193, 196
 - hagsglsmd 196
 - internal components 195
 - internal groups 194
 - join request 196
 - meta-group 196
 - nameserver 195
 - namespace 195
 - Protocols 198
 - providers 193
 - Source-target facility 200
 - subscriber 193
 - sundered namespace 200
 - Voting 199
 - 1-phase 199
 - n phase 199
- GSAPI 195

H

- HACMP 185

- HAI, see also High Availability Infrastructure 185
- Half duplex 48
- hardmon 110, 172
- hardmon principal 171
- hardware address 273
- Hardware Perspective 120
 - Controlling hardware 295
 - icon view 296
 - Monitoring hardware 295
 - panes 296
 - sphardware command 295
 - system objects 295
- hardware_protocol 112
- Hashed Shared Disks 404
- hatsd 189
- HDX
 - See Half Duplex
- heartbeat 447
- high availability 433
- High Availability Cluster Multiprocessing 185
- High Availability Cluster Multiprocessing Enhanced Scalability (HACMP/ES) 433
- High Availability Cluster Multiprocessing Enhanced Scalability Concurrent Resource Manager (HACMP/ESCRM) 433
- High Availability Control Workstation 45
- High Availability Infrastructure 185
- High Performance Fortran 476
- High Performance Gateway Node 39
- High Performance Supercomputer Systems Development Laboratory 4
- home directories 387
- home node 453
- host impersonation 142
- host responds 290
- host responds daemon 207, 209
- HPSSDL 4
- HPSSL 5
- hrd
 - See host responds daemon

I

- I/O pacing 445
- IBM Support Tools 333
 - css.snap 240, 336
 - Gathering AIX Service Information 334
 - Gathering SP-specific Service Information 336
 - Service Director 333

- snap 334
 - vsd.snap 337
- Identification 140
- IEEE POSIX 1003.1-1996 471
- impersonation 140, 142
- implementation 257
- INDEPENDENT 477
- Initial Program Load 260
- in-place buffers 465
- insecure networks 148
- Install Ethernet, 51
- installation 257
- integrated DCE login 147
- Intermediate Switch Board 22
- IPL
 - See Initial Program Load
- ipsrcrouteforward 445
- ipsrcrouterecv 445
- ipsrcroutesev 445
- ISB
 - See Intermediate Switch Board

J

- job command file 502
- job control API 512
- Job Switch Resource Table (JRST) 511

K

- K5MUTE 176
- kcnd 176, 178
- Keberos master key 163
- Kerberos 141, 142, 329, 380
- kerberos port 176
- Kerberos principal 304
- kerberos4 port 176
- Key Distribution Center 169
- key mode 119
- kshell port 176, 178
- kvalid_user 178

L

- LAPI_Amsend 475
- LAPI_Get 475
- LAPI_Put 475
- Launch Pad 119, 293, 294
- libc.a 176
- libspk4rcmd.a 176

- libvaliduser.a 178
- loadable authentication module 147
- LoadAvg 512
- LoadL_starter 457
- LoadLeveler 497
 - Accounting 323
 - Job Resource Data 323
 - Job Resource Information - User Accounts 325
 - llsummary 325
 - parallel jobs 323
 - serial jobs 323
 - administration manual 323
 - cluster 497
 - scheduler 511
 - SYSPRIO 501
 - user priority 501
- Low-Level API 475
- lppsourc 259, 285

M

- MACHPRIO 512
- Massachusetts Institute of Technology 141, 148
- MasterMachPriority 512
- Message Passing Interface 463
- Message Passing Interface Forum 463
- Message Passing Library 476
- mirroring 290, 438
- MIT-MAGIC-COOKIE-1 145
- mksysb 266, 382
- monitor 109
- Monitor and Control Node (MACN) 110
- Monitoring 126
- MPI datatypes 464
- MPI I/O 465
- MPI Version 1.1 464
- MPI Version 1.2 465
- MPI Version 2 465
- multi-homed hosts 157
- mutual authentication 155

N

- n2 problem 185
- National Security Agency 141
- Negotiator 497
- netboot 287
- Netfinity 112, 115, 269
- network boot 284, 287

- Network Connectivity Table 189
- Network File System 259
- Network Information System 339, 348
 - client 349
 - login control 147
 - maps 349
 - Master Server 348
 - Slave Server 348
- Network installation 55
- Network Installation Management 258, 284
- Network Module 448
- Network Option 263
- Network Time Protocol 131
 - ntp_config 133
 - ntp_server 133
 - ntp_version 133
 - peer 132
 - stratum 132
 - timemaster 133
- nfd 115
- NFS
 - See Network File System
- NIM
 - See Network Installation Management
- NIM pull mode 260
- NIM push mode 260
- NIS
 - See Network Information System
- node
 - dependent node 38
 - external node 34
 - High node 26
 - Internal Nodes 26
 - standard node 26
 - Thin node 26
 - Wide node 26
- node database 272
- node supervisor interface 114
- non-concurrent 438
- none 133
- NONE callback 181
- nonlocsrcroute 445
- NTP
 - See Network Time Protocol
- ntp_config 133, 134
- ntp_server 133
- Nways LAN RouteSwitch 57

O

- Oak Ridge National Laboratory 475
- One-Sided Communication 465
- OpenMP 479
- ORNL 475

P

- PAIDE
 - See Performance Aide for AIX
- Pallas GmbH 479, 484, 488, 491
- panes 120
- Parallel Operating Environment 215
- Parallel Tools Consortium 480
- Parallel Virtual Machine 215, 475
- partition manager daemon 455
- Partitioning the SP System
 - See System Partitioning
- passwd_overwrite 148
- PDT
 - See Performance Management Tools
- peer 132
- perfagent.server 186, 265
- perfagent.tools 186, 265
- performance 263
- Performance Aide for AIX 265
- Performance Management 311
 - AIX utilities 311
 - PerfPMR 312
- Performance Monitoring 305
 - Performance Toolbox (PTX/6000) 312
 - Performance Toolbox Parallel Extensions 313
 - System Performance Measurement Interface 312
- Performance Optimization with Enhanced 4
- Performance Toolbox (PTX/6000) 265, 312
 - 3dmon 312
 - Performance Agent 312
 - Performance Manager 312
 - Remote Statistics Interface (RSI) 313
 - xmservd daemon 312
- Performance Toolbox Parallel Extensions 206, 313
 - 3dmon 319
 - Central Coordinator nodes 315
 - Collection of SP-specific data 314
 - Collector nodes 315
 - Data Analysis and Data Relationship Analysis 315
 - Data Manager nodes 315

- Installation and Customization 316
- monitoring hierarchy 316
- ptpe.docs 316
- ptpe.program 316
- ptpectrl 318
- ptpehier 316, 318
- SP runtime monitoring 314
- spdmapid 315
- spdmcold 315
- spdm脾d 315
- ssp.ptpegui 316
- xmperf 319
- personal password 140
- Phoenix 185
- PING_CLIENT_LIST 446
- plain text passwords 142
- planning 257
- PMAN Subsystem 329
 - default events 332
 - Event Management client 331
 - pmand 330
 - pmandefaults script 332
 - pmanrmd 330
 - sp_configd 330
 - ssp.pman 329
 - subscribed events 332
- POE
 - See Parallel Operating Environment
- Point-to-Point Communication 464
- poll rate 111
- poor passwords 140
- Portland Group 479
- POWER 4
- Power Supplies 23
- POWER3 32
- PowerPC 30
- principal 149
- private key 146, 150
- Private Network 441
- Problem Management 208, 326
 - AIX Error Logging 327
 - BSD syslog 327
 - pmanrmd 208
 - SP Error Logs 328
- Process Creation 465
- Process Topologies 464
- PROCESSORS 477
- PROCLAIM messages 56
- Profiling 464

- pssp_script 167
- PTPE
 - See Performance Toolbox Parallel Extensions
- PTX
 - See Performance Toolbox (PTX/6000)
- PTX/6000
 - See Performance Toolbox (PTX/6000)
- public key 146, 150
- Public Network 442
- PVM
 - See Parallel Virtual Machine
- pvm_spawn/pvmfspawn 475

Q

- quorum 279

R

- RAS 8
- rcmd 176, 179
- rcmd principal 179, 180
- r-commands 173
- Read-Only Storage 260
- realm 157
- receive 464
- Recoverable Virtual Shared Disk 405
- reduction operations 464
- registry 147
- Reliable Messaging 189
- remote execution commands 173
- remote node 454
- replay attacks 140, 148, 153
- resource group 434, 440
- Resource Identifier 203
- Resource Manager 511
- Resource Monitor Application Programming Interface 202
- Resource Monitors 201, 205
 - aixos 208
 - external 205
 - aixos 208
 - CSSLogMon 208
 - harmlid 208
 - harmpld 207
 - hmrrmd 207
 - pmanrmd 208
 - SDR 208
 - internal 205
 - Membership 208

- Response 208
- resource variables 206
 - observation interval 206
 - reporting interval 206
- Resource Variables
 - IBM.PSSP.Membership.LANadapter.state 213
 - IBM.PSSP.Response.Host.state 209
 - IBM.PSSP.Response.Switch.state 209
- restore 375
- RFC 1416 143
- RFC 1508 143
- RFC 1510 142, 148
- RJ-45 48
- RMAPI
 - See Resource Monitor Application Programming Interface
- root.admin 172
- ROS
 - See Read-Only Storage
- rotating 434
- routing 50, 53
- RPM 480
- RS232 443
- RVSD Failover 407

S

- S70 112, 269
- s70d 113
- S7A 112, 269
- SAMI 112, 269
- SCHEDULER_API 512
- SCHEDULER_TYPE 512
- Scheduling Machine 497
- SCSI 280
- SDR 190, 192, 204
 - See also System Data Repository
- SDRCreateFile 104
- SDRCreateSystemFile 104
- SDRDeleteFile 104
- SDRReplaceFile 104
- SDRRetrieveFile 104
- secondary Kerberos servers 157
- secret key 146, 149
- secret password 143
- secure shell 145
- Security
 - ftp 141, 143
 - rcp 142, 143
 - rexec 142
 - rlogin 142
 - rsh 142, 143, 175
 - telnet 141, 143
 - xauth 145
 - xhost 144
- security administration 139
- security policy 139
- Security Server 168
- send 464
- Serial Network 443
- Service Processor 112
- service ticket 149, 160
- Services
 - kerberos 158
 - kerberos_admin 158
 - kerberos4 158
 - kerberos-adm 158
 - krb_prop 158
 - pmv2 456
- session key 151
- set_auth_method 143
- settokens.c 461
- setup_authent 164
- shell port 177
- Simple Network Management Protocol 329
- site environment 268
- SLIM 112, 269
- SMPI
 - See System Performance Measurement Interface
- SNMP
 - See Simple Network Management Protocol
- SP access control (SPAC) 340, 346
- SP Accounting 319
 - accounting class identifiers 321
 - Configuring 320
 - Node-exclusive-use accounting 320
 - Parallel job accounting 320
 - record consolidation 319
 - spacctnd 321
 - ssp.sysman fileset 319
 - user name space 323
- SP LAN 47
- SP Perspectives 116, 119, 293
 - Event Perspective 294
 - Hardware Perspective 293, 295
 - Launch Pad 293, 294
 - Performance Monitor Perspective 294

- SP Resource Center 294
- spmon -g 295
- system management tasks 293
- System Partitioning Aid 294
- Virtual Shared Disk Perspective 294
- SP Switch 215
 - Administration and Management 225
 - autojoin 237
 - autojoin attribute 231
 - Automatic Node Unfence 226, 231
 - automatic unfence 237
 - Centralized Error Logging 226
 - Clock Master 238
 - css.logevnt method 245
 - css.snap 240, 241, 245, 247
 - css.summlog 240, 241
 - cssadm 231, 235, 237
 - cssadm daemon 227, 228
 - Eannotator 222, 223
 - Eclock 238, 239
 - Eclock topology file 239
 - Efence 232
 - Error Logging 240
 - error logging daemon 240
 - error notification objects 242
 - Error recovery 217
 - Estart 217, 238
 - Eunfence 233
 - Fencing nodes 237
 - Global Shutdown 238
 - Intermediate Switch Board 218, 239
 - ISB 220
 - link 220
 - log consolidation 241
 - Log Files 247
 - logevnt.out 245
 - management 215
 - Node Switch Board 218, 239
 - operation 215
 - out.top 248
 - port number 221
 - primary backup node 217
 - primary node 217
 - rc.switch 235
 - Resource Variables 241
 - route table 224
 - Route Table Generator 216, 218, 223
 - routing table database 218
 - secondary node 218
 - Switch Admin Daemon 226
 - switch chip number 220
 - Switch Clocks 238
 - Switch Management 235
 - switch port number 221
 - switch scan 232
 - Switch Topology File 218
 - Switch-Dependent Application Startup 226, 235
 - switch-to-switch connection 221
 - swtlog 241
 - TBIC
 - See Trail Blazer Interface Chip
 - topology file 222
 - topology filename format 218
 - Trail Blazer Interface Chip 233
 - Unfencing nodes 237
 - user space protocol 215
 - Worm 216
- SP Switch frame 22
- SP Switch Router 39
- SP System Tuning 305
 - Adapter Receive Queue size 309
 - Adapter Transmit Queue size 309
 - Administrative Ethernet 306
 - chgcass 308
 - default transmit queue 310
 - External Networks 309
 - Large Configurations 305
 - Receive Queues 310
 - rpool and spool 307
 - rpoolsize and spoolsize parameters 307
 - SP Daemons 306
 - SP Switch 307
 - Subsystems 306
 - tcp_sendspace, tcp_recvspace 310
 - Transmit queues 309
 - Tunable Network Options 306
 - udp_sendspace, udp_recvspace 310
- SP user management (SPUM) 339, 341
- SP_ports 111
- SP1 5
- SP2 6
- SP-Attached Server 35
- spdata 264
- spk4rsh 176, 178
- SPMI
 - See System Performance Measurement Interface

- spmon 119
- SPOT 260
- spsitenv 133
 - consensus 133
 - internet 133
 - none 133
- SSA 280
- standalone 258
- state 111
- stratum 132
- Stripe Group Manager 413
- Structured Socket Messages 455
- Submit-only Machine 498
- subnet 50
- Subsystems
 - Emonitor 237
 - Event Management 296
 - sdrd 91, 98
 - swtadmd 227
 - swtlog 241
- supercomputer 4
- supervisor card 24
- supervisor microcode 271
- switch clock 287
- switch responds 290
- switch topology 286
- symmetric encryption 146, 149
- Symmetric Multiprocessor 463
- syncd 445
- synchronization 444, 447
- Sysctl 180
- syslogd 111
- SYSTEM 147
- SYSTEM callback 181
- system characteristics 6
- System Data Repository 91, 104
 - /etc/inittab 100
 - /etc/rc.sp 100
 - /etc/SDR_dest_info 98, 100, 101
 - /spdata/sys1/sdr/system/locks directory 102
 - Adapter class example 93
 - Adding AIX Files to the 104
 - alias IP address 98
 - Attributes 92
 - Backup 107
 - changing an object 104
 - Classes 92
 - default partition 94
 - Locking 102
 - Manipulating SDR Data 103
 - multiple SDR daemons 102
 - Objects 92
 - Partitioned classes 94
 - Restore 107
 - SDR Daemon 98
 - SDR Data Model 92
 - SDR directory structure 93, 95
 - SDR Log File 106
 - SDR_test 105
 - SDRArchive 103, 107
 - SDRChangeAttrValues 104, 106
 - SDRClearLock 102, 106
 - sdrd 91, 98, 99
 - SDRGetObjects 105
 - SDRListClasses 106
 - SDRRestore 103, 107
 - SDRWhoHasLock 102, 106
 - Shadow Files 103
 - SP class example 93
 - SP_NAME 100
 - Syspar_map class 100
 - syspar_name 100
 - System classes 94
 - System partitioning and the 91
 - User Interface 102
- System Monitor
 - command line 304
 - system hardware 304
- System Monitoring 293
- System Partitioning 95
 - /etc/inittab 100
 - /etc/rc.sp 100
 - /etc/SDR_dest_info 99, 100, 101
 - alias IP address 98
 - default IP address 99
 - default partition 98
 - Partitioning Rules 96
 - primary partition 98
 - sdrd 99
 - SP_NAME 100
 - syspar_addr attribute 99
 - Syspar_map class 99, 100
 - syspar_name 100
 - Why Partition? 97
- System Performance Measurement Interface 188, 206
- System Resource Controller 237

T

- Tcl 180
- thewall 263
- Thin-wire Ethernet 47
- ticket cache 154, 173
- ticket cache file 159
- ticket forwarding 176
- Ticket-Granting Service 149, 150
- Ticket-Granting Ticket 149, 151, 160, 284
- Tivoli 7
- TMSCSI 443
- TMSSA 443
- TOD 236
- Topology 253
- Topology Services 187
 - Adapter Membership 189, 190
 - Adapter Membership Group 189, 190, 191, 192
 - Crown Prince 190, 191
 - DEATH_IN_FAMILY 191
 - Group Leader 190, 191
 - hatsd 188
 - heartbeat 191, 193
 - frequency 191
 - sensitivity 191
 - machines.lst 190, 191, 192
 - Mayor 190
 - Neighbor 191
 - Network Connectivity Table 189, 192
 - Node Membership 189
 - Prepare To Commit 192
 - Proclaim Packet 191
 - Reliable Messaging 189, 201
 - Singleton group 189, 191
 - topology table 192
- TP
 - See Twisted Pair
- Trailblazer 6
- transmit queue size 263
- trusted third party system 140
- trusted third-party 148
- Twisted Pair 48

U

- UDP 189
- UDP/IP 193
- UNIX domain sockets 202
- UNIX Domain Stream 189

- UNIX mode bits 140
- Unshielded Twisted Pair 48
- uplink 54
- user database 140
- UTP
 - See Unshielded Twisted Pair

V

- Vampir 488, 491
- VampirTrace 491
- Virtual Front Operator Panel 172
- Virtual Front Operator Panel (VFOP) 119
- Virtual Shared Disk 395
- Virtual Shared Disk Perspective 304
 - /etc/sysctl.acl 304
 - /etc/sysctl.vsd.acl 304
 - shared disk management 304
 - shared disk management tasks 304
 - spvsd command 304
- VSD 96, 250
- VSD States 403
- Vulcan 5

W

- wall_clock_limit 511
- Wladawsky-Berger, Irving 5
- workstation cluster 463
- Worm 218

X

- X Windows 144
- Xauthority 144
- XL Fortran 479
- XL High Performance Fortran 478
- XL HPF 479
- xntpd 134
- xntpd.c 134

546 The RS/6000 SP Inside Out

ITSO Redbook Evaluation

The RS/6000 SP Inside Out
SG24-5374-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at <http://www.redbooks.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!)

SG24-5374-00
Printed in the U.S.A.

The RS/6000 SP Inside Out



SG24-5374-00